

SEP

SECRETARÍA DE
EDUCACIÓN PÚBLICA



TECNOLÓGICO NACIONAL DE MÉXICO



INSTITUTO TECNOLÓGICO DE CD. GUZMÁN

TITULACIÓN INTEGRAL
TESIS PROFESIONAL

TEMA:

**DISEÑO DE UNA PLATAFORMA PARA
LA IMPLEMENTACIÓN DE CONTROL
EN CUATRO MOTORES_2**

QUE PARA OBTENER EL TÍTULO DE:

INGENIERA EN ELECTRÓNICA

PRESENTA:

CASTAÑEDA AVIÑA PERLA RUBI

ASESOR(A):

MIE. JOSÉ MARÍA HERNÁNDEZ OCHOA

CD. GUZMÁN JALISCO, MÉXICO, ENERO DE 2017

Cd. Guzmán, Municipio de Zapotlán el Grande, Jal, 27/ENERO/2017

ASUNTO: Liberación de Proyecto para Titulación Integral.

M.C. FAVIO REY LUA MADRIGAL
JEFE DE LA DIVISION DE ESTUDIOS PROFESIONALES
PRESENTE




Por este medio le informo que ha sido liberado el siguiente proyecto para la Titulación Integral:

Nombre del Egresado:	PERLA RUBI CASTAÑEDA AVIÑA
Carrera:	INGENIERIA ELECTRONICA
No. De Control:	12290344
Nombre del Proyecto:	DISEÑO DE UNA PLATAFORMA PARA LA IMPLEMENTACION EN CUATRO MOTORES_2
Producto:	TITULACION INTEGRAL

Agradezco de antemano su valioso apoyo en esta importante actividad para la formación profesional de nuestros egresados.

ATENTAMENTE


M.E.H. MARCO ANTONIO SOSA LOPEZ
JEFE DEL DEPTO. ELECTRICA Y ELECTRONICA

		
M.I.E. JOSE MARIA HERNANDEZ OCHOA Asesor	M.I.E. CARLOS ENRIQUE MACIEL GARCIA Revisor	M.I.E. FAVIO REY LUA MADRIGAL Revisor

C.p.expediente
JRGV/MASL/adc




S.E.P. TecNM
INSTITUTO TECNOLÓGICO
DE CD. GUZMAN
DEPTO. ELECTRICA Y
ELECTRONICA
Av. Tecnológico No. 100 C.P. 49100 A.P. 150
Cd. Guzmán, Jal. Tel. Conmutador (341) 5 75 20 50
www.itcg.edu.mx



www.itcg.edu.mx/Sistemas de Gestión/Calidad

AGRADECIMIENTOS

Primeramente quiero agradecer a Dios por darme la fuerza para lograr mis objetivos, al Instituto Tecnológico de Ciudad Guzmán por brindarme formación académica y las herramientas necesarias para desarrollarme ampliamente en el ámbito profesional.

A mis padres Alma y Humberto, por su apoyo constante e incondicional y por darme la confianza, comprensión y los valores para perseverar y realizar todo lo que me proponga.

A los profesores de la institución por su disposición a ayudar a los alumnos a resolver dudas y cuestionamientos.

A mis hermanos Brenda y Humberto por su apoyo y ayuda. A Alejandro por creer en mí y brindarme su comprensión y respaldo en todo momento.

RESUMEN EJECUTIVO

En este proyecto se elaboró una plataforma de cuatro motores o cuadricóptero, abarcando cada paso desde el diseño y armado del frame hasta la interconexión y comunicación entre sus componentes, por medio de una tarjeta Arduino Mega 2560. Se desarrollaron cada una de las etapas que componen el programa de un controlador PID para el control y la estabilización del cuadricóptero, esto se realizó debido a que la institución no cuenta con un prototipo en el que puedan observarse de forma interactiva los efectos de cada una de las acciones de control: proporcional, integral y derivativa; en un sistema lazo cerrado. El presente proyecto es una contribución a los equipos electrónicos pertenecientes al área del departamento de Ingeniería electrónica, debido a que brinda la oportunidad a los estudiantes de desarrollar algoritmos de control para posteriormente implementarlos en el cuadricóptero, esto significa una ventaja ya que facilita el análisis de la respuesta de un sistema de control, respecto a otros equipos en los que no son tan observables los efectos del controlador.

ABSTRACT

In this project has been elaborated a four motor platform or quadcopter including every step from the design and construction of the frame to the interconnection and communication between its components through an Arduino Mega 2560 board. There were developed all of the steps that compose the PID controller program for the control and stabilization of the quadcopter, this was done because the institution doesn't count with a prototype in which can be observed interactively all the effects of control actions: proportional, integral and derivative in a closed loop system. This project represents a contribution to the electronic equipment that belong to the area of the electronic engineering department, because it offers the students the opportunity to develop control algorithms for a later implementation in the quadcopter thus means an advantage as it facilitates the control system response analysis respect to other equipment in which the effects of the controller are not as observable.

ÍNDICES

ÍNDICE DE CONTENIDO

AGRADECIMIENTOS	I
RESUMEN EJECUTIVO	III
ABSTRACT	III
ÍNDICES	IV
INTRODUCCIÓN	XII
CAPÍTULO I. GENERALIDADES DEL PROYECTO	1
1.1 ANTECEDENTES DEL PROYECTO	2
1.2 PLANTEAMIENTO DEL PROBLEMA	3
1.3 JUSTIFICACIÓN	3
1.4 OBJETIVOS	3
1.4.1 Objetivo General	3
1.4.2 Objetivos Específicos	3
1.5 HIPÓTESIS	4
1.6 ALCANCES	5
1.7 LIMITACIONES	5
1.8 CRONOGRAMA DE ACTIVIDADES	6
CAPÍTULO II. MARCO TEÓRICO	9
2.1 VEHÍCULO AÉREO NO TRIPULADO (UAV)	10
2.1.2 Ventajas Y Desventajas Entre Los UAV Multirotor	11
2.2 CUADRICÓPTERO	12
2.2.1 Ventajas Y Desventajas de un Cuadricóptero	12
2.2.2 Principio De Operación	13
2.3 SISTEMA MECÁNICO DE UN CUADRICÓPTERO	16
2.3.1 Frame	16
2.3.2 Hélices	17
2.4 SISTEMA ELECTRÓNICO DE UN CUADRICÓPTERO	18
2.4.1 Motores De Corriente Directa Sin Escobillas (Brushless)	18

2.4.2	Variador De Velocidad Electrónico (Esc).....	21
2.4.3	Batería	22
2.4.4	Sensores Dinámicos.....	24
2.4.5	Giroscopio Y Acelerómetro IMU (Inertial Measurement Unit)	24
2.4.6	Protocolo De Comunicación	28
2.4.7	Filtro Complementario	28
2.5	SISTEMA DE CONTROL DE UN CUADRICÓPTERO.....	29
2.5.1	Arduino	29
2.5.2	Algoritmos De Control.....	30
2.6	SISTEMA DE COMUNICACIÓN MEDIANTE RADIOFRECUENCIA	34
CAPÍTULO III. CARACTERIZACIÓN DEL ÁREA DE TRABAJO		36
3.1	GENERALIDADES DE LA EMPRESA	37
3.2	ANTECEDENTES DE LA EMPRESA	37
3.3	MISIÓN	38
3.4	VISIÓN	38
3.5	OBJETIVOS	39
3.6	VALORES	39
3.7	POLÍTICA DE CALIDAD	40
3.8	DESCRIPCIÓN DEL ÁREA DE NEGOCIOS.....	40
3.9	PUESTO ASIGNADO Y FUNCIONES	40
CAPÍTULO IV. DESCRIPCIÓN DE LAS ACTIVIDADES DESARROLLADAS		41
4.1	ACTIVIDADES DEL PROYECTO	42
4.1.1	Diseño E Implementación Del Frame	42
4.1.2	Elección De Los Componentes Del Cuadricóptero.....	45
4.1.3	Prueba A Las Hélices	58
4.1.4	Calibración Del Sistema De Comunicación De Radiofrecuencia	59
4.1.5	Calibración De Los ESC'S Y Prueba A Los Motores	67
4.1.6	Calibración IMU	71
4.1.7	Encendido Y Control Simultáneo De Velocidad En Cuatro Motores.....	78
4.1.8	Alarma De Bajo Voltaje En La Batería	82
4.1.9	Controlador proporcional-integral-derivativo (PID).....	87

4.1.10	Control del cuadricóptero en lazo cerrado	90
4.2	ACTIVIDADES ADICIONALES	98
CAPÍTULO V. RESULTADOS OBTENIDOS		99
5.1	RESULTADOS DEL CONTROLADOR PID	100
CAPÍTULO VI. CONCLUSIONES Y RECOMENDACIONES		108
6.1	CONCLUSIONES.....	109
6.2	RECOMENDACIONES	113
6.3	REFERENCIAS BIBLIOGRÁFICAS	114
6.4	ANEXOS	117
6.4.1	Programa De Prueba Para El Sistema De Radiofrecuencia	117
6.4.2	Programa Para Calibración De La IMU	119
6.4.3	Programa Para El Encendido Y Control Simultáneo De Velocidad En Cuatro Motores	120
6.4.4	Prueba A La Alarma De Bajo Voltaje En La Batería	121
6.4.5	Cálculo De Salidas Para Controlador proporcional-integral-derivativo (PID)	122
6.4.6	Cálculo De Los Pulsos Enviados A Cada ESC, Según El Movimiento Del Cuadricóptero	122
6.4.7	Lazo Cerrado De Control.....	123

ÍNDICE DE TABLAS

Tabla 1. Tabla comparativa entre diversos modelos de baterías.	23
Tabla 2. Especificaciones del motor brushless A2212	46
Tabla 3. Especificaciones del ESC 30 A	48
Tabla 4. Especificaciones de las hélices 1045	49
Tabla 5. Especificaciones de la batería	50
Tabla 6. Especificaciones del cargador de batería	52
Tabla 7. Especificaciones del TGY-i6.....	54
Tabla 8. Especificaciones del MPU6050.	55
Tabla 9. Especificaciones del Arduino Mega.....	57

ÍNDICE DE FIGURAS

Figura 2.1. UAV de uso militar.....	10
Figura 2.2. Diagrama de fuerzas de un cuadricóptero.....	13
Figura 2.3. Movimiento Yaw	14
Figura 2.4. Movimiento Pitch	14
Figura 2.5. Movimiento Roll	15
Figura 2.6. Movimiento vertical.....	15
Figura 2.7. Frame de un cuadricóptero (Álvarez Salgado & Rodríguez Lira , 2015).....	16
Figura 2.8. Ejemplo de hélices (Faría Hernández, (n.d.)).....	17
Figura 2.9. Esquema de un motor de cc sin escobillas que ilustra el principio de operación.....	18
Figura 2.10. Motor Brushless.....	20
Figura 2.11. Ejemplo de un ESC	21
Figura 2.12. Ejemplo de una batería Li-Po	23
Figura 2.13. MPU-6050 Arduino.....	24
Figura 2.14. Esquema de construcción de un acelerómetro.	25
Figura 2.15. Representación gráfica del comportamiento de un acelerómetro.....	26
Figura 2.16. Representación del efecto coriolis.....	27
Figura 2.17. Tarjetas de Arduino. (Vicedo, 2014).....	29
Figura 2.18. Sistema en lazo cerrado con controlador PID	31
Figura 2.19. Respuesta de controladores P, PI y PID	33
Figura 2.20. Esquema del controlador PID, aplicado a un sistema de control en lazo cerrado.....	34
Figura 2.21. Transmisor de radio	Figura 2.22. Receptor de radio . 35
Figura 4.1. Base del cuadricóptero.....	43
Figura 4.2. Brazo del cuadricóptero.....	43
Figura 4.3. Cubierta del cuadricóptero.	44
Figura 4.4. Ensamblaje del cuadricóptero.	44
Figura 4.5. Motor y accesorios	47
Figura 4.6. ESC.....	48

Figura 4.7. Hélices con sus respectivos adaptadores de orificio	49
Figura 4.8. Batería.....	51
Figura 4.9. Transmisor (der.) y Receptor (izq.) RF Turnigy-i6	53
Figura 4.10. IMU MPU-6050 (Arduino)	55
Figura 4.11. Arduino Mega 2560 (Arduino.cl)	56
Figura 4.12. Prueba de balance en las hélices.....	59
Figura 4.13. Diagrama pinout de la tarjeta Arduino Mega (bigtronica)	60
Figura 4.14. Ancho de pulso para cada canal	61
Figura 4.15. Diagrama de conexión entre Arduino Mega y receptor de radio..	62
Figura 4.16. Conexión física entre Arduino Mega y receptor de radio.....	62
Figura 4.17. Datos correspondientes a posición central.....	63
Figura 4.18. Datos correspondientes a la posición de las palancas	64
Figura 4.19. Datos correspondientes a la posición de las palancas	64
Figura 4.20. Datos correspondientes a la posición de las palancas.....	65
Figura 4.21. Datos correspondientes a la posición de las palancas	65
Figura 4.22. Diagrama para observar los pulsos en la entrada del receptor en el osciloscopio	66
Figura 4.23. Conexiones para observar los pulsos de entrada del receptor (1500µs)	66
Figura 4.24. Pulso mínimo de entrada canal 1 (1000 µs).....	67
Figura 4.25. Pulso mínimo de entrada canal 1 (1000 µs).....	67
Figura 4.26. Calibración de ESC con palanca de arranque en posición máxima	67
Figura 4.27. Calibración de ESC con palanca de arranque en posición máxima	68
Figura 4.28. Diagrama de conexión para calibración de ESC	68
Figura 4.29. Diagrama de conexión para probar los motores.....	69
Figura 4.30. Conexión para probar los motores	69
Figura 4.31. Ancho de pulso con palanca de arranque en posición mínima ...	70
Figura 4.32. Ancho de pulso con palanca de arranque en posición media	70
Figura 4.33. Ancho de pulso con palanca de arranque en posición máxima ...	70
Figura. 4.34. Representación gráfica de la IMU (Robologs).....	72

Figura 4.35. Diagrama de conexión entre Arduino Mega y la IMU MPU-6050.	74
Figura 4.36. Conexión física entre Arduino Mega y la IMU MPU-6050.	74
Figura 4.37. Diseño para la medición de los ángulos.	75
Figura 4.38. Datos correspondientes a la posición de la IMU.....	75
Figura 4.39. Datos correspondientes a la posición de la IMU.....	76
Figura 4.40. Datos correspondientes a la posición de la IMU.....	76
Figura 4.41. Datos correspondientes a la posición de la IMU.....	77
Figura 4.42. Datos del movimiento yaw. a) IMU estabilizado, b) IMU en movimiento anti-horario c) IMU en movimiento horario.	78
Figura 4.43. PCB de la tarjeta distribuidora de potencia	80
Figura 4.44. Diagrama de conexión para el encendido de cuatro motores simultáneamente	81
Figura 4.45. Conexión física para el encendido de cuatro motores simultáneamente	81
Figura 4.46. Conexión física del divisor de voltaje con el Arduino mega.....	83
Figura 4.47. Lectura de datos con el voltaje de la batería a) Voltaje máximo b) Voltaje mínimo.....	84
Figura 4.48. Diagrama de conexión para la alarma de bajo voltaje en la batería	85
Figura 4.49. Prueba de la alarma de bajo voltaje en la batería.	86
Figura 4.50. PCB para conexión de componentes con la tarjeta Arduino	86
Figura 4.51. Diagrama de flujo del proceso de control	90
Figura 4.52. Configuración de los ejes del MPU-6050 (Arduino).....	92
Figura 4.53. Configuración de motores y ejes de movimiento.....	93
Figura 4.54. Diagrama final de conexión del cuadricóptero en lazo cerrado ...	97
Figura 4.55. Ensamblaje completo del cuadricóptero en lazo cerrado y conexión física.....	97
Figura 4.56. Simulador de vuelo giroscópico.....	98
Figura 5.1. Constantes del controlador PID, primera prueba	101
Figura 5.2. Constantes del controlador PID, primera prueba	102
Figura 5.3. Constantes del controlador PID, segunda prueba	102
Figura 5.4. Constantes del controlador PID, tercera prueba.....	103

Figura 5.5. Constantes del controlador PID, cuarta prueba.....	104
Figura 5.6. Constantes del controlador PID, quinta prueba.....	104
Figura 5.7. Constantes del controlador PID, sexta prueba	105
Figura 5.8. Estabilización del cuadricóptero con error.....	106
Figura 5.9. Constantes del controlador PID, séptima prueba	106
Figura 5.10. Cuadricóptero nivelado aun con perturbaciones	107

INTRODUCCIÓN

El presente trabajo muestra el proceso realizado para el diseño y puesta en marcha de un cuadricóptero para la implementación de algoritmos de control PID a modo de prácticas por alumnos del Instituto Tecnológico de Ciudad Guzmán.

El control permite prever y corregir las desviaciones que puedan darse antes, durante y tras realizar una actividad según unos estándares establecidos con anterioridad. El sistema de control permite una operación del proceso más fiable y sencilla, al encargarse de obtener unas condiciones de operación estables, y corregir toda desviación que se pudiera producir en ellas respecto a los valores de ajuste. (Moreno, 2013).

El presente trabajo está estructurado en 6 capítulos, en el capítulo I se describen los motivos que desencadenaron el desarrollo del proyecto, se describen los problemas que se pretenden resolver con el desarrollo del proyecto, a su vez se establecen los objetivos a lograr y se delimita el alcance del proyecto. También se presenta una organización estructurada del desarrollo del proyecto.

El capítulo II está compuesto por los conceptos técnicos necesarios para desarrollar de manera óptima el proyecto, conceptos tales como métodos, definiciones, técnicas y modo de funcionamiento de los componentes del cuadricóptero.

El capítulo III consta de la información descriptiva acerca de la empresa en la que se desarrollara el proyecto. En el capítulo IV, se hace referencia a todas las actividades desarrolladas en el transcurso del proyecto, estas se encuentran divididas en etapas como lo son:

- La elección de los materiales
- Diseño y construcción del frame
- Establecimiento y pruebas para la comunicación mediante radiofrecuencia
- Obtención de valores y calibración de la Unidad de Medición Inercial (IMU)

- Calibración, pruebas y encendido simultaneo de los cuatro motores del cuadricóptero
- Desarrollo del controlador PID mediante la plataforma de programación Arduino
- Condiciones de inicio del cuadricóptero y cálculo de los pulsos enviados a cada variador de velocidad (ESC) para provocar el movimiento del cuadricóptero
- Unión de todas las partes para crear un lazo de control en lazo cerrado que permita la estabilidad del cuadricóptero y la autocorrección de acuerdo a parámetros establecidos anteriormente.

En el capítulo V se muestra el análisis de los resultados obtenidos, poniendo en evidencia las pruebas al funcionamiento del controlador proporcional-integral-derivativo (PID) y como es que se logra paso a paso la calibración de este. Respecto a esto se logró alcanzar el 100% de los resultados planteados como objetivos.

En el capítulo VI quedan establecidas las conclusiones obtenidas del desarrollo y el análisis previo de los resultados. A su vez, se expresan las recomendaciones que surgieron durante el desarrollo del proyecto para facilitar su probable continuación.

CAPÍTULO I. GENERALIDADES DEL PROYECTO

1.1 ANTECEDENTES DEL PROYECTO

El proyecto se va a desarrollar en el Departamento de Eléctrica y Electrónica del Instituto Tecnológico de Cd. Guzmán, este departamento cuenta con la infraestructura necesaria para crear condiciones óptimas para el desarrollo de investigación (básica y/o aplicada) en la línea que se propone, tales como:

- Laboratorio de Cómputo totalmente equipado.
- Infraestructura de red (cableada e inalámbrica) adecuada a las actividades de investigación y docencia.
- Suscripciones a diversas bases de datos, mismas que pueden ser consultadas vía internet.
- Dos edificios correspondientes al área de eléctrica y electrónica con diversas áreas y laboratorios.
- Aulas equipadas para la proyección de materiales digitales y con conectividad inalámbrica.

La idea de desarrollar este proyecto fue propuesta por un cuerpo académico del área de Electrónica y surgió debido a que el departamento de Electrónica no cuenta con un dispositivo en el que los alumnos puedan probar los algoritmos de control implementados en clase y obtener resultados observables de manera interactiva de la estabilidad o inestabilidad que se produce en un sistema ante la variación de las acciones de control.

A través de este proyecto se pretende ayudar a consolidar la especialidad de Mecatrónica en la carrera de Ingeniería Electrónica que ofrece el Instituto Tecnológico de Cd Guzmán, la cual aún no cuenta con material suficiente para realización de prácticas en cuestión de robótica. Además, contribuye a impulsar que la carrera de Ingeniería Electrónica mantenga su acreditación.

1.2 PLANTEAMIENTO DEL PROBLEMA

El problema a resolver consiste en que la institución no cuenta con algún dispositivo en el que se apliquen de manera tangible los algoritmos de control.

1.3 JUSTIFICACIÓN

La construcción de esta plataforma se propone motivar a los alumnos a interesarse en la generación de estos algoritmos y a realizar la experimentación necesaria sin riesgo a dañar ningún dispositivo. La construcción de esta plataforma se realizará a través de modelado en 3D y estos diseños se imprimirán en materiales ligeros los cuales permitan estabilidad y maniobrabilidad, esta plataforma contará con cuatro motores los cuales deberán mantener la plataforma estable mediante algoritmos de control.

1.4 OBJETIVOS

1.4.1 Objetivo General

Diseñar, implementar y probar una plataforma autónoma de cuatro motores, utilizando técnicas de teoría de control para mantener su estabilidad.

1.4.2 Objetivos Específicos

- Realizar el diseño de la plataforma autónoma (cuadricóptero), tomando en cuenta el aerodinamismo de la misma.
- Implementar la plataforma, mediante modelado 3D en materiales ligeros que favorezcan la ligereza de la aeronave.

- Elegir los componentes necesarios para implementar el cuadricóptero tomando en cuenta su relación precio-calidad y procurando elegir materiales que permitan realizar modificaciones al proyecto así como agregar nuevas funcionalidades en el futuro.
- Probar los componentes del cuadricóptero de manera individual para asegurar su correcto funcionamiento, previo a su interconexión con otros componentes del cuadricóptero.
- Desarrollar e implementar el programa de control del cuadricóptero.
- Realizar pruebas al programa para comprobar su correcto funcionamiento y asegurar que controle los movimientos del cuadricóptero manteniendo su estabilidad.
- Implementar una plataforma cuya función sea la de un simulador giroscópico de vuelo que brinde soporte al cuadricóptero para realizar las pruebas necesarias sin causar daños a su estructura.

1.5 HIPÓTESIS

El uso de una plataforma para la implementación de un controlador PID en cuatro motores permite a los estudiantes desarrollar habilidades para probar los algoritmos de control.

1.6 ALCANCES

Diseñar y construir una plataforma autónoma de cuatro motores para implementar en ella diferentes algoritmos de control. Se realizarán una variedad de diseños para la plataforma y se optara por la mejor, se probará en diversas condiciones de operación y mediante varios algoritmos de control, para posteriormente analizar su eficiencia y realizar una comparación encontrando sus condiciones de operación óptimas.

1.7 LIMITACIONES

En el transcurso del proyecto se presentaron algunas limitaciones como el que los materiales tardaron mucho tiempo en llegar lo que ocasiono que la realización de las pruebas se retrasaran, a pesar que la mayoría de los materiales llegaron hicieron falta varios como el conector de la batería y el cargador del mismo por lo que se optó por utilizar una fuente. El armado de la plataforma para las pruebas fue rediseñado por inconvenientes de fricción los cuales podrían causar problemas en las pruebas de vuelo. También se detectó una falla en uno de los motores por lo que se decidió utilizar otro, también se realizaron varias placas shield para el arduino mega 2560 debido a inconvenientes con los dispositivos y sus dimensiones.

1.8 CRONOGRAMA DE ACTIVIDADES

Actividad\mes	Agosto				Septiembre				Octubre				Noviembre			
Revisión del estado del arte	X	X														
Armado y calibración de la impresora 3D	X	X														
Elección de los materiales			X													
Elaboración de diseños para el frame			X	X												
Construcción del diseño del cuadricóptero				X	X											
Armado del cuadricóptero					X	X										
Calibración de las hélices						X										
Prueba del radiocontrol						X										
Prueba del IMU							X									

Diseño y fabricación de la placa distribuidora de potencia							X										
Calibración de los ESC's							X										
Prueba de la respuesta de los motores con el radiocontrol								X									
Programación y prueba de la alarma de bajo voltaje de la batería								X									
Pruebas de funcionamiento sin el control PID									X								
Diseño y fabricación de la placa shield para el arduino										X							
Crear algoritmos de control												X					

Prueba de algoritmos de control PID											X	X	X		
Redacción reporte de residencia profesional				X				X				X		X	X

CAPÍTULO II. MARCO TEÓRICO

2.1 VEHÍCULO AÉREO NO TRIPULADO (UAV)

Un vehículo aéreo no tripulado por sus siglas en inglés UAV (Unmanned Aerial Vehicle), es un vehículo cuyo control puede ser autónomo o comandado por un operador en tierra (Castañeda García, Henao , & Valencia , 2016). Los UVA se aplican en situaciones que implican peligros para vehículos tripulados, como pueden ser incendios, aplicaciones militares, así como inspección de terrenos, líneas eléctricas y sitios de difícil acceso.

Las principales ventajas de un UAV son: bajo costo de operación respecto a una aeronave tripulada, facilidad de manejo, al no tener tripulantes puede acceder a sitios riesgosos sin arriesgar vidas humanas y puede incorporar muchos sensores. En la figura 2.1 se muestra un UAV de uso militar.



Figura 2.1. UAV de uso militar

De acuerdo al tipo de control que emplea un UAV y como menciona (Olea Hernández & Sánchez López, 2015) se pueden clasificar en:

- **Autónomo y adaptativo:** Es absolutamente gobernado por sus sistemas abordo, sin la intervención de un operador, este tipo de UAV puede replanificar su vuelo en base a los cambios ocurridos en su entorno.
- **Monitorizado:** Su operación es semiautónoma, el operador controla la retroalimentación de la aeronave, es decir, sus decisiones.
- **Supervisado:** El control recae mayormente en el operador, ya que a aeronave lleva a cabo pocas funciones de manera autónoma.
- **Autónomo no adaptativo:** Sigue una rutina preprogramada por lo cual no es capaz de cambiar esa rutina para adaptarse a los cambios de su entorno.
- **Mando directo por un operador:** Responde directamente a los mandos del operador.

2.1.2 Ventajas Y Desventajas Entre Los UAV Multirotor

- **Tricopter:** Construcción simple y económica, sin embargo tiene bajo empuje así como menor estabilidad y tiempo de vuelo.
- **Cuadricóptero:** Mecánicamente más simple que el tricóptero ya que tiene un mayor empuje (considerando aeronaves del mismo peso), tiene mayor estabilidad y tiempo de vuelo.
- **Hexacopter:** Mayor potencia, capacidad de carga que un cuadricóptero. Si hay una falla en un motor la aeronave puede aterrizar sin daños (redundancia). Su desventaja es que su tamaño es mayor al igual que su precio.

- Octocopter: Mantiene todas las ventajas del Hexacopter, sin embargo sus desventajas son que requieren más energía para mantener el vuelo y su precio es muy elevado (Morales Delgado, 2015).

El cuadricóptero es muy maniobrable respecto a otros multirrotores, debido a su estructura y la configuración de sus motores, tiene características de estabilidad y buena capacidad de carga por ello se eligió este tipo de UAV para el proyecto.

2.2 CUADRICÓPTERO

Un cuadricóptero o en inglés quadrotor se define en el artículo de (Álvarez Salgado & Rodríguez Lira , 2015) como una aeronave no tripulada que se eleva y se desplaza mediante el movimiento de cuatro motores, con sus respectivas hélices, colocados en los extremos de una estructura (generalmente en forma de cruz), que le proporciona la tracción necesaria para su vuelo. La estabilidad y los movimientos del cuadricóptero se controlan mediante la regulación de la potencia enviada a los motores, esto se refleja en la variación de velocidad de giro de las hélices.

2.2.1 Ventajas Y Desventajas de un Cuadricóptero

Este tipo de aeronaves no tripuladas poseen cualidades de maniobrabilidad, así como una alta relación carga/peso ya que vehículos pequeños pueden llevar cargas mayores a su propio peso.

Sin embargo una de sus desventajas es que tienen un bajo nivel de autonomía ya que requieren ser alimentados mediante baterías. Otra de sus desventajas es el sistema de control y estabilidad de estas aeronaves ya que se requiere incorporar mecanismos de estabilización.

2.2.2 Principio De Operación

Un cuadricóptero es definido en el artículo de (Olea Hernández & Sánchez López, 2015) como una aeronave de seis grados de libertad ($x, y, z, \text{pitch}, \text{roll}$ y yaw). Los rotores originan las fuerzas y momentos que actúan en el cuadricóptero.

Para balancear el torque total de la aeronave, la estabilidad y el par motor se requiere que dos pares de motores roten en sentido horario, mientras que los otros dos deben rotar en sentido anti horario. Como se observa en la figura 2.2 se utilizan dos sistemas de referencia el sistema “T” o a la tierra y el sistema “B” o a la propia aeronave.

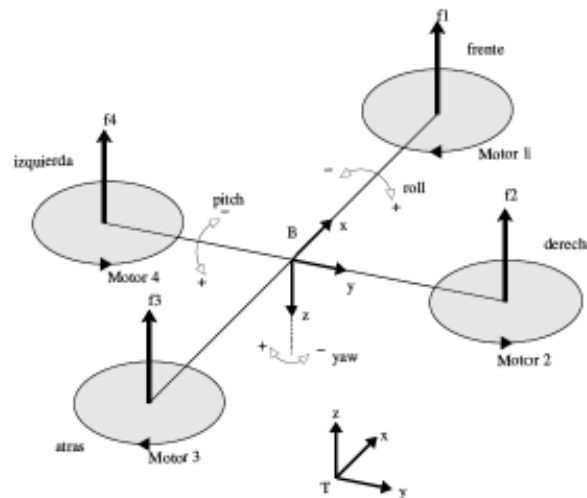


Figura 2.2. Diagrama de fuerzas de un cuadricóptero.

En el artículo de (Álvarez Salgado & Rodríguez Lira , 2015) se describen los movimientos del cuadricóptero, la combinación de estos tres movimientos es lo que permite maniobrar al cuadricóptero.

- Yaw:

Es el movimiento que se origina cuando el cuadricóptero gira sobre su eje vertical (Figura 2.3). Se logra incrementando por igual la potencia de giro de los motores uno y tres y disminuyendo la potencia de los motores dos y cuatro, en consecuencia se incrementa el par motor creando un giro contrario a las hélices que rotan con menor potencia.

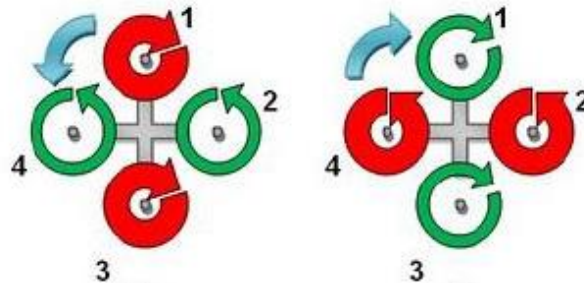


Figura 2.3. Movimiento Yaw

- Pitch:

Este movimiento origina el desplazamiento hacia adelante y atrás, para lograr esto la aeronave mantiene la potencia en el rotor uno (opuesto al sentido deseado), reduce al mínimo la del rotor y mantiene los dos motores restantes a potencia media, de esta forma la sustentación del motor uno provoca que la aeronave se incline y se desplace en el sentido deseado (Figura 2.4).

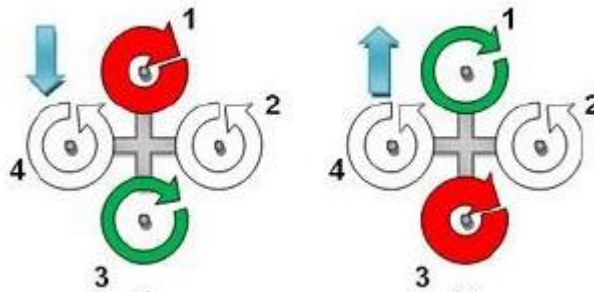


Figura 2.4. Movimiento Pitch

- Roll:

Este movimiento origina los desplazamientos a la derecha y a la izquierda, aplica el mismo principio del pitch pero lateralmente (Figura 2.5). Roll y pitch son movimientos en torno a los ejes horizontales del cuadricóptero, la inclinación de estos ejes provoca un movimiento lineal en el plano horizontal cuya velocidad depende del ángulo (ángulo de ataque), mientras que la dirección depende de la orientación del cuadricóptero.

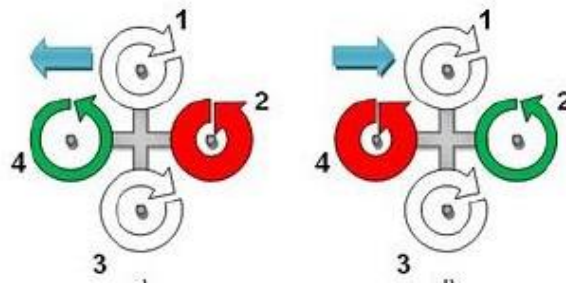


Figura 2.5. Movimiento Roll

- Vertical:

Para provocar un ascenso, descenso o mantener el vuelo estacionario de un cuadricóptero, es necesario que la fuerza de sustentación generada por los cuatro motores con sus respectivas hélices sea mayor que la fuerza gravitatoria (Figura 2.6). El ascenso y descenso de la aeronave, se produce al variar la potencia de los cuatro motores en igual magnitud para no modificar los otros grados de libertad.

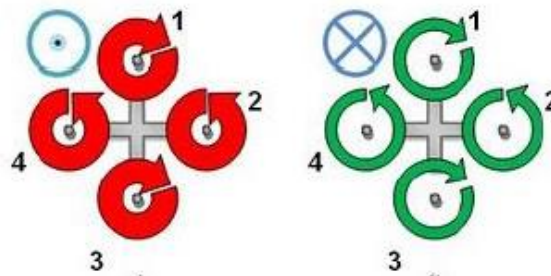


Figura 2.6. Movimiento vertical

2.3 SISTEMA MECÁNICO DE UN CUADRICÓPTERO

2.3.1 Frame

El frame, marco o armazón es la parte del cuadricóptero que se encargara de soportar el peso de los componentes y tener una estructura aerodinámica. El frame, como el mostrado en la figura 2.7 tendrá la forma de una cruz con una base octagonal funcionando también como una caja para nuestras piezas sensibles a los cambios climáticos.



Figura 2.7. Frame de un cuadricóptero (*Álvarez Salgado & Rodríguez Lira , 2015*)

Existen una gran variedad de estructuras para el frame una de ellas son mencionadas en el artículo de (José, 2015) como los son en “+”, en “X”, en “Y” o en “H”. Los más utilizados son en “+” y en “X” debido a que los brazos del frame son iguales y el control se encuentra en el centro del gravedad del frame.

Los materiales con los que se puede hacer un frame varía desde frames caseros hechos de madera hasta frames comerciales hechos de fibra de carbono pasando por los hechos en plástico por impresoras 3D como menciona (Vicedo, 2014).

2.3.2 Hélices

La correcta elección de las hélices, es muy importante ya que de la sustentación producida por la superficie giratoria de éstas dependerá el funcionamiento óptimo de los motores y el empuje que estos puedan generar.

En cuanto al número de palas en una hélice, una hélice con pocas palas (2, por ejemplo) se caracteriza por una alta eficiencia, sin embargo requiere una mayor cantidad de RPM para mover el mismo volumen de aire y es más ruidosa. Contrariamente a esto, una hélice con un mayor número de palas (3, 4, 5) implica una disminución de las RPM, del ruido y de la eficiencia, así como un incremento en el volumen de aire movido (Olea Hernández & Sánchez López, 2015). En la Figura 2.8 se muestra un tipo de hélices.

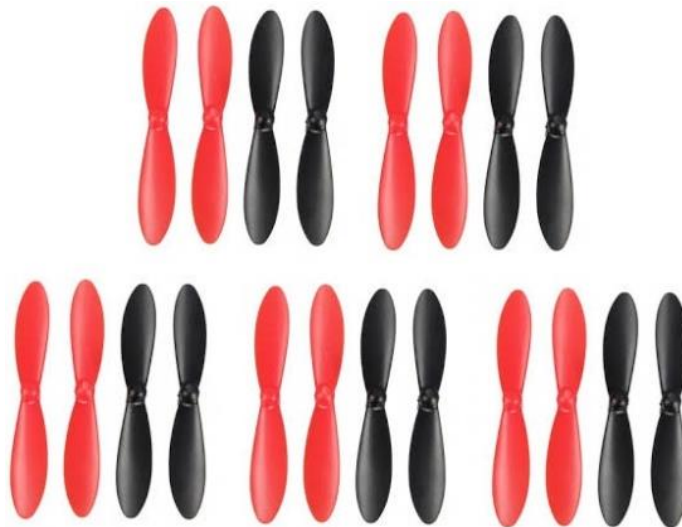


Figura 2.8. Ejemplo de hélices (Faría Hernández, (n.d.))

2.4 SISTEMA ELECTRÓNICO DE UN CUADRICÓPTERO

2.4.1 Motores De Corriente Directa Sin Escobillas (Brushless)

Como se menciona en el artículo de (Enríquez Harper, 2012) los motores de C.D. sin escobillas se emplean en aplicaciones en las que se deben suprimir los riesgos producidos por los arcos y chispas en las escobillas, estos arcos eléctricos son producto del conjunto mecánico conmutador-escobillas y tienen como consecuencia una limitación en la velocidad de operación del motor y el voltaje.

Como menciona (Guru & Hiziroglu, 2003) el motor de C.C. sin escobillas se compone de un devanado multifase arrollado sobre un estator no saliente y un rotor de imán permanente magnetizado radialmente. En la Figura 2.9 se muestra el diagrama esquemático de un motor de C.C. sin escobillas. La conmutación necesaria para producir la rotación se logra aplicando voltaje directo o alterno a los devanados de fase individuales, a través de una operación de conmutación secuenciada.

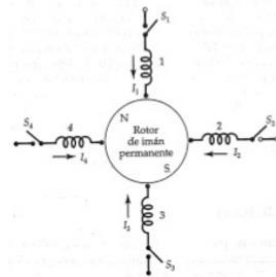


Figura 2.9. Esquema de un motor de cc sin escobillas que ilustra el principio de operación

De acuerdo a (Kosow, 1992) existen tres diferentes categorías, en las que se agrupan los tipos de motores de C.C. sin escobillas:

- Motores de C.C. de conmutación electrónica.
- Motores de C.A. alimentados con C.C. a través de un inversor C.C. /C.A.
- Motores de C.C. de rotación limitada.

- Ventajas de los motores Brushless:

Los motores de C.C. sin escobillas poseen ventajas respecto a los motores de C.C. con colector y escobillas, a pesar de ser más caros a igualdad de potencia:

1. Requieren muy poco mantenimiento.
2. Tienen mayor vida útil.
3. No presentan chispas, lo que los hace menos peligrosos.
4. A causa de su funcionamiento no se forman gases ni partículas de colector o escobillas.
5. Pueden funcionar en distintos ambientes: sumergidos en líquidos, con gases combustibles y sellados herméticamente.
6. Presentan mayor rendimiento que los motores convencionales de C.C.
7. Brindan una respuesta más rápida y características constantes par de salida-corriente de entrada.

- Desventajas de los motores Brushless:

1. Dimensiones totales mayores debido al volumen adicional del equipo electrónico necesario para su control.
2. Mayor coste inicial.
3. Poca variedad de tamaños.

En el artículo de (Olea Hernández & Sánchez López, 2015) menciona que los motores brushless (Figura 2.10) presentan una mayor eficiencia y relación potencia- peso, respecto a los motores DC. Estos motores trabajan con corriente alterna, por medio de variadores de velocidad, normalmente se alimentan de una señal trifásica

sinusoidal, sin embargo en la practica la alimentación se da mediante pulsos. Este tipo de motores son habitualmente empleados en cuadricópteros ya que proporcionan mayor potencia y rendimiento de batería.



Figura 2.10. Motor Brushless

Dentro de la gama de los motores brushless existen dos diferentes clases:

Los motores Outrunner ofrecen un par mayor, sin embargo tienen menor velocidad. Por otro lado, los motores Inrunner son más eficientes a altas velocidades.

Según (Cotte Corredor & Moreno Pineda, 2010) para el control de este tipo de motores se requiere conocer la posición del rotor en cada instante, para ellos se utilizan dos técnicas, obedeciendo a la composición del motor se separan en dos tipos: sensed (con sensores) y sensorless (sin sensores).

Los motores sensed, cuentan con sensores de efecto hall que indican la ubicación del rotor. Comúnmente cuentan con un sensor para cada bobinado del motor, separados 120° entre sí.

En los motores sensorless, la posición del rotor se define por medio del sondeo del efecto producido en las bobinas por la fuerza electromotriz, debido a que no cuenta con sensores.

2.4.2 Variador De Velocidad Electrónico (Esc)

De acuerdo a (Olea Hernández & Sánchez López, 2015) un variador de velocidad electrónico, por sus siglas en inglés Electronic Speed Controller (ESC) es un circuito electrónico (Figura 2.11) que tiene como finalidad variar la velocidad de un motor eléctrico, sin importar las perturbaciones o carga a las que sea sometido, teniendo la capacidad de actuar también como freno dinámico.

El ESC constituye la etapa de potencia ya que controla los motores, mediante la interpretación de los datos provenientes de la emisora radiocontrol o desde una unidad de control y en base a esto suministra una señal pulsante (Pulse Width Modulation) PWM que ocasiona una variación de RPM (revoluciones por minuto) en los motores, resultando así en un movimiento del cuadricóptero.

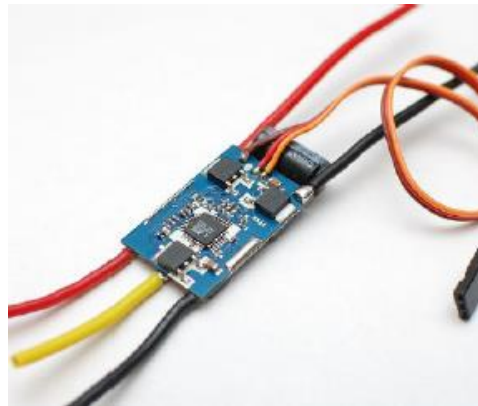


Figura 2.11. Ejemplo de un ESC

Como menciona (Castañeda García, Henao , & Valencia , 2016) un ESC tiene una entrada de alimentación proveniente de la batería y una salida alterna trifásica que controla al motor alimentando sus bobinados en cierta secuencia. Para un cuadricóptero se recomienda usar un ESC que soporte una frecuencia de actualización más rápida (refresh rate), esto otorgará a los motores la facultad de ajustar rápidamente su respuesta. Generalmente, se debe sobredimensionar el ESC, es decir, que se debe elegir un ESC que soporte de 10 a 25% más que el consumo máximo de corriente demandado por el motor.

2.4.3 Batería

Debido a que el cuadricóptero es un sistema portátil la alimentación de los elementos que lo componen es un factor importante a tener en cuenta. De acuerdo a (Álvarez Salgado & Rodríguez Lira , 2015) una batería o acumulador eléctrico recargable, es un dispositivo con la facultad de almacenar y posteriormente entregar la energía eléctrica requerida.

Las baterías son dispositivos recargables (a diferencia de las pilas), están constituidas por varios acumuladores dispuestos en serie para incrementar su capacidad de entrega de tensión de salida.

Las características deseables en baterías de cualquier sistema autónomo son: que proporcionen una elevada potencia y energía, ser de bajo precio y mantenimiento, seguras ante condiciones externas a las que pueda estar sometido el cuadricóptero.

Los parámetros de una batería a tomar en cuenta son:

- La tensión de salida, medida en volts (V).
- La capacidad eléctrica, medida en referencia a los tiempos de carga y descarga en amperio-hora o miliamperio-hora (Ah o mAh, también puede medirse en Coulombs, C). Este parámetro define que tan rápido se descarga la batería.
- Densidad de energía. Es la propiedad que indica cuanta energía es almacenada por Kg de batería, se mide en $A \cdot h \cdot V / Kg$ (Morales Delgado, 2015).
- Tasa de descarga. Indica la velocidad a la que se descarga la batería, es decir, la intensidad máxima que la batería puede proporcionar, se mide en A.

Para este proyecto en particular, se decidió emplear el tipo de batería Li-Po (Polimero de Litio) y su respectivo cargador, debido a que tienen una densidad de energía más elevada que otros modelos tal y como se muestra en la tabla comparativa 1. Este tipo de baterías no poseen efecto memoria; sin embargo son más costosas.

Tipo	Energía / Peso	Tensión por elemento(V)	Duración (número de recargas)	Tiempo de carga	Auto descarga por mes(% del total)
Plomo	30-40 WH/KG	2V	1000	8-16H	5%
Ni-Fe	30-55 Wh/kg	1.2V	>= de 10000	4-8H	10%
Ni-Cd	48-80 Wh/kg	1.25V	500	10-14H	30%
Ni-Mh	60-120 Wh/kg	1,25V	1000	2h-4h	20%
Li-ion	110-160 Wh/kg	3.16V	4000	2h-4h	25%
Li-Po	100-130 Wh/kg	3.7v	5000	1h-1.5h	10%

Tabla 1. Tabla comparativa entre diversos modelos de baterías.

El consumo máximo del motor debe ser menor que la descarga máxima de la batería, para evitar que se reduzca la vida útil de ésta. En la Figura 2.12 se muestra una batería de Li-Po



Figura 2.12. Ejemplo de una batería Li-Po

2.4.4 Sensores Dinámicos

Si se desea realizar un dispositivo que pueda interactuar con el medio ambiente y pueda corregir las perturbaciones por medio de un algoritmo de control se necesitaran sensores más específicos aquellos sensores con características dinámicas tales como el acelerómetro y giroscopio.

Usualmente estos dos sensores son utilizados juntos, es decir en un solo circuito integrado esto es porque mientras el acelerómetro se encarga de medir las aceleraciones en los tres ejes y el giroscopio detecta las derivadas de los giros ya que mide la velocidad de rotación del mismo sobre los tres ejes.

2.4.5 Giroscopio Y Acelerómetro IMU (Inertial Measurement Unit)

La definición en español es unidad de medición inercial como su nombre lo dice es un dispositivo que se encarga de medir la velocidad, la orientación y la fuerza gravitacional utilizando el acelerómetro y giroscopio como sus principales sensores para detectar los movimientos rotacionales tanto como yaw, pitch y roll mediante el giroscopio. En la figura 2.13 se observa el dispositivo MPU-6050. (Arduino, n.d)



Figura 2.13. MPU-6050 Arduino.

Este dispositivo es conveniente debido a su bajo costo en el mercado y por su capacidad de rapidez de recibir y enviar datos a la tarjeta Arduino gracias a que cuenta con el protocolo de comunicación I²C, también se decidió utilizarlo ya que existen una gran cantidad de informes que pueden facilitar información sobre ayuda su funcionamiento. Además de que consta de 3 acelerómetros y 3 giroscopios con la opción de agregarle un magnetómetro.

- Acelerómetro:

Un acelerómetro es un dispositivo el cual es capaz de medir la aceleración de coordenadas, es decir mide el cambio de velocidad en el espacio. Cabe destacar que miden la fuerza de gravedad sobre sí mismo incidiendo toda la fuerza sobre cualquiera de los tres ejes ortogonales. El valor que se envía del acelerómetro al Arduino varía entre un rango de -16384 a 16384.

Para saber la teoría del funcionamiento se recurrirá al artículo escrito por (José, 2015) el cual menciona que el acelerómetro contiene un material piezo eléctrico casi sólido, en su mayoría de cuarzo cargado eléctricamente el cual tiene una masa conocida y en consecuencia se comprime. Esta presión produce una alteración en la alineación de los electrones y protones en el material percibiéndose así una variación a la carga eléctrica que resulta proporcional al esfuerzo experimentado por el material, el cual es proporcional a la aceleración de la gravedad por la segunda ley de Newton $F=ma$. En la figura 2.14 se representa el esquema de construcción de un acelerómetro.

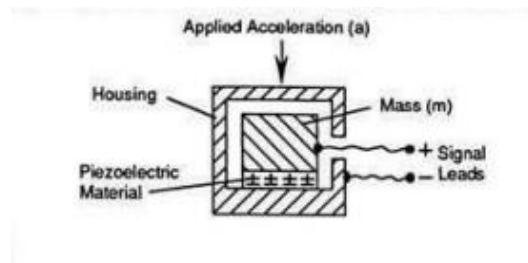


Figura 2.14. Esquema de construcción de un acelerómetro.

En el artículo de (Martín-Gil, 2012) se explica el funcionamiento de un acelerómetro del siguiente modo, se puede imaginar una pelota de tenis encerrada en una especie de dado gigante. Este dado será el acelerómetro, donde las paredes del mismo serán capaces de medir la fuerza que aplicamos sobre ellas. Si se mueve el dado hacia la izquierda, la pelota chocará contra la pared izquierda. La fuerza que se mida en esta pared será la manera de medir la aceleración.

Si se mueve el dado en diagonal, la pelota hará fuerza en dos paredes en vez de una, pero siguiendo la misma idea. Pero si se deja el dado en el suelo gracias a la gravedad, la pared inferior medirá una fuerza, y sin embargo no hay cambio de velocidad en el dado (acelerómetro). En la figura 2.15 se representa el comportamiento del acelerómetro en un ambiente ilustrativo para una mejor comprensión.

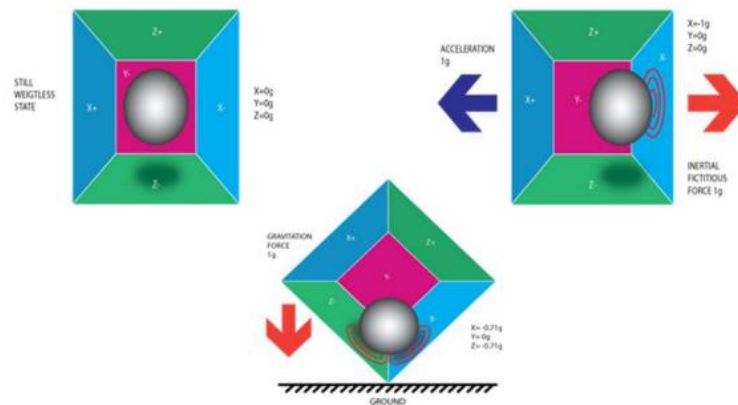


Figura 2.15. Representación gráfica del comportamiento de un acelerómetro.

- Giroscopio:

El giroscopio o también llamado giróscopo surgió en 1852 desarrollado por Foucault aunque se cree que ya existían antecedentes propuestas por el alemán Johann Bohnenberger a quien se le atribuye al descubrimiento del giroscopio.

A pesar de su descubrimiento en 1813 estos aparatos tenían grandes dimensiones y su aplicación era puramente militar. Gracias al desarrollo de la electrónica se construyeron giroscopios MEMS los cuales funcionan con el efecto coriolis para tomar medidas, esto nos lleva a preguntarnos como es que funciona este efecto (José, 2015) menciona que el efecto coriolis está presente en todos los cuerpos de rotación, incluida la Tierra.

Este efecto consiste en la distancia percepción del movimiento al observar desde un sistema de referencia rotatorio (no inercial) a un objeto que se encuentra fuera de este sistema. El siguiente ejemplo explica el funcionamiento, si un objeto se encuentra situado en el origen de un círculo y este pretende ir en línea recta del centro a un punto A sin embargo debido a que el círculo se encuentra girando en sentido anti horario el objeto llegara a un punto B debido a que mientras el objeto se dirige hacia el punto A en línea recta el círculo o la plataforma sobre la que se desplaza sigue girando.

En la figura 2.16 se hace referencia a este efecto llamado coriolis.

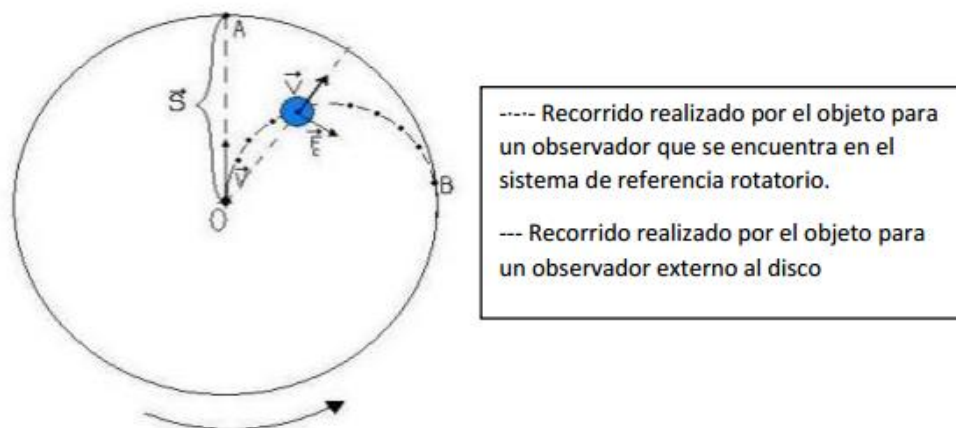


Figura 2.16. Representación del efecto coriolis.

2.4.6 Protocolo De Comunicación

I²C es un protocolo de comunicación y por sus siglas en inglés Inter-integrated circuit también conocido como TWI de Two Wire literalmente dos cables, es un sistema utilizado para comunicar circuitos integrados entre sí.

Su principal característica es poder utilizar dos líneas para transmitir la información:

- SDA: utilizada para transferir los datos como los ceros y unos.
- SLA: sirve para enviar la señal de reloj.

Cada dispositivo conectado al I²C tiene una dirección que lo identifica del resto de dispositivos y se puede configurar en maestro o esclavo. Un dispositivo maestro es aquel que inicia la transmisión de datos y genera la señal de reloj (Artero, 2014).

2.4.7 Filtro Complementario

Debido a las variaciones de las mediciones que pueden llegar a tener tanto el giroscopio como el acelerómetro es importante tener un filtro que sea capaz de limpiar la señal de ruidos y perturbaciones generadas por las vibraciones del cuadricóptero. Es por esto que se hará uso del filtro Kalman el cual funciona recogiendo variables en el tiempo y convirtiéndola en una sola sin ruido para describir cómo es que trabaja el filtro Kalman se hará referencia al artículo realizado por (Martín-Gil, 2012) El proceso consta de dos etapas. En una primera fase de predicción, se hace una estimación del estado actual de la variable y de su nivel de incertidumbre. Una vez llega la siguiente medición (que estará corrupta, debido a un cierto error y/o ruido), se hace una estimación del valor correspondiente utilizando una media ponderada, dando mayor peso, a los valores cercanos. Al tratarse de un algoritmo recursivo, se puede ejecutar en tiempo real, siendo necesario solamente, la medición actual y el estado anterior. No hace falta más información que esa.

2.5 SISTEMA DE CONTROL DE UN CUADRICÓPTERO

2.5.1 Arduino

Como controladora de vuelo se empleara una tarjeta electrónica de la marca Arduino. Arduino es una tarjeta de hardware libre que incorpora un microcontrolador reprogramable y una serie de pines hembra (los cuales están unidos internamente a las patillas de E/S del microcontrolador) que permiten conectar allí de forma muy sencilla y cómoda diferentes sensores y actuadores (Artero, 2014)

Arduino nació en el Instituto de Diseño de Interacción Ivrea como una herramienta fácil para el prototipo rápido, dirigido a estudiantes sin experiencia en electrónica y programación. Todas las tarjetas Arduino son completamente de código abierto, permitiendo a los usuarios crear de forma independiente y, finalmente, adaptarlos a sus necesidades particulares. El software también es de código abierto, y está creciendo a través de las contribuciones de los usuarios en todo el mundo. (what is arduino?, (n.d.)).

En la figura 2.17 se pueden observar las diferentes tarjetas con las que cuenta Arduino.

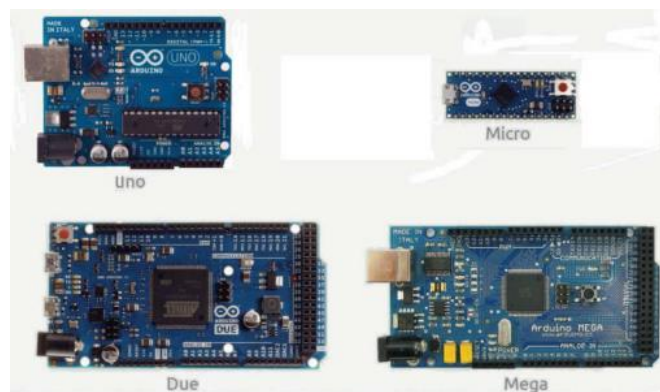


Figura 2.17. Tarjetas de Arduino. (Vicedo, 2014)

Es importante contar con una tarjeta Arduino que pueda trabajar como maestro o esclavo, es decir, que pueda controlar los datos y las señales de reloj desde la programación, es por eso que se eligió la tarjeta Mega de Arduino.

- Arduino Mega 2560:

El Arduino mega 2560 es una tarjeta basada en el microcontrolador ATmega2560, uno de sus principales características es la que cuenta con 54 pines de entrada/salida digitales los cuales 14 son utilizados como salidas analógicas PWM, 16 entradas analógicas y 4 receptores/transmisores serie TTL-UART. Consta de una memoria flash de 256 kilobytes, una memoria SRAM de 8 Kb y una EEPROM de 4 Kb. Su voltaje de trabajo es de 5v. (Artero, 2014).

- Arduino Mega ADK:

Es una tarjeta similar a la antes mencionada Mega 2560 con la única diferencia de que esta tarjeta ADK puede trabajar como un dispositivo “host USB”, es decir el host es el único que puede iniciar y controlar la transferencia de datos entre el resto de dispositivos conectados.

2.5.2 Algoritmos De Control

Como menciona en su artículo (Álvarez Salgado & Rodriguez Lira , 2015) un algoritmo es una secuencia de reglas capaz de resolver una variedad de problemas. Un algoritmo debe ser finito, eficiente y rápido sus reglas deben tener un orden específico y definir la acción a desarrollar.

(Cotte Corredor & Moreno Pineda, 2010) El PID es un algoritmo de control en lazo cerrado de alta precisión y estabilidad, que maniobra las señales de entrada a la planta en base al comportamiento de la planta a través del tiempo y de su tasa de cambio.

El controlador proporcional integral derivativo (PID) compara la variable de salida “y”, medida mediante un sensor, con un valor de referencia “y_o”. El error “e”, que es la diferencia entre el valor de referencia “y_o” y el valor medido de salida “y”, es procesado para calcular la nueva señal de entrada a la planta; dicha señal es llamada variable de control “u”, esta señal tiene la función de ajustar la salida para mantenerla en el valor de referencia a pesar de las perturbaciones que puedan existir. La Figura 2.18, muestra el esquema de un Sistema en lazo cerrado con controlador PID, en él se muestra el comparador, el controlador PID, la planta o sistema y la señal de retroalimentación que se mide mediante un sensor.

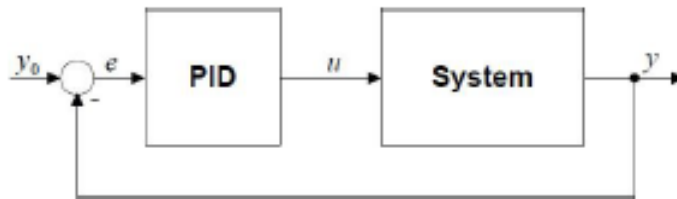


Figura 2.18. Sistema en lazo cerrado con controlador PID

El controlador PID se compone de tres acciones de control:

- Acción Proporcional:

Al usar este término por si solo se tiene error de estado estacionario en todos los casos, a excepción de sistemas con entrada igual a cero. El término proporcional actúa en el presente y da al sistema una señal de control proporcional al error, de acuerdo a la ecuación (1):

$$u(t) = K_p e(t) = K_p (y_o(t) - y(t)) \quad \text{Ec. (1)}$$

Donde K_p, es la ganancia proporcional.

- Acción Integral:

La acción integral actúa “recapitulando” el pasado, da al sistema una señal de control que corresponde a la suma del error previo. La suma del error continúa hasta que la variable de salida iguale al valor deseado, eliminando como resultado el error de estado estacionario provocado por la acción proporcional. Usando únicamente el término integral el sistema responde muy lentamente y usualmente con oscilación. Esta acción se expresa mediante la ecuación (2):

$$u(t) = K_i \int_0^t e(t) dt \quad \text{Ec. (2)}$$

Donde K_i es la ganancia integral, K_i puede expresarse en términos de la ganancia integral, K_p , donde T_i expresa el tiempo de integración, como se expresa en la ecuación (3):

$$K_i = \frac{K_p}{T_i} \quad \text{Ec. (3)}$$

- Acción Derivativa:

El término derivativo actúa anticipando el futuro, da al sistema una señal de control dependiente de la tasa de cambio del error. Este término mejora la respuesta a cambios repentinos del error realizando correcciones proporcionalmente a la misma velocidad con la que éste se produce, lo cual se refleja en respuestas más rápidas que con control P o PI. Este tipo de acción se debe emplear junto a otra acción de control, ya que por sí solo no responde al error de estado estable. Un valor muy alto del término derivativo generalmente desestabiliza el sistema y también incrementa su sensibilidad al ruido, por lo cual los controladores PI son comunes. La acción derivativa se describe en la ecuación (4):

$$u(t) = K_d \frac{de(t)}{dt} \quad \text{Ec. (4)}$$

Donde K_d es la ganancia de la acción derivativa, ésta puede expresarse en términos de K_p , donde T_d representa el tiempo de derivación, como se expresa en la ecuación (5):

$$K_d = K_p T_d \quad \text{Ec. (5)}$$

- Control proporcional-integral-derivativo, PID:

Generalmente se consigue un mejor desempeño del controlador empleando los términos proporcional, integral y derivativo juntos (PID). La figura 2.19 muestra la comparación entre la respuesta de controladores P, PI y PID. El PI supera al P eliminando el error de estado estacionario, y el PID mejora la respuesta del PI con una respuesta más rápida y sin sobre impulso.

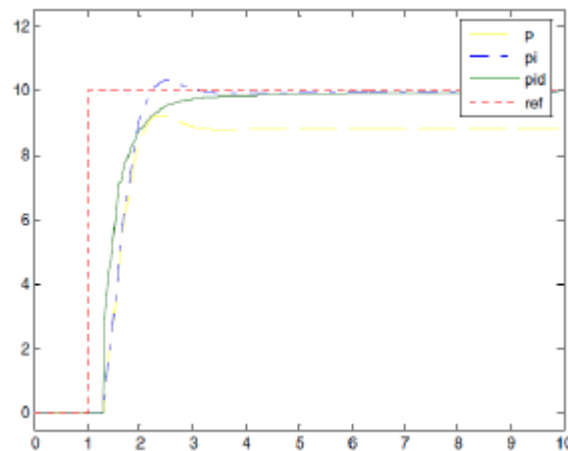


Figura 2.19. Respuesta de controladores P, PI y PID

Según (Hernández Gaviño, 2010) un controlador es proporcional-integral-derivativo cuando la salida del controlador $u(t)$ es proporcional al error $e(t)$, más una cantidad proporcional a la integral del error, sumado a una cantidad proporcional a la derivada del error, esto queda plasmado en la ecuación (6):

$$u(t) = K_p e(t) + K_i \int_0^t e(t) dt + K_d \frac{de(t)}{dt} \quad \text{Ec. (6)}$$

En la Figura 2.20 se muestra el esquema del controlador PID, aplicado a un sistema de control en lazo cerrado (Álvarez Salgado & Rodríguez Lira , 2015):

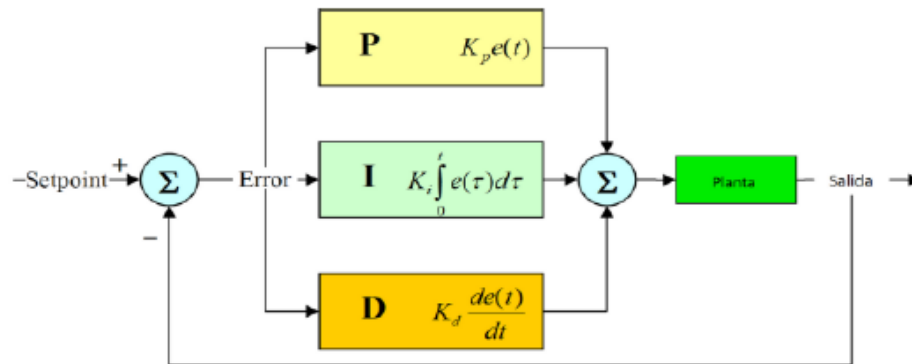


Figura 2.20. Esquema del controlador PID, aplicado a un sistema de control en lazo cerrado.

2.6 SISTEMA DE COMUNICACIÓN MEDIANTE RADIOFRECUENCIA

El sistema de comunicación consta del emisor, que es el empleado por el conductor, y el receptor que va conectado al controlador de vuelo. Habitualmente la frecuencia establecida para esta clase de control es la banda de 2.4GHz, de esta forma se evita que ocurran interferencias con otras señales de radio como se describe en el artículo de (Tabuchi Fukuhara, 2015) Para desarrollar el control de un cuadricóptero, se requiere un receptor de mínimo cuatro canales, en la Figura 2.21 y 2.22 se muestran un transmisor y un receptor de radiofrecuencia respectivamente.

Como especifica (Olea Hernández & Sánchez López, 2015) en su trabajo la mayor restricción y problemática de este sistema de comunicación inalámbrica, es el empleo de la misma banda de frecuencia por más de un usuario lo que puede ocasionar, influencia de una emisora cercana.



Figura 2.21. Transmisor de radio



Figura 2.22. Receptor de radio

CAPÍTULO III. CARACTERIZACIÓN DEL ÁREA DE TRABAJO

3.1 GENERALIDADES DE LA EMPRESA

El Instituto Tecnológico de Ciudad Guzmán se encuentra ubicado en Ciudad Guzmán, Municipio Zapotlán el Grande, Jalisco, México. A 3.5 km. del Centro de la Ciudad, en Avenida Tecnológico #100 por la carretera Cd. Guzmán el Fresno, correspondiente a la región centro-sur del Estado de Jalisco.

Razón social: Instituto Tecnológico de Ciudad Guzmán.

Teléfono: 01 (341) 575 20 50.

Fax: 01 (341) 575 20 50.

Correo electrónico: elecenica@itcg.edu.mx

Página web: <http://www.itcg.edu.mx/>

3.2 ANTECEDENTES DE LA EMPRESA

En Mayo de 1972 se hizo la petición al Sr. Presidente de la República, durante una de sus visitas a Cd. Guzmán, de la creación de una Institución Educativa de nivel superior en el Sur de Jalisco, que tuviera como finalidad propiciar el desarrollo cultural, técnico y económico de la región, así como reducir los flujos migratorios de los jóvenes estudiantes hacia las grandes ciudades en busca de su formación profesional. Así mismo, otro de los objetivos consistió en fomentar el arraigo de los egresados en sus lugares de origen.

El Instituto Tecnológico nace el 13 de Septiembre de 1972, sobre una extensión de terreno de 26 hectáreas con el nombre de Instituto Tecnológico Regional No.29 de Ciudad Guzmán, Jalisco.

El inicio de sus actividades se hace el 20 del mismo mes y año en las instalaciones del CERETI (hoy CETIS) en la ciudad de Guadalajara, contando con una población de 120 alumnos de nivel licenciatura, 15 docentes y 10 administrativos.

El 22 de Marzo de 1973, el CAPECE terminó la construcción de aulas y laboratorios, empezando en ese momento el nivel bachillerato ya en las instalaciones propias y en el mes de Junio del mismo año se integraron los estudiantes que se encontraban en el CERETI de Guadalajara, a partir de ese periodo el ITCG ha logrado un desarrollo en todo sentido apegado a los lineamientos emanados de la SEP. Así mismo el Instituto Tecnológico de Ciudad Guzmán forma parte del Tecnológico Nacional de México.

3.3 MISIÓN

Ofrecer servicios de Educación Superior Tecnológica, formando profesionales íntegros, con capacidad técnica y científica para dar solución a la problemática que presenta la modernización y contribuir responsablemente en el desarrollo sustentable, fortaleciendo los valores y actitudes de nuestra comunidad para que coadyuve a la conformación de una sociedad más justa y humana.

3.4 VISIÓN

Ser una Institución de Educación Superior Tecnológica de alto desempeño, formando profesionales acreditados a nivel internacional, caminando con rumbo hacia la mejora continua, con programas estratégicos de investigación y posgrado, acordes a las necesidades sociales, ofreciendo servicios de calidad, producto del desarrollo, participación y superación permanente de nuestro capital humano, guardando siempre una relación armoniosa, responsable y honesta entre el hombre y la naturaleza.

3.5 OBJETIVOS

- Académico: Gestionar los planes y programas de estudio para la formación profesional del estudiante.
- Planeación: Realizar la planeación, programación, presupuestación, seguimiento y evaluación de las acciones para cumplir con los requisitos del servicio.
- Vinculación: Contribuir a la formación integral de estudiantes a través de su vinculación con el sector productivo, la sociedad, la cultura y el deporte.
- Administración de Recursos: Determinar y proporcionar los recursos necesarios para lograr la conformidad con los requisitos del servicio educativo.
- Calidad: Gestionar la calidad para lograr la satisfacción del Estudiante.

3.6 VALORES

- El Ser Humano.
- El Espíritu de Servicio.
- El Liderazgo.
- El Trabajo en Equipo.
- La Calidad.
- El Alto Desempeño

3.7 POLÍTICA DE CALIDAD

El Instituto Tecnológico de Ciudad Guzmán establece el compromiso de implementar todos sus procesos, orientándolos hacia la satisfacción de sus clientes sustentada en la Calidad del Proceso Educativo, para cumplir con sus requisitos, mediante la eficacia de un Sistema de Gestión de la Calidad y de mejora continua, conforme a la norma ISO 9001:2008/NMX-CC-9001-IMNC-2008.

3.8 DESCRIPCIÓN DEL ÁREA DE NEGOCIOS

El área al cual se enfoca este proyecto corresponde al departamento de Ingeniería eléctrica y electrónica del Instituto Tecnológico de Cd. Guzmán el cual se encarga de formar profesionistas con competencias profesionales para diseñar, modelar, implementar, operar, integrar, mantener, instalar, administrar, innovar y transferir tecnología electrónica existente y emergente en proyectos interdisciplinarios, a nivel nacional e internacional, para resolver problemas y atender las necesidades de su entorno con ética, actitud emprendedora, creativa, analítica y comprometidos con el desarrollo sustentable

3.9 PUESTO ASIGNADO Y FUNCIONES

Residente en el departamento de Ingeniería eléctrica y electrónica, el proyecto necesita de conocimientos previos como saber programar en varias plataformas, entendimiento de circuitos eléctricos y conceptos básicos en teoría de control específicamente en el control PID. Se realizara un proyecto específico al área de electrónica para la implementación del control de cuatro motores en un cuadricóptero.

CAPÍTULO IV. DESCRIPCIÓN DE LAS ACTIVIDADES DESARROLLADAS

4.1 ACTIVIDADES DEL PROYECTO

4.1.1 Diseño E Implementación Del Frame

El frame es parte importante del desarrollo de la plataforma debido a que se encarga de llevar sobre él los componentes del cuadricóptero. Como se explicó anteriormente los cuadricópteros pueden tener diferentes formas, para este proyecto se utilizó la forma de X gracias a su maniobrabilidad y a su tendencia a la estabilidad.

La construcción del frame se realizó mediante una impresora 3D TRONXY la cual cuenta con dos tipos de materiales de impresión PLA y ABS, por lo general la más utilizada en la industria es el material ABS pero es un poco más pesado por lo que se decidió utilizar el material PLA que además es biodegradable y excelente para las piezas ligeras y huecas.

Las piezas fueron diseñadas en el software SolidWorks por su capacidad de hacer piezas con un alto grado de exactitud en las medidas, estas deben ser extremadamente exactas ya que de no ser así puede existir problemas con el giro de las hélices. El diseño de las piezas se realizó en forma organizada para llevar un control de las medidas y no tener algún error que pudiera poner en riesgo la estabilización del cuadricóptero. A continuación se presentan las partes que se diseñaron y el orden en el cual fueron diseñadas.

- Base:

La base fue la pieza principal ya que a partir de ésta se establecieron las medidas de cada brazo, tomando en cuenta para ello que el espacio entre brazos fuera lo suficientemente grande para permitir el giro de las hélices. La forma de la base es un octágono debido a que facilita colocar los brazos en forma de cruz y crea una superficie adecuada para colocar dispositivos dentro de ésta. Como se puede ver en la figura 4.1 (medidas están en milímetros) se determinó el grosor de la pieza de 5

mm, la altura de 20 mm, el largo entre los extremos de la pieza de 140 mm y el diámetro de los orificios de 5 mm por lo que es importante aclarar que la medida de 132.50 mm que se observa en la figura de la pieza se debe a que se realizaron unas viñetas para colocar una cubierta.

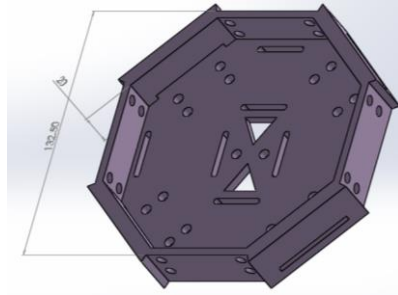


Figura 4.1. Base del cuadricóptero.

- Brazo:

Esta pieza fue diseñada con medidas anteriormente calculadas tomando en cuenta que las hélices miden 10 pulgadas, por lo tanto, los brazos deben tener un largo mínimo de 150 mm para que las hélices no impacten entre sí. Se incluyó una base para el motor al extremo del brazo, el diámetro y la altura de ésta base fueron considerados previamente considerando las medidas de los motores utilizados. En la figura 4.2 se aprecia la forma de los brazos y sus respectivas medidas, largo de aproximadamente 200 mm, base del motor 32 mm y 34 mm para la altura de la base del mismo, el extremo del brazo se diseñó más delgado para ensamblarlo a la base.

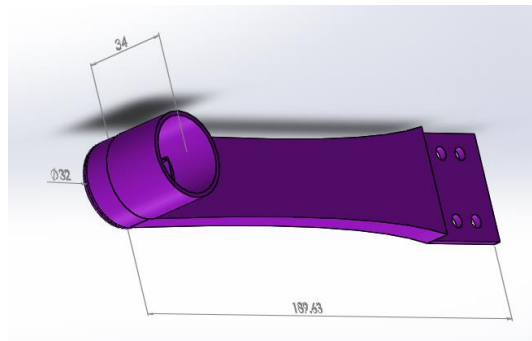


Figura 4.2. Brazo del cuadricóptero.

- Cubierta:

Esta pieza se diseñó para proteger a los dispositivos colocados dentro de la base y que de esta manera no sufran daños debido a las condiciones del medio ambiente. Las medidas se asignaron para que coincidieran con las medidas de la base, como se aprecia en la figura 4.3 la cubierta tiene el mismo largo que la base (140 mm), altura de 10 mm, la altura de la viñeta es de 35 mm y el grosor de 5 mm.

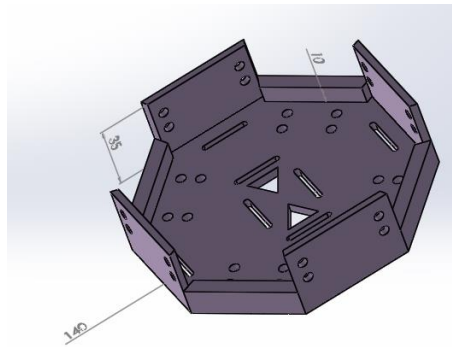


Figura 4.3. Cubierta del cuadricóptero.

Una vez terminadas las piezas se realizó el ensamblaje de las mismas, para verificar que no existieran fallas en las mediciones y corroborar que las hélices tengan espacio suficiente para girar libremente sin ningún riesgo.

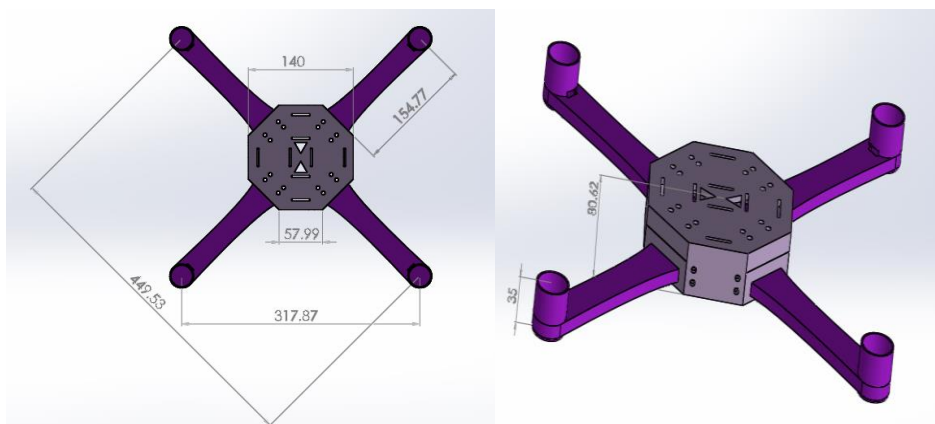


Figura 4.4. Ensamblaje del cuadricóptero.

En la figura 4.4 se observan las medidas finales del ensamblaje de todas las piezas. Las medidas más importantes son las que hay entre los motores tanto en diagonal (aproximadamente 450 mm) como de forma vertical (aproximadamente 320 mm); además de un grosor total de aproximadamente 80 mm.

4.1.2 Elección De Los Componentes Del Cuadricóptero

- Elección de los motores brushless:

La elección del motor brushless adecuado para un cuadricóptero, se basa en algunos parámetros a considerar y es de gran importancia ya que influye en el rendimiento de éste.

Para la elección del motor se consideró la aplicación que se le dará a la aeronave, debido a que se busca la estabilidad del cuadricóptero y no que éste alcance grandes velocidades, se optó por un motor brushless Outrunner A2212 de 1000 kV (Figura 4.5), que se considera un bajo kV ya que esto brindará al cuadricóptero un par elevado, sin embargo no alcanzará altas velocidades. El kV es el valor que indica el número de revoluciones por minuto a las que es capaz de girar el motor por cada Voltio que se le aplica.

En la tabla 2 se muestran las características y especificaciones del motor.

A2212 Motor Brushless	
kV	1000 Max
Eficiencia	80%
Eficiencia máxima de corriente	4-10A (>75%)
Capacidad de corriente	12A/ 60s
Corriente sin carga @ 10V	0.5A
No. de celdas	2-3 Li-Po
Dimensiones del Motor	φ 27.5* 30 mm
Diámetro del eje	φ 3.17mm
Peso	47 g

Tabla 2. Especificaciones del motor brushless A2212

Otro punto importante tomado en cuenta para elegir los motores fue el peso, ya que se recomienda que éste no sea muy elevado debido a que es un factor que podría afectar la eficiencia del cuadricóptero, por ende entre más peso se agregue al cuadricóptero se reducirá el tiempo de vuelo que la batería podrá brindar.

Existen otros parámetros importantes a tomar en cuenta para elegir el motor adecuado, como el empuje, el tamaño del motor y el voltaje con el que este funciona, sin embargo, debido a la naturaleza del proyecto y a los requerimientos que se ajustan a éste los factores decisivos para la elección del motor como se dijo anteriormente fueron el valor kV y el peso del motor que es de 47g.

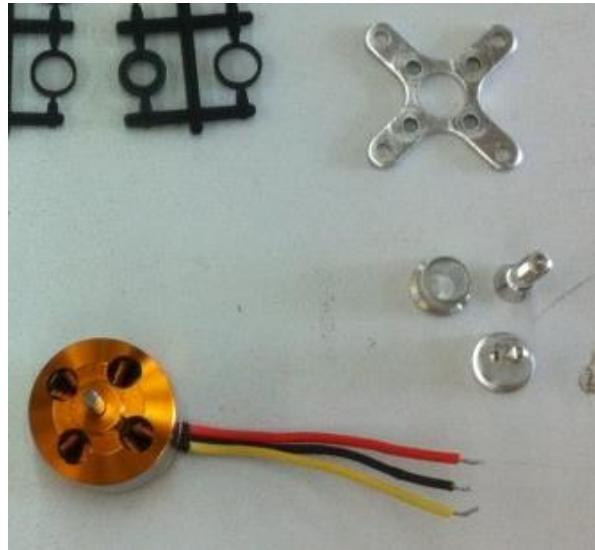


Figura 4.5. Motor y accesorios

- Controlador electrónico de velocidad (ESC):

Para la elección del ESC se debe tomar en cuenta el valor máximo de corriente de consumo del motor, buscando de esta manera que el variador pueda proporcionarle al motor la corriente necesaria. Este valor máximo de corriente especificado en la hoja de datos del motor debe sobredimensionarse; para ello se elige el variador incrementando aproximadamente un 20% más el valor de la corriente, como margen de seguridad. El motor elegido para este proyecto tiene un valor de corriente a eficiencia máxima de 4-10 Amperes por lo que el ESC debe tener la capacidad de proporcionar como mínimo 12 Amperes.

Sin embargo, en la búsqueda de los componentes del cuadricóptero se eligió un paquete que incluye un motor brushless de 1000kV (y sus accesorios), un par de hélices 1045 y un ESC con capacidad de corriente máxima de 30 Amperes, como el que se muestra en la figura 4.6 Con ello queda sobredimensionado el valor máximo de corriente que el motor demanda. En la tabla 3 se muestran las especificaciones del ESC.

ESC 30A	
Salida	Continua 30A
Entrada de Voltaje	Batería de litio de 2-4 celdas o batería de NiCd/NIMh de 5-12 celdas
Circuito eliminador de batería (BEC)	2A / 5V (Modo lineal)

Tabla 3. Especificaciones del ESC 30 A

A su vez es importante comprobar que el ESC elegido está clasificado para el número correcto de celdas de la batería. En este caso el ESC incluido en el paquete está clasificado para baterías de Litio de 2 a 4 celdas, de manera análoga al motor que está clasificado para baterías Li-Po (Polímero de Litio) de 2 a 3 celdas.



Figura 4.6. ESC

- Hélices:

Las hélices, como se dijo anteriormente, fueron incluidas en el paquete adquirido, por ello se puede tener la seguridad que son las adecuadas para el motor elegido, ya que de no ser así esto podría provocar daños a éste último. El paquete previamente mencionado incluía dos hélices de 10 pulgadas (Figura 4.7), una en sentido horario y la otra en sentido anti horario, además incluye diversos adaptadores de orificio para poder montarlas en el eje del motor. Es importante que dos de las hélices del motor giren en sentido horario y las dos restantes giren en sentido anti horario ya que de lo contrario el cuadricóptero no se estabilizará.



Figura 4.7. Hélices con sus respectivos adaptadores de orificio

En la tabla 4 se muestran las especificaciones de las hélices 1045.

Hélices 1045	
Diámetro del eje	6.0mm
Grosor del centro	9.7mm
Peso	15 g / par
Agujero ajustable por adaptador	(3mm, 3.2mm, 4mm, 5mm, 6mm, 6.35mm, 7.95mm)

Tabla 4. Especificaciones de las hélices 1045

- Batería y cargador de batería:

El criterio para elegir la batería adecuada para el cuadricóptero se basó en encontrar una relación entre capacidad y peso adecuado, ya que si bien es cierto que las baterías de mayor capacidad brindan un mayor tiempo de vuelo, también son más pesadas y pueden hacer ineficiente al cuadricóptero.

Por ello se eligió una batería que pudiera satisfacer las necesidades de corriente requerida por los ESC sin representar un problema de eficiencia, este valor se calcula tomando en cuenta que el cuadricóptero está compuesto por cuatro ESC's que pueden llegar a consumir 30A cada uno, esto daría como resultado un valor de 120A. En la tabla 5 se muestran las especificaciones de la batería elegida.

Batería Li-Po	
Capacidad mínima	2200 mAh
Configuración	3S1P/11.1 V/ 3 celdas
Constante de descarga	25C
Pico de descarga (10 s)	35C
Peso	188g
Tamaño	105 x 33 x 24mm
Conector de carga	JST-XH
Conector de descarga	XT60

Tabla 5. Especificaciones de la batería

Este valor implicaría una batería de gran tamaño, sin embargo, es importante tomar en cuenta que los ESC's fueron altamente dimensionados (de 12A a 30A) por ello se realizó el cálculo para la batería tomando en cuenta el valor calculado para los ESC's, es decir, 12A.

Por lo tanto el valor que la batería debe proporcionar es de 48A, es preferible dar cierta holgura a este valor para que la batería no trabaje a su máxima capacidad y así tenga una mayor vida útil. Debido a los requerimientos del motor y el ESC, se decidió que la batería a emplear sería una batería Li-Po de 3 celdas (Figura 4.8), con capacidad de 2200mAh (igual a 2.2Ah), que entrega 11.1V de salida y cuya constante de descarga es 25C.

Con estos parámetros es posible calcular que ésta batería es capaz de entregar:

$$2.2Ah (25C) = 55A$$

Este valor, es adecuado ya que es 7A mayor que la corriente requerida para alimentar a los cuatro motores.



Figura 4.8. Batería

Aunado a esto fue requerido un cargador para baterías Li-Po, para esto se tomó en cuenta que fuera el cargador adecuado para el tipo de batería ya que debido a los componentes químicos se requiere un cuidado especial para cada caso. Se tomó en cuenta que fuera adecuada al número de celdas de la batería y al voltaje que esta requiere.

En la tabla 6 se muestran las especificaciones del cargador.

Cargador de batería Li-Po	
Voltaje de entrada	AC 100V-240V
Corriente de salida	3x 700mA
Corriente balanceada de carga	850 mA
Corriente máxima de carga	3x 800mA
Pantalla	3x Bicolor LED
Material del armazón	ABS
Dimensión	90 x 55 x 35mm/3.5 x 2.2 x 1.4 pulg
Conector de poder	Conector EU
Peso	82g/2.87oz

Tabla 6. Especificaciones del cargador de batería

- Sistema de comunicación mediante radiofrecuencia:

Para el cuadricóptero es necesario tener un sistema de comunicación mediante radiofrecuencia que permita manipular el cuadricóptero a una distancia considerable y sobre todo, que cuente con un mínimo de 4 canales para los movimientos roll, pitch, yaw y el arranque de los motores.

En el mercado existen una gran variedad de sistemas de radiofrecuencia con muchos canales, pero para esta aplicación se empleará un receptor con 6 canales por lo que se optó por utilizar un Turnigy-i6 como el que se muestra en la figura 4.9.



Figura 4.9. Transmisor (der.) y Receptor (izq.) RF Turnigy-i6

Este dispositivo es un transmisor de 2.4Ghz con 6 canales de entrada que cuenta con el Sistema Digital Hopping (AFHDS), un interruptor de 3 posiciones para elegir los modos de vuelo que pueden ser utilizados con una tarjeta controladora de vuelo y con un receptor de 6 canales. Además cuenta con una pantalla retro iluminada, su programación es fácil y amigable con el usuario.

A continuación, en la tabla 7, se muestran las especificaciones más importantes del sistema de comunicación de una forma más detallada.

TGY-i6	
Frecuencia	2.4GHz
Modulación	GFSK
Banda	142
No. De canales de frecuencia	20
Potencia de salida	$\leq 20\text{dBm}$
Corriente de trabajo	$\leq 100\text{mA}$
Voltaje de funcionamiento	1.5v x4 AA
Dimensiones	174x89x190mm
Peso	392g
Resolución	1024 bits

Tabla 7. Especificaciones del TGY-i6.

- Sensores dinámicos:

El sensor es uno de los dispositivos más importantes en el cuadricóptero ya que de acuerdo a la lectura del sensor se podrán realizar las correcciones en un control PID. Por ende se utilizó el dispositivo IMU MPU-6050 el cual contiene un acelerómetro y un giroscopio. El MPU-6050 consta de 6 grados de libertad con seis ejes de salida y por lo tanto seis valores de salida, existen otros con 9 grados de libertad esto quiere decir que en su interior cuentan con un magnetómetro.

El MPU-6050 es un dispositivo basado en tecnología MEMS (Micro Electro Mechanical Systems), este dispositivo esta encapsulado en un solo chip el cual utiliza un protocolo de comunicación I²C. Ya que este dispositivo no puede mostrar ángulos como tal se requiere realizar una serie de cálculos.



Figura 4.10. IMU MPU-6050 (Arduino)

Como se puede apreciar en la figura 4.10 este dispositivo cuenta con una señal para los datos y una para la señal de reloj, su voltaje de entrada es de 3.3V aunque algunos cuentan con un regulador de voltaje que permite emplear un voltaje de entrada de 5V.

En la tabla 8 se muestran algunas de las características del IMU MPU-6050.

MPU-6050	
VDD	2.375v-3.46v
Vlógico	1.71v a VDD
Interfaz serial	I ² C
Pin 3	SCL
Pin 4	SDA

Tabla 8. Especificaciones del MPU6050.

Es importante señalar que la obtención de los datos en bruto es posible sin embargo estos no son ideales, es decir, que tienen un rango de error considerable que puede convertirse en un problema para su aplicación en el cuadricóptero. Por ello es necesario realizar una serie de cálculos para corregir los errores de lectura de la IMU y de esta manera poder utilizarlos en el cálculo del controlador PID.

- Controladora de vuelo:

La controladora de vuelo es el cerebro de un cuadricóptero es el dispositivo más importante ya que con él se puede realizar el control de los motores.

La tarjeta Arduino Mega fue seleccionada como controladora de vuelo debido a que está diseñada para proyectos complejos de robótica dando oportunidad al proyecto de garantizar un buen funcionamiento y retando al usuario a ir más allá de los límites del proyecto. Como se observa en la figura 4.11 la tarjeta Arduino cuenta con una cantidad de entradas analógicas suficientes para obtener las salidas para los ESC, con interrupciones externas para recibir las señales del sistema de comunicación además de contar con una salida de datos (SDA) y un salida de reloj (SCL) necesarias para conectar la IMU MPU-6050.



Figura 4.11. Arduino Mega 2560 (Arduino.c)

Esta tarjeta brinda la capacidad de acceder a librerías y programar en una plataforma amigable y fácil, además de ser una tarjeta económicamente accesible y adecuada para incorporar los sensores a utilizar. Un punto importante por el cual el Arduino mega es utilizado en este proyecto es el voltaje de entrada que puede soportar, ya que el cuadricóptero necesita una batería LiPo de 12V, el margen entre los límites

del voltaje de entrada brinda la seguridad de no provocar un accidente con voltajes parásitos que puedan llegar a existir. Por ultimo una de las ventajas de utilizar esta tarjeta es que brinda la oportunidad a los estudiantes de integrarle más aplicaciones en un futuro para obtener un mejor.

En la tabla 9 se observan con más detalle las especificaciones principales de la tarjeta Arduino mega 2560.

Arduino Mega	
Microcontrolador	ATmega 2560
Tensión de funcionamiento	5 v
Voltaje de entrada (recomendado)	7-12 v
Voltaje de entrada (limites)	6-20 v
E/S digitales	54 de los cuales 15 son pwm
Pines de entrada analógica	16
Corriente continua para pines de I/O	40 mA
Corriente CC para pin 3.3v	50 mA
Memoria flash	128 Kb
SRAM	8 Kb
EEPROM	4 Kb
Velocidad de reloj	16 MHz

Tabla 9. Especificaciones del Arduino Mega.

4.1.3 Prueba A Las Hélices

Verificar que las hélices adquiridas tendrán un buen desempeño al ser montadas como parte del cuadricóptero es muy importante, para ello es necesario realizar una prueba que indique si éstas están o no balanceadas, ya que debido a desperfectos en el proceso de fabricación éstas pueden no estar equilibradas correctamente.

Unas hélices mal balanceadas pueden provocar problemas de estabilización y vibraciones no deseadas en el cuadricóptero, que a su vez pueden afectar el control debido al ruido producido, las hélices a su vez influyen en el empuje que el cuadricóptero puede brindar y en la eficiencia de los motores ya que un desbalance causará en el motor un desgaste prematuro.

Existen dos tipos de balance en las hélices el primero es un balance del estado estático (balance horizontal de la hélice) de la hélice y el segundo es un balance del estado dinámico de la hélice, éste no se realiza en todos los casos, sin embargo, es importante en el caso de hélices grandes (consiste en hacer girar la hélice en el equilibrador y verificar que cada vez se detenga en una posición diferente sin balanceos hacia atrás). Debido a que las hélices a emplear en este cuadricóptero no son de gran tamaño, solo se llevo a cabo el balance estático.

Para verificar el balance estático de cada hélice puede emplearse un equilibrador de hélices; sin embargo esto puede realizarse sin emplear accesorios especiales. Por lo que se realizó la estabilización empleando un tornillo como el eje en el que se coloca la hélice a equilibrar, esto debe instalarse en un soporte con poco rozamiento, de esta manera si la hélice está desequilibrada no se mantendrá en posición paralela al suelo, es decir, la pala de mayor peso bajará más que la de menor peso.

Para corregir el desbalance de las hélices puede agregarse peso a la pala de menor peso, por ejemplo colocando pequeños pedazos de cinta adhesiva hasta lograr el equilibrio o de manera contraria disminuir el peso de la pala más pesada, por ejemplo lijándola hasta alcanzar el equilibrio entre ambas palas.

Al realizar la prueba de balanceo se observó que las hélices permanecían paralelas al suelo y sin ninguna inclinación, como se muestra en la figura 4.12; esto ocurrió con las cuatro hélices, por ello no fue necesario realizar ningún procedimiento de balanceo.



Figura 4.12. Prueba de balance en las hélices

4.1.4 Calibración Del Sistema De Comunicación De Radiofrecuencia

Como se especificó anteriormente se adquirió un sistema de radiofrecuencia, que cuenta con un transmisor y un receptor de radio de seis canales, para lograr la comunicación con el cuadricóptero. Los cuadricópteros requieren de mínimo cuatro canales para controlar sus movimientos, por lo que se requiere configurar estos cuatro canales para que respondan a las direcciones enviadas por el transmisor de radio, estos cuatro canales corresponderán a los tres movimientos básicos de un cuadricóptero roll (CH1), pitch (CH2) y yaw (CH4). El canal 3 corresponderá al arranque de los motores (CH3).

Por lo tanto, se realizó un programa en la tarjeta Arduino para calibrar los canales, asignando cada canal a su respectivo movimiento del cuadricóptero. El programa realizado muestra en pantalla (monitor serial de Arduino) la variación de la magnitud del ancho de pulso captado por el receptor al mover las palancas del transmisor que controlan el movimiento del cuadricóptero. De esta manera se verifica el correcto funcionamiento tanto del receptor como del transmisor y se obtiene un programa que

posteriormente será útil para comunicar los movimientos que el piloto lleva a cabo a distancia, de esta forma los datos que se obtendrán serán procesados mediante una controladora de vuelo y se verán reflejados en la velocidad de los motores y el movimiento del cuadricóptero.

El programa mencionado se basa en el empleo de interrupciones, de esta manera un cambio de estado en la señal captada por el receptor provocará una interrupción en el programa en curso. Para esto se requiere dar de alta cuatro interrupciones externas en la tarjeta Arduino, una por cada canal, de esta manera se logrará que inmediatamente se produzca un cambio en la señal de cualquiera de los canales el programa detectará el cambio y muestre el ancho de pulso actual.

Para activar las interrupciones es necesario revisar la hoja de datos del microcontrolador correspondiente a la tarjeta Arduino MEGA (ATmega 2560) y revisar cuales pines pueden provocar una interrupción externa, para así desarrollar la programación y el posterior conexionado.

De acuerdo a la hoja de datos del ATmega 2560 se produce una interrupción externa cuando ocurre un cambio de estado en alguno de los pines PCINT_{23:0}. Por ello, se decidió activar las interrupciones PCINT0, PCINT1, PCINT2 y PCINT3, que en la tarjeta Arduino corresponden a los pines 53, 52, 51 y 50 respectivamente; tal y como se muestra en la figura 4.13.

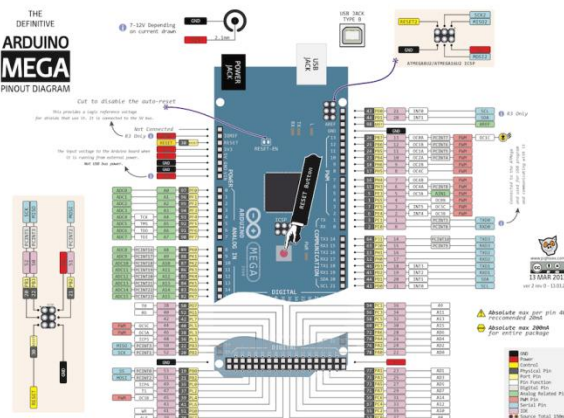


Figura 4.13. Diagrama pinout de la tarjeta Arduino Mega (*bigtronica*)

Una vez hecho esto se procede a activar las interrupciones, de acuerdo a la hoja de datos previamente mencionada, debe activarse el registro de control de interrupciones (Pin Change Interrupt Control Register PCICR). Este registro tiene tres bits (bit 2: PCIE2, bit 1: PCIE1 y bit 0: PCIE 0) y debido a que se eligieron los pines PCINT0, PCINT1, PCINT2 y PCINT3 para causar las interrupciones, corresponde activar el PCIE0, que es específico para las interrupciones PCINT_{7:0}. A su vez cada uno de los pines elegidos para interrupción se activa individualmente mediante el registro PCMSK0.

Posteriormente deben definirse las instrucciones a ejecutar cuando ocurre una interrupción, para ello debe activarse el vector de interrupciones PCINT0 esto se realiza mediante la instrucción ISR(PCINT0_vect). Una vez activadas las interrupciones, se procede a definir las ecuaciones que calcularan el ancho del pulso.

Se tomará como ejemplo el canal uno ya que se aplica el mismo procedimientos a los cuatro canales. Cuando ocurre una interrupción se inicia el conteo del tiempo en microsegundos y se guarda en una variable llamada timer0, si el estado de la variable llamada canal_anterior_1 cambia de 0 a 1, se asigna a la variable el estado alto (1) y se guarda el valor de timer0 a la variable timer1, mientras que el timer0 sigue contando el tiempo. Cuando el canal_anterior_1 cambien de 1 a 0, se le asigna a la variable un estado bajo.

Por lo tanto para calcular el ancho de pulso Δt del canal 1, que representa al movimiento Roll del cuadricóptero, es necesario restar el valor del timer1 al timer0. En la figura 4.14 se muestra lo anterior de manera gráfica.

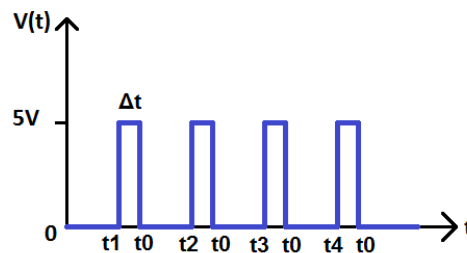


Figura 4.14. Ancho de pulso para cada canal

Debido a que los datos serán observados en el monitor serial de la plataforma Arduino, debe agregarse al código el comando que inicializa la comunicación serial (`Serial.begin(9600)`) y posteriormente se agrega una rutina para imprimir los valores obtenidos. El programa final se muestra en la sección de anexos en el subtítulo “6.4.1 Programa de prueba para el sistema de radiofrecuencia”.

Una vez hecho esto se realizan las conexiones necesarias entre la tarjeta Arduino Mega y el receptor de radio. En la figura 4.15 y 4.16, se muestra el diagrama de conexión y su conexión física respectivamente.

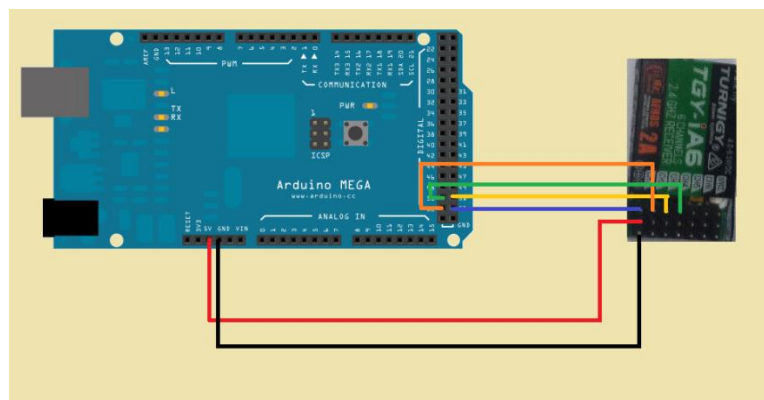


Figura 4.15. Diagrama de conexión entre Arduino Mega y receptor de radio

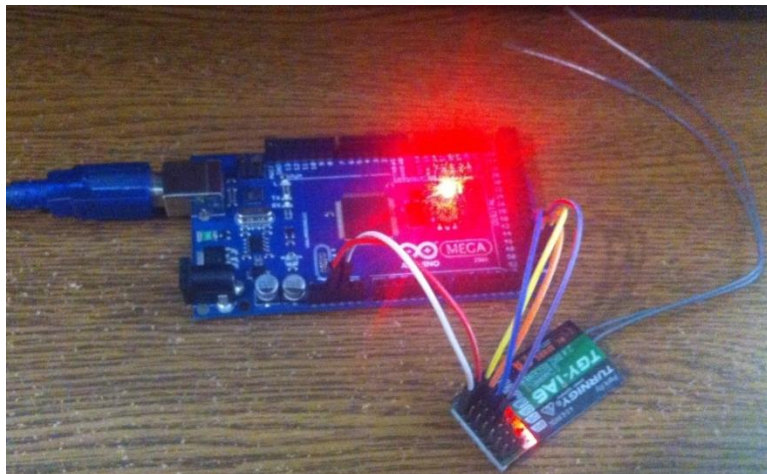


Figura 4.16. Conexión física entre Arduino Mega y receptor de radio

Después se procede a verificar el programa y a subirlo, para visualizar los datos se abre el monitor serie y se comprueba que los datos que ahí aparecen sean correctos. Para comenzar se enciende el transmisor de radio y si las dos palancas se encuentran en su posición central todos los valores deben ser aproximadamente 1500µs (Figura 4.17).

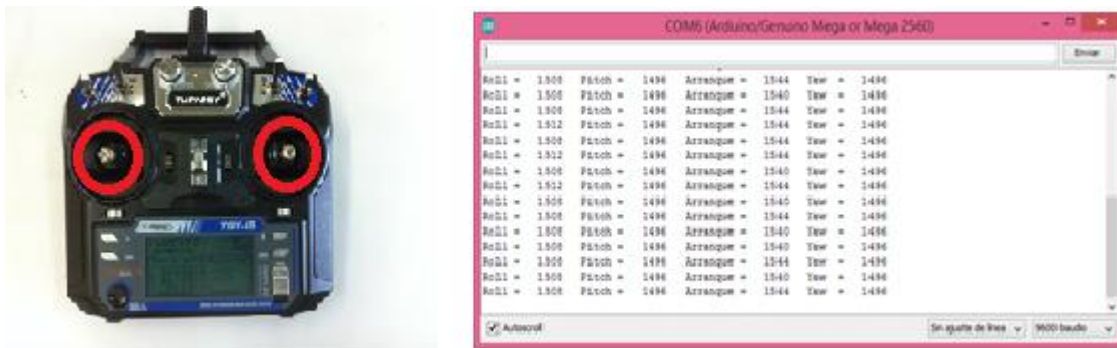


Figura 4.17. Datos correspondientes a posición central

El movimiento de la palanca de la izquierda hacia arriba o abajo corresponde al canal 3 que es el arranque de los motores, mientras que mover esta palanca a la izquierda o a la derecha modifica el ancho de pulso del canal 4 que corresponde al movimiento yaw del cuadricóptero.

El movimiento de la palanca derecha hacia la izquierda o la derecha corresponde al movimiento canal 1 que es a su vez el movimiento roll, mientras que mover esta palanca hacia arriba o hacia abajo modifica el valor del canal 2 que corresponde al movimiento pitch.

Al realizar el movimiento que se muestra en la figura 4.18 el ancho de pulso del arranque baja a aproximadamente $1000\mu\text{s}$, mientras que, si la palanca se mueve al lado contrario se obtendrá un pulso de aproximadamente $2000\mu\text{s}$.

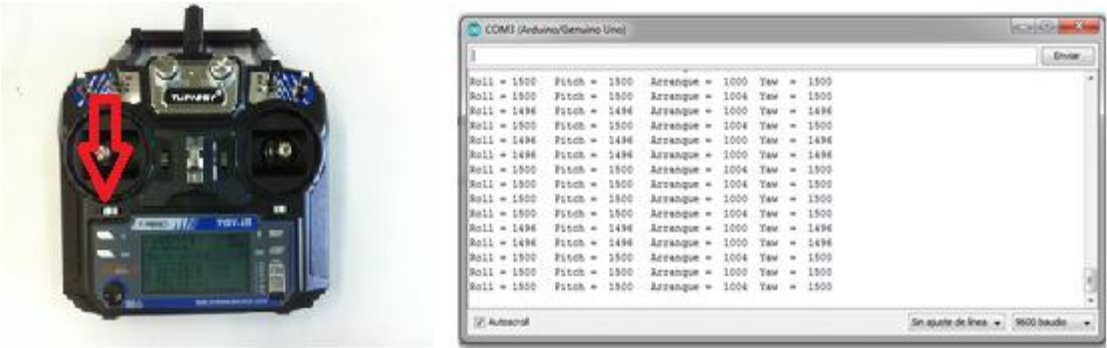


Figura 4.18. Datos correspondientes a la posición de las palancas

Al realizar el movimiento que se muestra en la figura 4.19 el ancho de pulso de yaw se incrementa a aproximadamente $2000\mu\text{s}$, mientras que, si la palanca se mueve al lado contrario se obtendrá un pulso de aproximadamente $1000\mu\text{s}$.

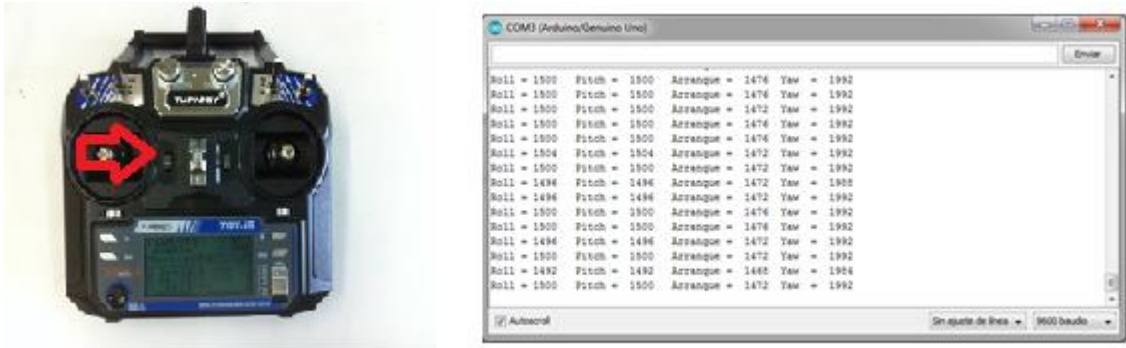


Figura 4.19. Datos correspondientes a la posición de las palancas

Al realizar el movimiento que se muestra en la figura 4.20 el ancho de pulso de Roll decremento a aproximadamente $1000\mu\text{s}$, mientras que, si la palanca se mueve al lado contrario se obtendrá un pulso de aproximadamente $2000\mu\text{s}$. A su vez queda demostrado que al mover dos palancas ambos valores se modifican, en este caso el valor roll y el valor de arranque.

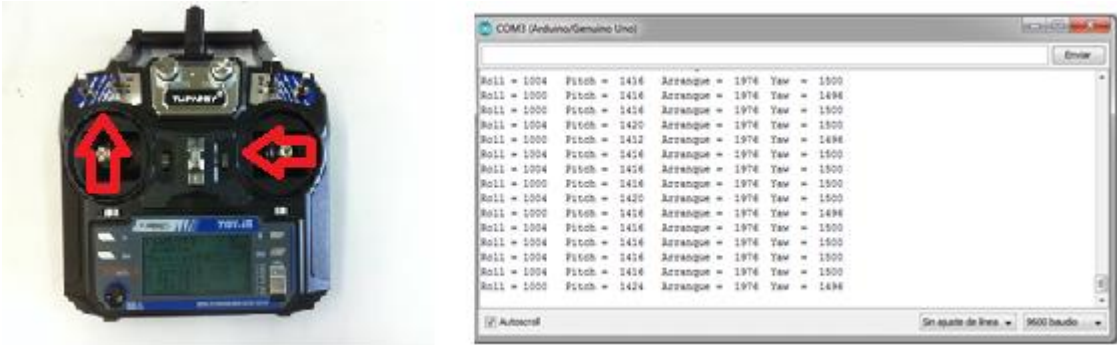


Figura 4.20. Datos correspondientes a la posición de las palancas

Al realizar el movimiento que se muestra en la figura 4.21 el ancho de pulso pitch incremento a aproximadamente $2000\mu\text{s}$, mientras que, si la palanca se mueve al lado contrario se obtendrá un pulso de aproximadamente $1000\mu\text{s}$.

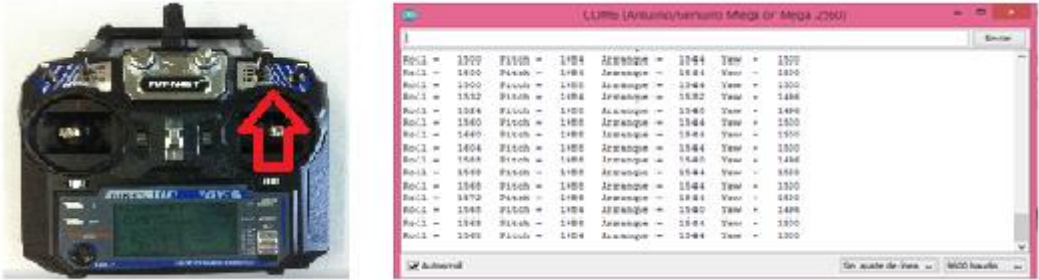


Figura 4.21. Datos correspondientes a la posición de las palancas

Para verificar estos resultados de una manera diferente se decidió observar la señal de entrada del receptor mediante un osciloscopio y de esta manera comprobar que la variación de la magnitud del ancho de pulso coincida con los datos obtenidos mediante Arduino. Esta prueba se realizó con cada uno de los canales, pero para fines prácticos se ejemplificará solo el canal uno ya que el proceso es repetitivo. En la figura 4.22 se muestra la conexión a implementar, en este caso la tarjeta Arduino uno solo se empleó para alimentar con 5V al receptor.

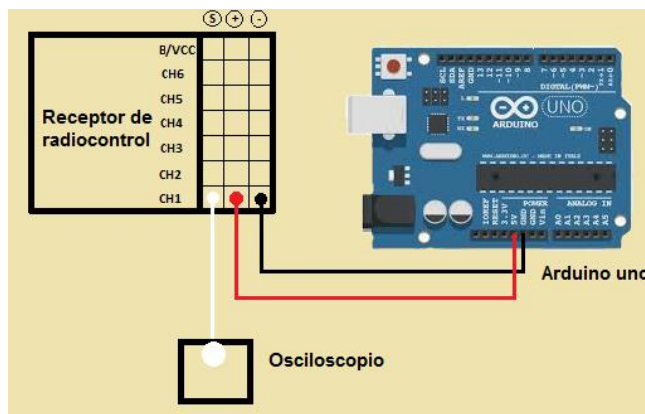


Figura 4.22. Diagrama para observar los pulsos en la entrada del receptor en el osciloscopio

En la figura 4.23 se muestra el armado físico y se observa una onda de $1500\mu\text{s}$.

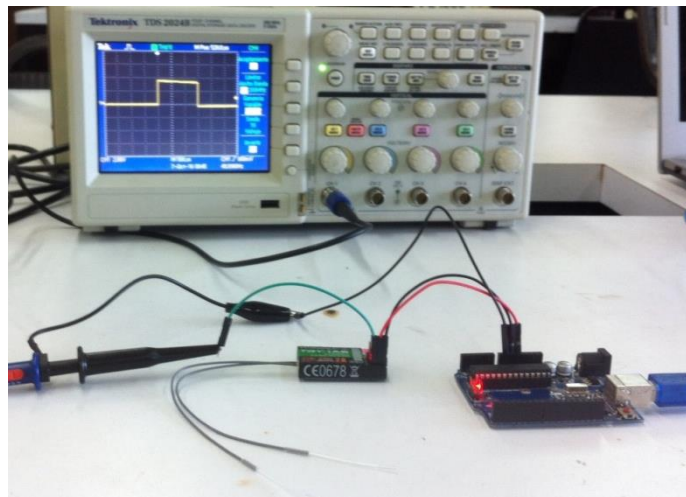


Figura 4.23. Conexiones para observar los pulsos de entrada del receptor ($1500\mu\text{s}$)

Al mover la palanca del canal 1 (Roll) totalmente a la izquierda la magnitud del ancho de pulso disminuyo a $1000\mu\text{s}$ (figura 4.24).

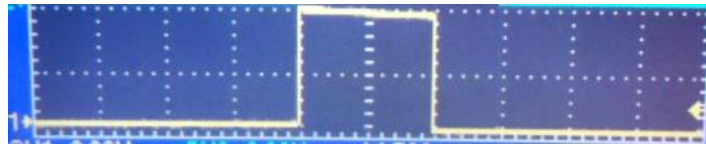


Figura 4.24. Pulso mínimo de entrada canal 1 ($1000\mu\text{s}$)

Al mover la palanca del canal 1 (Roll) totalmente a la derecha la magnitud del ancho de pulso incremento a $2000\mu\text{s}$ (figura 4.25). Estos datos como se puede observar concuerdan con lo obtenido mediante el programa implementado.

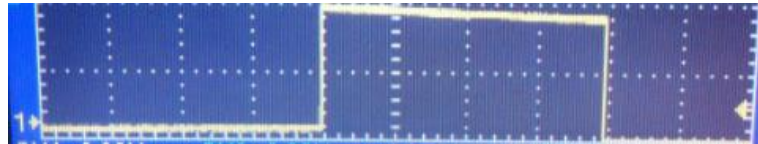


Figura 4.25. Pulso mínimo de entrada canal 1 ($1000\mu\text{s}$)

4.1.5 Calibración De Los ESC'S Y Prueba A Los Motores

Los ESC's deben ser calibrados previamente a ser utilizados, existen varias formas de llevar a cabo este proceso, sin embargo, se eligió realizar la calibración manual para cada ESC, la cual se lleva a cabo mediante la emisora de radiofrecuencia. Este proceso consiste en indicar al ESC cuál es el rango entre la posición mínima ($1000\mu\text{s}$) y máxima ($2000\mu\text{s}$) de la palanca de arranque de la emisora (throttle).



Figura 4.26. Calibración de ESC con palanca de arranque en posición máxima

Primeramente se procede a subir la palanca de arranque (canal 3) a su posición máxima, una vez hecho esto se enciende la emisora (Figura 4.26); de esta manera, cuando el ESC sea encendido estará automáticamente en modo calibración y fijará de esta manera el máximo de la palanca de arranque. Posteriormente se conecta el variador al canal 3 del receptor como se muestra en la figura 4.27, teniendo en cuenta que los pines del ESC y la emisora queden conectados en el orden correcto.



Figura 4.27. Calibración de ESC con palanca de arranque en posición máxima

Después se conecta la batería Li-Po al variador, de esta manera se escucha el sonido que produce la comunicación entre la emisora y el receptor. El receptor cuenta con un led que se enciende al ser alimentado y que indica que está en funcionamiento.

Como consecuencia a esto el variador emite dos sonidos, cuando esto ocurre la palanca de arranque debe bajarse inmediatamente a su posición mínima. Así se habrán calibrado los valores máximo y mínimo de la palanca de arranque, la confirmación de la correcta calibración la emite el variador mediante una serie de sonidos. Esta prueba debe realizarse para cada variador en la figura 4.28 se muestra el diagrama de conexión.

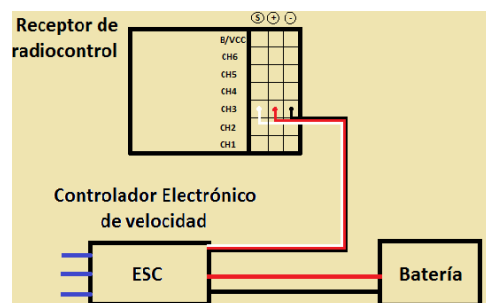


Figura 4.28. Diagrama de conexión para calibración de ESC

Para verificar que los motores funcionen correctamente se realizó una prueba a cada uno de ellos, para realizar esta prueba las conexiones son las mismas que para la calibración de los ESC's a excepción de que en este caso se conecta el motor a los cables que se encuentran en el otro extremo del ESC (Figura 4.29). Esta prueba consiste en observar la variación de velocidad en el motor respecto a la variación del ancho de pulso que se le aplica mediante la transmisora de radiofrecuencia.

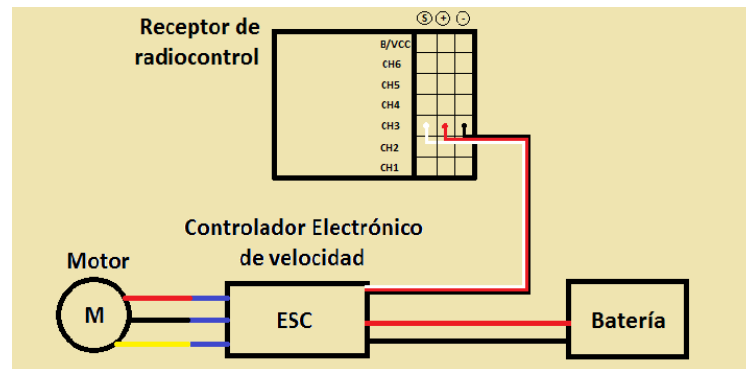


Figura 4.29. Diagrama de conexión para probar los motores

La conexión en físico se muestra en la figura 4.30.

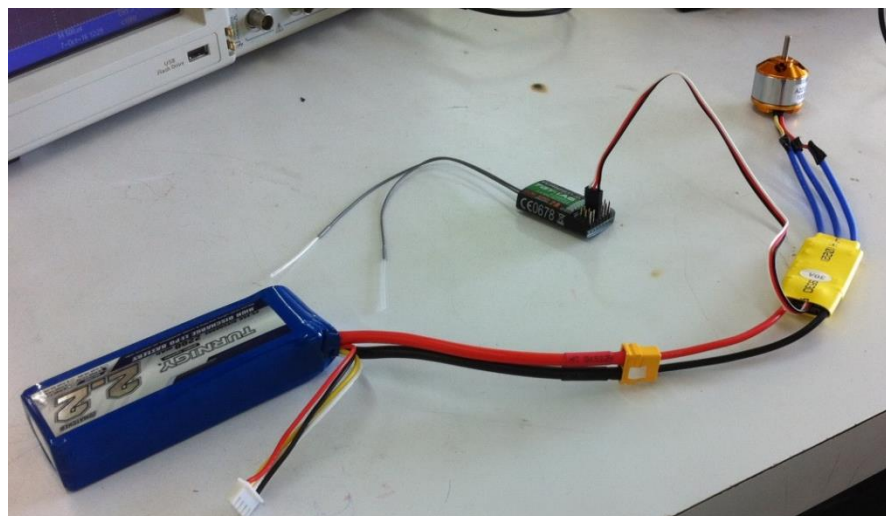


Figura 4.30. Conexión para probar los motores

Para observar la variación de ancho de pulso se conecta la sonda del osciloscopio al pin de señal del receptor. Al inicio con la palanca de arranque al mínimo se obtiene la siguiente señal (Figura 4.31), esta señal corresponde a $1000\mu\text{s}$ ya que el osciloscopio está en una escala de $500\mu\text{s}$ por cuadro. En este punto el motor está parado.

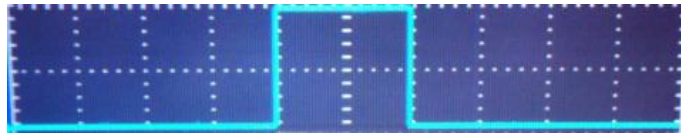


Figura 4.31. Ancho de pulso con palanca de arranque en posición mínima

Si la palanca de arranque se mueve hasta la mitad de su posición total se puede notar que el motor se acelera y se obtiene el siguiente pulso que equivale a $1500\mu\text{s}$ (Figura 4.32).

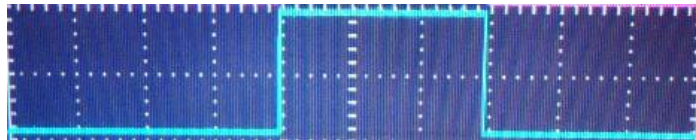


Figura 4.32. Ancho de pulso con palanca de arranque en posición media

Si la palanca de arranque se mueve hasta su posición máxima se puede notar que el motor se acelera a su máxima capacidad y se obtiene el siguiente pulso que equivale a $2000\mu\text{s}$ (Figura 4.33).

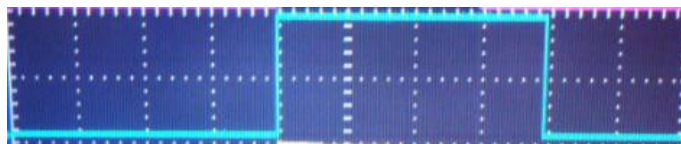


Figura 4.33. Ancho de pulso con palanca de arranque en posición máxima

De esta manera es posible verificar que los motores funcionan y varían su velocidad en proporción a la posición de la palanca de arranque y que no presentan algún daño o defecto de fábrica.

4.1.6 Calibración IMU

Como se ha mencionado anteriormente el MPU-6050 es el sensor mediante el cual se realizaran las correcciones del controlador, sin embargo este dispositivo nos entrega datos en bruto. Por lo tanto, mediante un código se realizaran los cálculos correspondientes para corregir el error de los datos que son enviados por el MPU-6050, para esto es necesario iniciar el código del programa utilizando la librería <wire.h> la cual contiene el protocolo de comunicación I²C, una vez declarada esta librería es necesario saber cómo se activa en el MPU-6050 el protocolo de comunicación I²C.

Por ello es necesario recurrir a la hoja de datos del mismo, la cual indica que es el registro 0x68, también es importante buscar en la hoja de datos el factor de sensibilidad tanto del acelerómetro (16384) como del giroscopio (131), esto es para obtener datos más aproximados que los datos en bruto y por último se definirá una constante para la conversión de radianes a grados, esta constante está dada por la ecuación (7).

$$180/\pi = 57.295779 \quad \text{Ec. (7)}$$

Posteriormente se declararan las variables para el acelerómetro y el giroscopio en los ejes x, y, z y los ángulos correspondientes. Una vez que se definan las constantes y las variables es necesario inicializar el I²C como maestro y la transmisión con el MPU-6050, además activar el registro solicitado (0x6B) y establecer el registro de inicio en ceros; por ultimo terminar la transmisión y configurar el monitor serial para observar los resultados.

La parte del código que se ejecutara cíclicamente se dividirá en 5 partes.

- Lectura de los valores del acelerómetro en bruto.

Para leer los datos del acelerómetro es necesario inicializar la transmisión con el MPU-6050, pedir el registro 0x3B el cual corresponde al acelerómetro en X, a partir del registro 0x3B se piden 6 bytes ya que a cada variable del acelerómetro (AcelX

,AcelY y AcelZ) se le añadirá al byte alto el bajo y se guardara en cada variable ocupando dos bytes cada uno.

- Cálculo de los ángulos X y Y con los datos en bruto del acelerómetro.

Para calcular el ángulo es necesario comprender que la IMU detecta la fuerza de gravedad sobre el eje que esta perpendicular al suelo como se aprecia en la figura 4.34 si la IMU esta perpendicular al suelo el eje Z detecta los 9.8 m/s^2 de la gravedad, por lo que si se gira 90° ahora el eje X el que detecta los 9.8 m/s^2 .

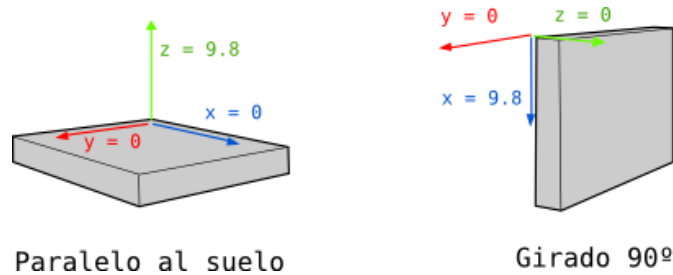


Figura. 4.34. Representación gráfica de la IMU (Robologs)

Ahora que se sabe cómo funciona y que medición existe en los ejes del acelerómetro es posible calcular el ángulo X y Y con la ecuación (8). Es importante mencionar que las aceleraciones (AcelX, AcelY y AcelZ) tienen que dividirse sobre el factor de sensibilidad anteriormente mencionado al colocarlo en la formula.

$$\text{Ángulo en X} = \arctg\left(\frac{y}{\sqrt{x^2 + z^2}}\right)$$

Ec. (8)

$$\text{Ángulo en Y} = \arctg\left(\frac{x}{\sqrt{y^2 + z^2}}\right)$$

Como se observa no es posible calcular el ángulo en el eje Z debido a que los cálculos son en base a la gravedad, por lo tanto, para calcular este ángulo sería necesario contar con un magnetómetro.

- Lectura de los valores del giroscopio en bruto.

Para leer los datos del giroscopio es necesario inicializar la transmisión con el MPU-6050, pedir el registro 0x43 el cual corresponde al giroscopio en X, a partir del registro 0x43 se piden 4 bytes ya que a cada variable del acelerómetro (GiroX y GiroY) se le añadirá al byte alto el bajo y se guardara en cada variable ocupando dos bytes cada uno.

- Cálculo de los ángulos X, Y y Z con los datos en bruto del giroscopio.

El giroscopio mide la velocidad angular, es decir, el número de grados que gira en un segundo por lo que el cálculo resulta de una manera sencilla. Como se aprecia en la ecuación (9), es necesario conocer el ángulo anterior más el ángulo que marca el giroscopio por el tiempo que transcurre desde que se realizó el cálculo de esta fórmula Δt .

$$\text{Ángulo en } X = \text{Ángulo en } X(\text{anterior}) + \text{Giro}X(\Delta t) \quad \text{Ec. (9)}$$

El código se ejecutara en un tiempo del orden de los milisegundos, así que solo se dividirá el valor de la lectura en bruto sobre el factor de sensibilidad.

- Filtro complementario

Una vez que se obtuvieron los ángulos es momento de realizar un cálculo que elimine el ruido debido a que la IMU puede ser sometida a perturbaciones que logran hacer que las mediciones sean inestables. Es por esto que se recurrirá al filtro complementario, este filtro combina un filtro pasa altas y un filtro pasa bajas por lo que la fórmula es la resultante de combinar estos dos filtros como se muestra en la ecuación (10).

$$\text{Ángulo}(\text{filtro}) = 0.98 \left(\text{Ángulo}(\text{filtro}) + (\text{Ángulo}(\text{Giro}))(\Delta t) \right) + 0.02((\text{Ángulo}(\text{Acel})) \quad \text{Ec. (10)}$$

Esta fórmula es igual para el ángulo X y Y. Las constantes de 0.98 y 0.02 pueden ser cambiadas siempre y cuando la suma de ambas sea de 1. El código final realizado se encuentra en el apartado anexos “6.4.2 Programa Para Calibración De La IMU”.

Una vez terminada la programación se procederá a la conexión física de los pines entre la tarjeta Arduino mega y la IMU MPU-6050, para lo cual será necesario conectar los pines correspondientes SDA y SCL del MPU-6050 a los SDA y SCL de la tarjeta de Arduino mega que corresponden a los pines 20 y 21. En la figura 4.35 se observa el diagrama de conexiones entre los dos dispositivos.

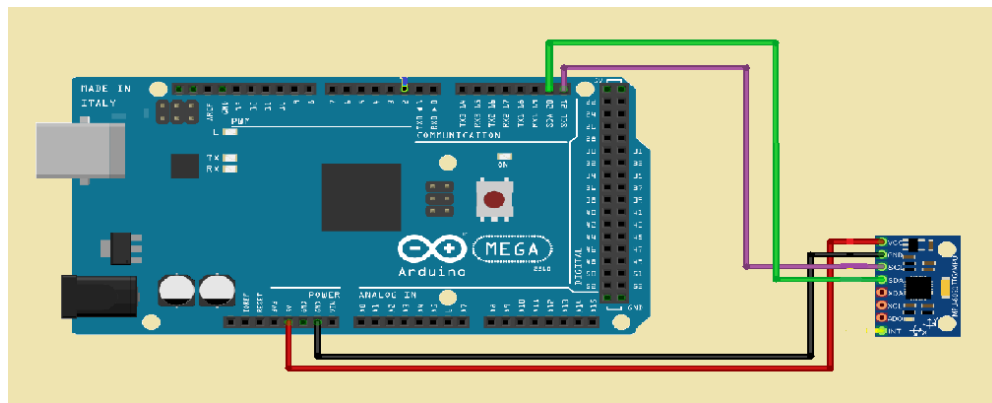


Figura 4.35. Diagrama de conexión entre Arduino Mega y la IMU MPU-6050.

Posteriormente se verifica el código del programa y se procede a cargarlo a la tarjeta Arduino. En la figura 4.36 se muestra la conexión física de los dispositivos.

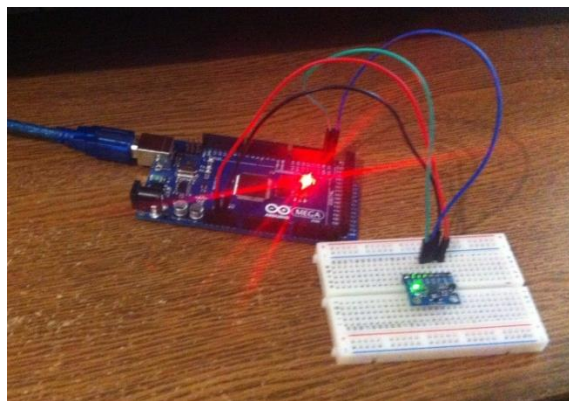


Figura 4.36. Conexión física entre Arduino Mega y la IMU MPU-6050.

Ya que se subió el código a la tarjeta Arduino Mega y se verifico el programa, se inician las pruebas correspondientes para verificar las mediciones de los ángulos. Para probar las mediciones se utilizó una regla que puede acomodarse en cualquier ángulo como se observa en la figura 4.37. Se puede observar que para estas pruebas se utilizó una tarjeta Arduino Uno debido a que las conexiones con el Arduino mega no permitían posicionar bien el protoboard donde se encuentra la IMU. Por lo tanto las conexiones del SDA y SCL del Arduino Mega se cambiaron por A4 y A5 respectivamente.

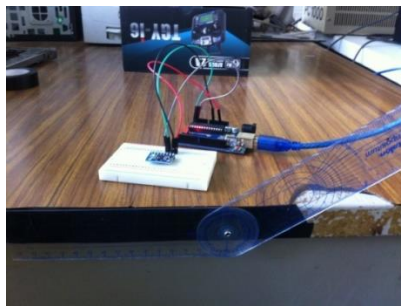
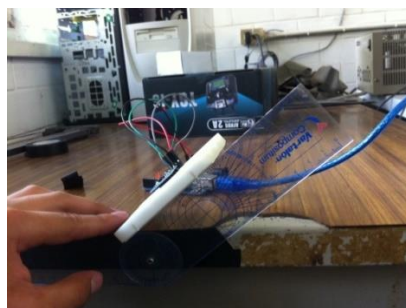


Figura 4.37. Diseño para la medición de los ángulos.

La primera prueba se realizó con un ángulo de 40°. En la figura 4.38 se muestran los datos obtenidos en el monitor serial correspondientes a la posición del IMU. Se realizó la prueba colocando el protoboard de forma horizontal paralelo a la mesa y rotándolo para verificar que la medición sea en el ángulo X correspondiente al movimiento roll con signo positivo. Es importante mencionar que la medición en el ángulo Roll se debe a la inestabilidad de la conexión entre el protoboard y el IMU.



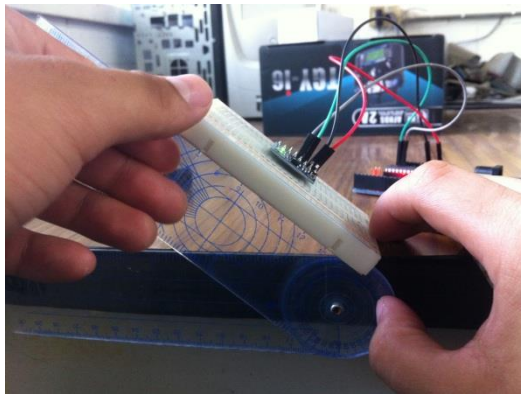
```

Angulo X      : 40.06 -- Angulo Y      : 2.82
Angulo X      : 40.06 -- Angulo Y      : 2.83
Angulo X      : 40.08 -- Angulo Y      : 2.82
Angulo X      : 40.08 -- Angulo Y      : 2.83
Angulo X      : 40.08 -- Angulo Y      : 2.83
Angulo X      : 40.08 -- Angulo Y      : 2.83
Angulo X      : 40.08 -- Angulo Y      : 2.83
Angulo X      : 40.08 -- Angulo Y      : 2.83
Angulo X      : 40.08 -- Angulo Y      : 2.85
Angulo X      : 40.09 -- Angulo Y      : 2.86
Angulo X      : 40.09 -- Angulo Y      : 2.87
Angulo X      : 40.09 -- Angulo Y      : 2.88
Angulo X      : 40.09 -- Angulo Y      : 2.88
Angulo X      : 40.08 -- Angulo Y      : 2.89
Angulo X      : 40.07 --

```

Figura 4.38. Datos correspondientes a la posición de la IMU.

La segunda medición corresponde a un ángulo roll en sentido contrario para verificar que el ángulo sea negativo ya que es muy importante saber en qué sentido del eje se mueve la IMU. En la figura 4.39 se observa como el protoboard está colocado en dirección opuesta a la prueba anterior.



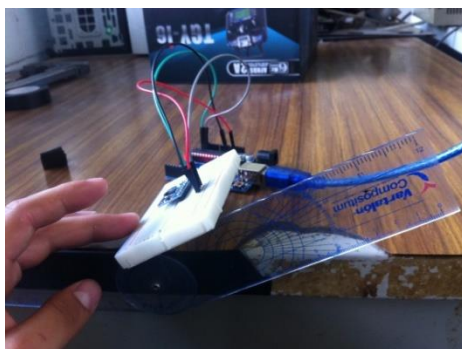
```

Angulo X      : -40.62  --  Angulo Y      : -0.15
Angulo X      : -40.61  --  Angulo Y      : -0.15
Angulo X      : -40.61  --  Angulo Y      : -0.15
Angulo X      : -40.60  --  Angulo Y      : -0.15
Angulo X      : -40.58  --  Angulo Y      : -0.19
Angulo X      : -40.57  --  Angulo Y      : -0.23
Angulo X      : -40.56  --  Angulo Y      : -0.24
Angulo X      : -40.56  --  Angulo Y      : -0.24
Angulo X      : -40.55  --  Angulo Y      : -0.25
Angulo X      : -40.54  --  Angulo Y      : -0.25
Angulo X      : -40.52  --  Angulo Y      : -0.26
Angulo X      : -40.51  --  Angulo Y      : -0.27
Angulo X      : -40.50  --  Angulo Y      : -0.27
Angulo X      : -40.50  --  Angulo Y      : -0.26
Angulo X      : -40.49  --

```

Figura 4.39. Datos correspondientes a la posición de la IMU.

Para esta tercera prueba se analizó la medición en el ángulo Y correspondiente al movimiento pitch por lo que se colocó el protoboard de forma distinta, como se muestra en la figura 4.40, aquí también es necesario que la medición aparte de ser exacta tenga el signo que debe de llevar.



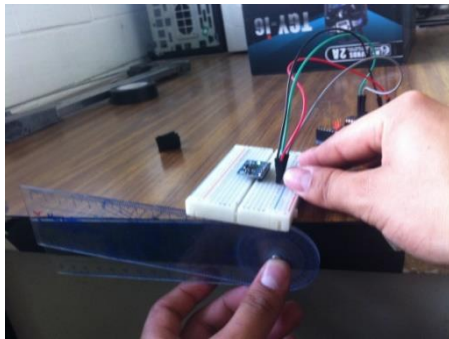
```

Angulo X      : 0.72  --  Angulo Y      : 30.54
Angulo X      : 0.72  --  Angulo Y      : 30.56
Angulo X      : 0.73  --  Angulo Y      : 30.54
Angulo X      : 0.74  --  Angulo Y      : 30.56
Angulo X      : 0.75  --  Angulo Y      : 30.54
Angulo X      : 0.75  --  Angulo Y      : 30.53
Angulo X      : 0.75  --  Angulo Y      : 30.50
Angulo X      : 0.76  --  Angulo Y      : 30.48
Angulo X      : 0.75  --  Angulo Y      : 30.48
Angulo X      : 0.76  --  Angulo Y      : 30.45
Angulo X      : 0.76  --  Angulo Y      : 30.44
Angulo X      : 0.77  --  Angulo Y      : 30.43
Angulo X      : 0.78  --  Angulo Y      : 30.42
Angulo X      : 0.79  --  Angulo Y      : 30.40
Angulo X      : 0.80  --

```

Figura 4.40. Datos correspondientes a la posición de la IMU.

Para la última prueba se cambió la dirección de la IMU para corroborar la medición del ángulo en otro rango más pequeño y su signo como se muestra en la figura 4.41.



```

Angulo X      : -0.76  --  Angulo Y      : -10.11
Angulo X      : -0.75  --  Angulo Y      : -10.12
Angulo X      : -0.75  --  Angulo Y      : -10.13
Angulo X      : -0.75  --  Angulo Y      : -10.14
Angulo X      : -0.75  --  Angulo Y      : -10.15
Angulo X      : -0.75  --  Angulo Y      : -10.16
Angulo X      : -0.75  --  Angulo Y      : -10.16
Angulo X      : -0.75  --  Angulo Y      : -10.17
Angulo X      : -0.75  --  Angulo Y      : -10.18
Angulo X      : -0.75  --  Angulo Y      : -10.18
Angulo X      : -0.75  --  Angulo Y      : -10.18
Angulo X      : -0.74  --  Angulo Y      : -10.18
Angulo X      : -0.74  --  Angulo Y      : -10.18
Angulo X      : -0.74  --  Angulo Y      : -10.19
Angulo X      : -0.74  --  Angulo Y      : -10.19
Angulo X      : -0.74  --

```

Figura 4.41. Datos correspondientes a la posición de la IMU.

Como ya se mencionó anteriormente no es posible calcular el ángulo en el eje Z pero es importante tener una lectura en el movimiento yaw debido a que es un dato importante para el cálculo del pulso que se enviarán a los ESC's, lo que importa de estos datos es el signo, ya que como se muestra en la figura 4.42 el signo determina en qué sentido se gira la IMU. La IMU muestra en el monitor serial unos valores de -15 a 15 cuando está en movimiento ya que cuando permanece estática se estabiliza en 0. En el código se calcula de la siguiente forma:

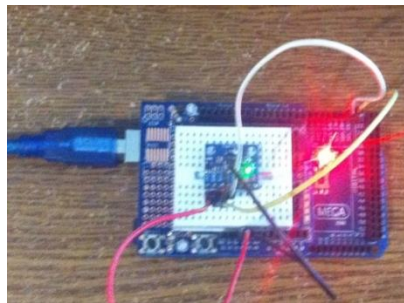
$$\text{imu_vp_yaw} = 0.98 * (\text{imu_vp_yaw} + A_giro_Z * 0.010) + 0.004;$$

Donde 0.004 es una constante calculada de un promedio, en base al offset observado en varias pruebas.

```

Angulo Z (Yaw) : 0.75
Angulo Z (Yaw) : 0.77
Angulo Z (Yaw) : 0.79
Angulo Z (Yaw) : 0.80
Angulo Z (Yaw) : 0.82
Angulo Z (Yaw) : 0.83
Angulo Z (Yaw) : 0.85
Angulo Z (Yaw) : 0.86
Angulo Z (Yaw) : 0.88
Angulo Z (Yaw) : 0.89
Angulo Z (Yaw) : 0.90
Angulo Z (Yaw) : 0.92
Angulo Z (Yaw) : 0.93
Angulo Z (Yaw) : 0.95
Angulo Z (Yaw) : 0.96

```

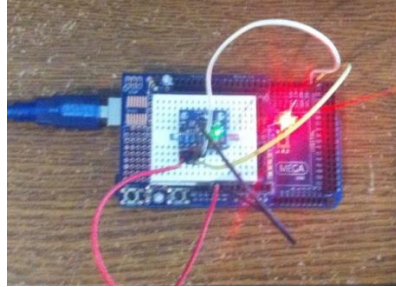


a)

```

Angulo Z (Yaw) :0.62
Angulo Z (Yaw) :1.30
Angulo Z (Yaw) :2.05
Angulo Z (Yaw) :2.53
Angulo Z (Yaw) :3.22
Angulo Z (Yaw) :3.72
Angulo Z (Yaw) :4.28
Angulo Z (Yaw) :4.43
Angulo Z (Yaw) :4.66
Angulo Z (Yaw) :4.85
Angulo Z (Yaw) :4.90
Angulo Z (Yaw) :5.00
Angulo Z (Yaw) :5.10
Angulo Z (Yaw) :5.26
Angulo Z (Yaw) :5.45

```

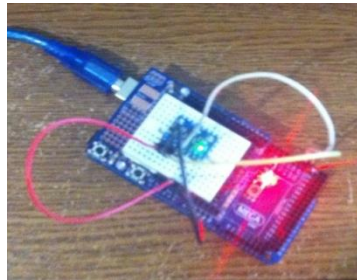


b)

```

Angulo Z (Yaw) :-9.16
Angulo Z (Yaw) :-8.92
Angulo Z (Yaw) :-8.61
Angulo Z (Yaw) :-7.94
Angulo Z (Yaw) :-7.79
Angulo Z (Yaw) :-7.60
Angulo Z (Yaw) :-7.42
Angulo Z (Yaw) :-7.23
Angulo Z (Yaw) :-7.02
Angulo Z (Yaw) :-6.77
Angulo Z (Yaw) :-6.53
Angulo Z (Yaw) :-6.34
Angulo Z (Yaw) :-6.18
Angulo Z (Yaw) :-6.03
Angulo Z (Yaw) :-5.88

```



c)

Figura 4.42. Datos del movimiento yaw. a) IMU estabilizado, b) IMU en movimiento anti-horario c) IMU en movimiento horario.

4.1.7 Encendido Y Control Simultáneo De Velocidad En Cuatro Motores

En este apartado se describe el programa implementado para encender y variar la velocidad de los cuatro motores simultáneamente, para ello se incorpora la tarjeta Arduino Mega para recibir las interrupciones del receptor y procesarlas mediante un código preestablecido y en base a esto, enviar una salida a los ESC's que a su vez variarían la velocidad de los motores.

El código a implementar para este fin es similar al empleado para calibrar el sistema de radiofrecuencia; pero a diferencia de este último, en este código no se requiere la impresión de valores ya que se centra en activar los cuatro motores brushless del cuadricóptero al mismo tiempo. Esto es de gran importancia, debido a que al implementar el control PID será necesario que la tarjeta Arduino lleve a cabo el control y que a su vez al presentarse alguna interrupción provocada por el

transmisor, esta pueda ejecutar instrucciones específicas para lograr llevar a cabo las instrucciones establecidas por el piloto a distancia. De esta manera la tarjeta Arduino Mega también tiene la función de enviar pulsos de salida al ESC para que este sea capaz de variar la velocidad y en consecuencia el rumbo de los motores.

Debido a que la mayor parte del código de este apartado ha sido descrito anteriormente (apartado “4.1.4 Calibración Del Sistema De Comunicación De Radiofrecuencia”), solo se describirán las partes agregadas al código.

Primeramente se configura el registro del puerto digital K, estableciendo sus cuatro bits más altos (bits 4, 5,6 y 7) como salidas, esto se refleja en la siguiente instrucción `DDRK |= B11110000`. Se eligieron estos bits únicamente porque son pines digitales, pero en realidad pueden emplearse cualquiera de los pines digitales mientras sean distintos a los elegidos para las interrupciones externas que reciben la señal del transmisor.

Posteriormente se inicia un conteo de tiempo en microsegundos y se almacena en una variable llamada `timerX`.

El ancho de pulso de los ESC's puede variar, por ello es necesario establecer una frecuencia de actualización y dedicar una parte del código a asegurar que ésta sea siempre igual al valor establecido, en este caso se decidió que sea igual a 250Hz (que equivale a 4000 μ s), es decir, que el controlador se actualizará para verificar nuevas instrucciones 250 veces por segundo.

Para lograr que la frecuencia de actualización sea siempre igual a 250Hz se define un ciclo `while`, que mediante una condición, permite que los pulsos enviados a los ESC's inicien después de transcurridos 4000 μ s. Una vez hecho esto el valor del `timerX` se reestablece y se ponen en alto las salidas que van a los ESC's (salidas digitales, cuatro bits más altos del puerto K), posteriormente se calcula el tiempo en que el pulso enviado a los ESC's cae a un nivel bajo (`timer_canal_x`; `x=1, 2, 3 o 4`) esto asegura que la frecuencia de actualización sea siempre igual a 250Hz, ya que una vez transcurridos los 4000 μ s el ciclo volverá a comenzar y se actualizará si es que ocurre un cambio en el ancho de pulso; este cálculo se realiza para cada uno de

los ESC's, todos se calculan en base al canal receptor 3 debido a que corresponde a la palanca de arranque y es la encargada de encender los motores.

Si las salidas, es decir los cuatro bits más altos del puerto K, se encuentran en bajo entrarán a un nuevo ciclo para asegurar que el timer correspondiente a cada canal (timer_canal_x; x=1, 2, 3 o 4) no ha tenido un cambio en más de 4000µs, por lo cual la salida correspondiente a cada canal será puesta en bajo. El código completo realizado para esta prueba se muestra en el apartado anexos "6.4.3 Programa Para El Encendido Y Control Simultáneo De Velocidad En Cuatro Motores".

Para realizar la conexión entre la batería y los cuatro ESC's con sus respectivos motores fue necesario diseñar una tarjeta distribuidora de potencia en una tarjeta de circuito impreso (PCB). El PCB se diseñó colocando los ESC's en paralelo a la batería, de esta manera los cuatro ESC's recibirán el voltaje integro de la batería, mientras que la corriente proporcionada por la batería se repartirá entre los cuatro ESC's.

En la figura 4.43 se muestra el PCB descrito anteriormente.

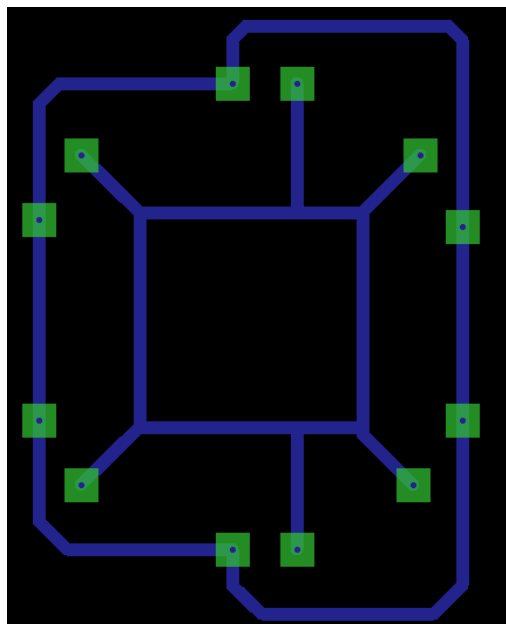


Figura 4.43. PCB de la tarjeta distribuidora de potencia

En la figura 4.44 se muestra el diagrama de conexión para encender los cuatro motores del cuadricóptero simultáneamente. Debido a que la tarjeta Arduino Mega requiere de alimentación; se decidió que la energía necesaria se tomará de la batería, para proteger a la tarjeta Arduino de posibles cortos circuitos se colocó un diodo entre el positivo de la batería y el pin Vin.

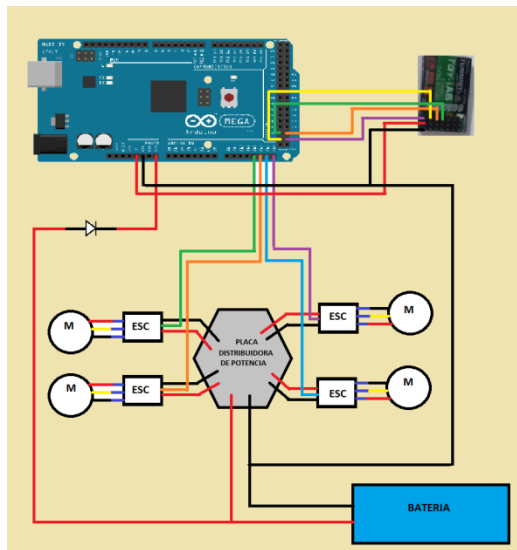


Figura 4.44. Diagrama de conexión para el encendido de cuatro motores simultáneamente

En la figura 4.45 se muestra la conexión física realizada para encender los cuatro motores del cuadricóptero simultáneamente.

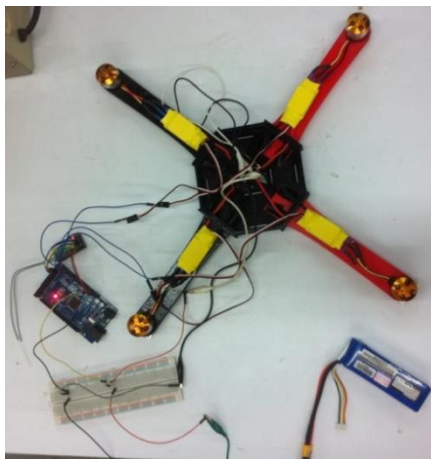


Figura 4.45. Conexión física para el encendido de cuatro motores simultáneamente

4.1.8 Alarma De Bajo Voltaje En La Batería

Los motores necesitan de toda la capacidad de la batería, debido a esto es necesario saber cuánto voltaje está proporcionando la batería a los ESC's y emitir una alerta cuando este caiga por debajo de 10.5V ya que si la batería se descarga por debajo de este valor, podría causar serios problemas al cuadricóptero. El código realizado se muestra en el apartado de anexos "6.4.4 Prueba A La Alarma De Bajo Voltaje En La Batería".

La tarjeta Arduino Mega cuenta con entradas analógicas con las que puede leer valores en un rango de 0 a 5 volts, donde 5 volts corresponde a 1023 datos debido a que tiene una resolución de 10 bits. Sabiendo que la batería entrega un voltaje máximo de 12.3 V, es necesario diseñar un divisor de voltaje para limitar el voltaje que leerá la entrada analógica a 5V, esto se puede calcular con la fórmula de la ecuación (11).

$$V_{in} = \left(\frac{R_x}{R_y + R_x} \right) V_t \quad \text{Ec. (11)}$$

Donde V_{in} es el voltaje que entrara a la tarjeta Arduino en un rango de 0 a 5 volts, R_x y R_y son resistencias elegidas en un rango de 0 a 10 k Ω y V_t es el voltaje máximo de la batería. Por lo tanto, como se aprecia en la ecuación (11) existen dos variables desconocidas R_x y R_y . Se eligió R_x como una resistencia fija de 1k Ω , por lo tanto, al realizar el despeje de la ecuación (11), resulta la ecuación (12).

$$R_y = R_x - \frac{R_x}{V_{in}/V_t} \quad \text{Ec. (12)}$$

Sustituyendo valores en la ecuación (12).

$$R_y = 1000\Omega - \left(\frac{1000\Omega}{5v/12.3v} \right) = 1460\Omega$$

Debido a que no existe un valor comercial de 1460Ω se eligió colocar en serie una resistencia de 1000Ω y una de 470Ω obteniendo un valor de 1470Ω que es cercano al valor requerido. Ya que se realizó la elección de los componentes se procede a realizar el programa en la plataforma Arduino. En la figura 4.46 se muestra la conexión física de los componentes. Es importante mencionar que en la imagen se usaron dos resistencias de 220Ω solo para realizar la prueba, para el ensamblaje final se utilizó una resistencia de 470Ω .

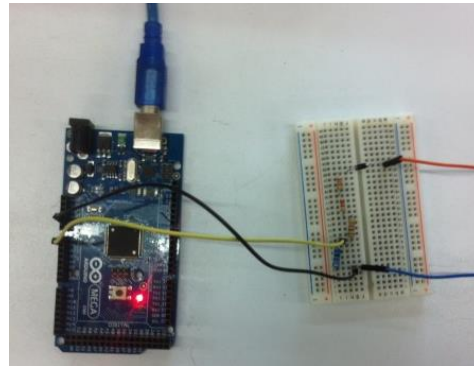


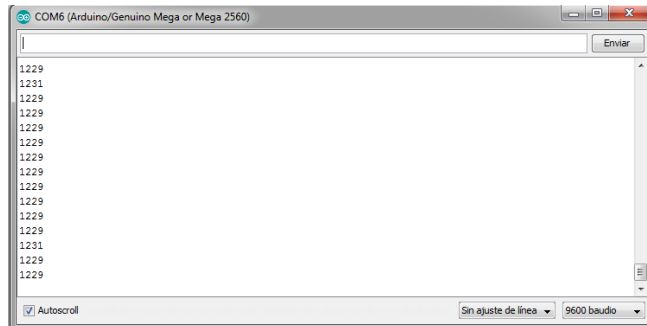
Figura 4.46. Conexión física del divisor de voltaje con el Arduino mega.

El código comienza declarando una variable entera (int) llamada `voltaje_bateria`, posteriormente se configura la transmisión para observar los valores en el monitor serial con el comando `Serial.begin(9600)`; una vez hecho esto se dará un valor a la variable, la cual corresponde a la siguiente línea de código:

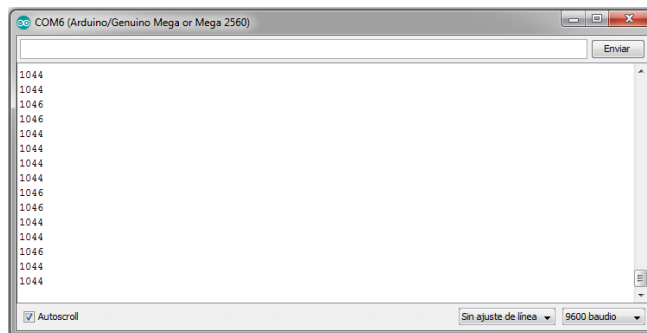
$$\text{Voltaje_bateria} = (\text{analogRead}(A0) + 70) * 1.2023$$

Donde `analogRead()` es un comando que permite leer datos de voltaje de 0 a 5V, desplegando estos valores en números de 0 a 1023. A0 es el pin analógico que se eligió para leer el voltaje. El número 70 corresponde a la compensación de la caída de voltaje en el diodo, este dato se obtiene midiendo la caída de voltaje en el diodo.

El valor 1.2023 corresponde a una constante que permite tener una lectura proporcional al voltaje de la batería sin la necesidad de hacer conversiones. Esta constante se obtiene de dividir 1230/1023, donde 1230 es proporcional al voltaje máximo de la batería 12.3 volts. En la figura 4.47 se muestran los datos obtenidos con el voltaje de la batería al máximo y el mínimo requerido para la batería, respectivamente.



a)



b)

Figura 4.47. Lectura de datos con el voltaje de la batería a) Voltaje máximo b) Voltaje mínimo.

Una vez obtenidos estos datos se requiere de un filtro para eliminar los ruidos que pueden llegar a existir en la etapa de pruebas. Este filtro se compone del 92% del voltaje de la batería más el 8% de la lectura del pin analógico de la tarjeta Arduino, por lo tanto, el código queda de la siguiente forma:

$$\text{voltaje_bateria} = \text{voltaje_bateria} * 0.92 + (\text{analogRead}(A0)+70) * 0.0962;$$

Donde 0.92 corresponde al 92% y 0.0962 representa el 8% de 1.2023.

Una vez que se obtiene una lectura correcta y sin ruido del voltaje de la batería se procede a realizar una alarma que, mediante el encendido de un led, constituirá una alerta visual indicando a la persona al mando del cuadricóptero que es momento de desconectar la batería y recargarla. Cuando el voltaje de la batería sea menor a 10.5V se encenderá el led y permanecerá encendido hasta que la batería sea cargada o hasta que el voltaje caiga por debajo de 6V. Para implementar esta alarma se utilizó una condición if que tiene la función de verificar el voltaje de la batería y si es menor a 1050 (datos que lee la tarjeta Arduino, equivalente a 10.5V) el led se activara.

El diagrama de conexión de la alarma de bajo voltaje en la batería se muestra en la figura 4.48.

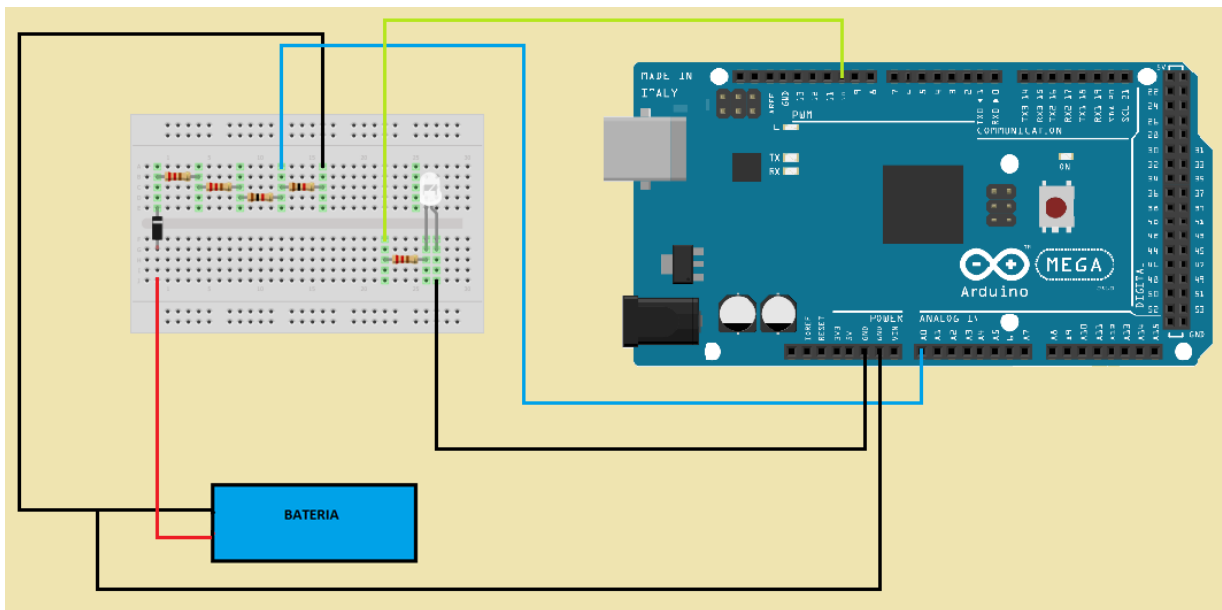


Figura 4.48. Diagrama de conexión para la alarma de bajo voltaje en la batería

Como se muestra en la figura 4.49, cuando en el monitor serial se tiene una lectura menor o igual a 1050 el led se encenderá.

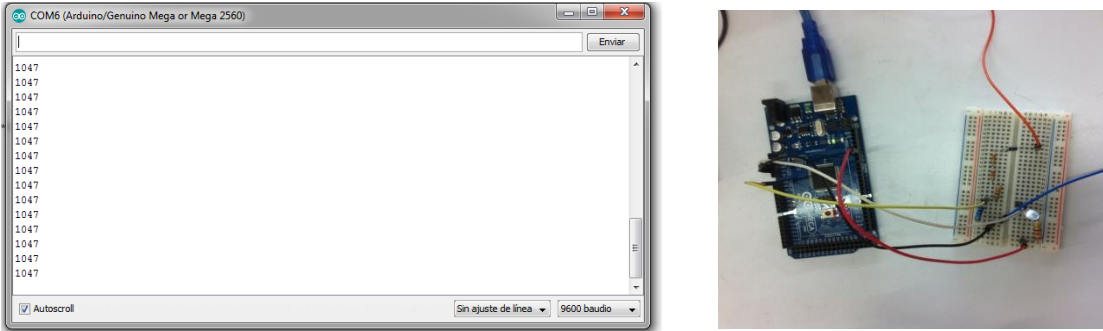


Figura 4.49. Prueba de la alarma de bajo voltaje en la batería.

Es importante realizar esta prueba debido a que si la batería no entrega el voltaje correspondiente a cada ESC estos pueden dañarse al igual que la batería.

Para realizar las conexiones de la alarma de bajo voltaje y de otros componentes como el MPU-6050 con la tarjeta Arduino se diseñó la tarjeta de circuito impreso (PCB) que se muestra en la figura 4.50.

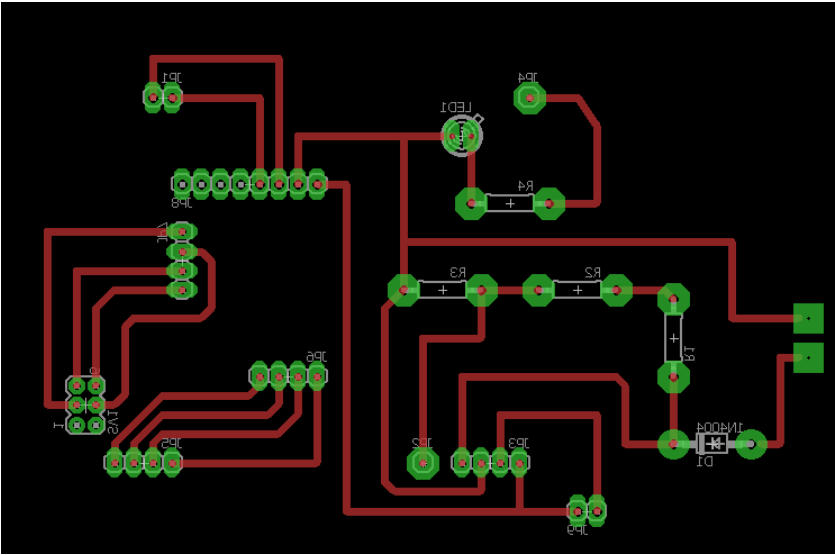


Figura 4.50. PCB para conexión de componentes con la tarjeta Arduino

4.1.9 Controlador proporcional-integral-derivativo (PID)

Los cuadricópteros son muy inestables por naturaleza, por ello se decidió implementar un control PID para lograr que el cuadricóptero se mantenga siempre nivelado y estable sin importar la habilidad de la persona encargada de su manejo.

Se implementó un controlador PID para los movimientos del motor roll y pitch. Debido a que la unidad de medición inercial empleada en este proyecto (IMU MPU 6050), no cuenta con un magnetómetro no es posible medir el ángulo en yaw, por lo tanto, este movimiento se omitirá en el PID. Sin embargo, esto no afectará la estabilidad del cuadricóptero y aun tendrá libertad de movimiento.

Para llevar a cabo el controlador PID es necesario recordar la ecuación que lo representa (Ec.6, Capítulo II). De acuerdo a esta ecuación para obtener la salida del controlador PID es necesario obtener cada una de las acciones de control (proporcional, integral y derivativa) y sumarlas. Estas acciones de control son el producto de la multiplicación del error, la integral del error y la derivada del error por las constantes correspondientes a cada acción de control, es decir, K_p , K_i y K_d respectivamente.

A continuación se ejemplificará lo anteriormente expuesto, mediante el desarrollo del controlador PID para el movimiento Roll:

Primeramente se calcula el error, este se obtiene al restar el set point para el movimiento roll a la variable de proceso de roll, que corresponde al valor obtenido mediante el giroscopio (grados por segundo).

$$\text{error_pid} = \text{imu_vp_roll} - \text{setpoint_roll};$$

Una vez hecho esto se procede a calcular el término integral, este término corresponde al producto de la ganancia del término integral roll por el error calculado anteriormente. En la expresión original del término integral se multiplica la ganancia por la integral del error, por lo tanto, se requiere encontrar una forma de integrar el error. Integrar el error sirve para minimizarlo hasta que este se vuelva cero, mientras

esto no ocurra la acción integral seguirá incrementándose; a su vez la integral se define como la generalización de la suma de infinitos sumandos.

Por lo tanto se infiere que para integrar el error es necesario sumar el término integral obtenido al mismo, es decir:

$$\text{término integral} = \text{término integral} + (\text{ganancia integral} * \text{error})$$

Esto queda establecido en la siguiente línea de código:

```
i_roll += ganancia_i_roll * error_pid;
```

Debido a que el término integral aumenta mientras el error no sea cero, elimina el error de estado estable que se produce con la ganancia proporcional. Este término tiende a tener sobre impulsos que alcanzan valores muy altos y oscilaciones, por ello es necesario colocar una condición que limite el valor máximo positivo que puede alcanzar el término integral y una que limite el valor máximo negativo. Ambas condiciones tienen la misma estructura y están descritas a continuación:

```
if (i_roll > salida_maxima_roll) i_roll = salida_maxima_roll;
```

```
else if(i_roll < (salida_maxima_roll * -1)) i_roll = salida_maxima_roll * -1;
```

Una vez hecho esto se procede a calcular la salida del PID para el movimiento roll. Como se especificó anteriormente, esto se lleva a cabo sumando los términos de cada uno de las acciones de control. Para el término proporcional se multiplica la constante proporcional por el error, el término integral se definió anteriormente por lo que solo se coloca en la suma.

El término derivativo, de manera similar al integral, está compuesto por la constante derivativa por la derivada del error. Para obtener la derivada del error, es necesario saber que una derivada constituye la razón de cambio de una variable a través del tiempo, por lo tanto se requiere conocer el cambio en el error; para esto basta con agregar una variable que almacene el valor del error una vez que este ocurra (error pasado) y restar esta variable al error actual, de manera que este valor se convertirá en el error pasado cuando se realice el cálculo nuevamente. Al conocer la razón de

cambio del error, el término derivativo puede “predecir” el comportamiento futuro del error, disminuir el sobreimpulso y las oscilaciones.

El código para la salida del controlador PID roll se muestra a continuación:

```
salida_pid_roll = (ganancia_p_roll * error_pid) + i_roll + (ganancia_d_roll * (error_pid  
– ultimo_error_roll));
```

Al igual que en el término integral, es necesario limitar el valor máximo (tanto positivo como negativo) que debe alcanzar la salida del PID, esto queda definido en las siguientes líneas de código:

```
if(salida_pid_roll > salida_maxima_roll) salida_pid_roll = salida_maxima_roll;  
  
else if(salida_pid_roll < (salida_maxima_roll * -1)) salida_pid_roll =  
salida_maxima_roll * -1;
```

Por último, se declara la variable que almacena el valor del error actual que se convertirá en el error pasado en el siguiente ciclo.

```
ultimo_error_roll = error_pid;
```

Este código se repite para el movimiento pitch, únicamente se debe cambiar las variables para que representen a su respectivo movimiento. Como se explicó anteriormente el movimiento yaw no se implementará en este proyecto, sin embargo si se quisiera implementar en un futuro el código también sería igual.

El código completo para calcular las salidas del controlador PID para cada uno de los movimientos se muestra en el apartado de anexos “6.4.5 Calculo De Salidas Para Controlador proporcional-integral-derivativo (PID)”.

4.1.10 Control del cuadricóptero en lazo cerrado

Para implementar el control del cuadricóptero en lazo cerrado es necesario conocer el funcionamiento básico del controlador PID y su interacción con los demás componentes del cuadricóptero, para ello se desarrolló la estructura del lazo cerrado de control tomando como referencia la propuesta por (Brokking), en 2015. En la figura 4.51 se observa gráficamente el proceso.

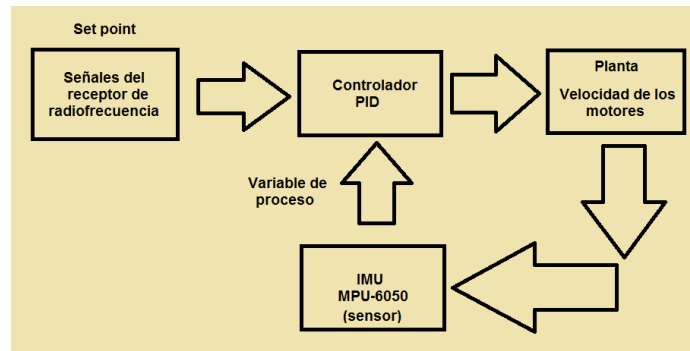


Figura 4.51. Diagrama de flujo del proceso de control

Como se observa en el diagrama de flujo de la figura 4.51 el proceso de control del cuadricóptero se divide en varias etapas, las partes sobre el procesamiento de los datos enviados por el sensor MPU-6050 y la obtención de las señales del receptor de radiofrecuencia ya han sido explicadas anteriormente. Estas etapas constituyen la variable de proceso (valor medido) y el set point (valor deseado) respectivamente, estas entradas servirán para que el PID pueda llevar a cabo su acción de control y realizar la corrección necesaria para que la variable de proceso alcance al valor deseado y se obtenga como resultado la salida (movimiento del cuadricóptero) deseada.

La etapa de la unidad de medición inercial (MPU-6050) no se modificó en lo absoluto, únicamente se eliminó la parte del código encargada de imprimir en el monitor serial los valores obtenidos. Las salidas producto de este programa (`imu_vp_roll`, `imu_vp_pitch` e `imu_vp_yaw`) constituyen el valor medido o real. El código para la alarma de bajo voltaje en la batería forma parte del lazo de control aunque no se

contemple en el diagrama de flujo y también va a permanecer sin modificaciones. A su vez la etapa para calcular la salida del controlador PID va a permanecer sin modificaciones, únicamente se procederá a agregar condiciones para inicializar y apagar los motores y establecer el valor del set point. Estas condiciones se describirán posteriormente.

- Señales del receptor de radiofrecuencia

Para esta etapa, se empleó el código para el encendido y control simultaneo de velocidad en cuatro motores, ya que este programa permite dar al PID el valor del set point o valor deseado calculando el ancho de pulso para la posición de las palancas de control del transmisor (Canal_receptor_1, Canal_receptor_2, Canal_receptor_3 y Canal_receptor_4) mediante interrupciones. Este valor corresponde a la señal del receptor de radiofrecuencia, esta señal será procesada por el controlador PID y emitirá una salida correctiva cuya función es enviar los pulsos adecuados a cada ESC para lograr el movimiento deseado del cuadricóptero.

Este programa ya contiene el código necesario para lograr que la frecuencia de actualización sea siempre igual a 250Hz. Sin embargo, este código solo enciende los cuatro motores a la misma velocidad simultáneamente, esto quiere decir que al variar la palanca del canal receptor 3 (arranque) los cuatro motores siempre incrementan o disminuyen su velocidad en la misma medida.

Esto es un problema, ya que para provocar un movimiento en el cuadricóptero se requiere que los motores giren a distintas velocidades para así provocar una cierta inclinación o giro, a su vez estas señales deben ser dependientes de los demás canales receptores y no solo del canal 3.

Por lo tanto se requiere modificar el código para lograr que las señales que se envíen a los ESC's sean independientes para cada uno de ellos, esto se ve reflejado en las siguientes líneas de código:

```
timer_canal_1 = esc_1 + timerX;
```

```
timer_canal_2 = esc_2 + timerX;
```

```
timer_canal_3 = esc_3 + timerX;
```

```
timer_canal_4 = esc_4 + timerX;
```

El código para obtener las variables `esc_1`, `esc_2`, `esc_3` y `esc_4` se explicará más adelante.

- Cálculo de los pulsos enviados a cada ESC, según el movimiento del cuadricóptero

Es importante una vez terminado el control calcular el pulso que cada ESC debe tener según el movimiento que el cuadricóptero vaya a llevar a cabo, para esto se requiere realizar una serie de sumas y restas, cuyo significado se refiere a la configuración del IMU (como se muestra en la figura 4.52) y no a la operación matemática. Estas marcas indican que hacia donde apunta la flecha curvada los datos serán enviado con signo positivo de lo contrario el signo será negativo.

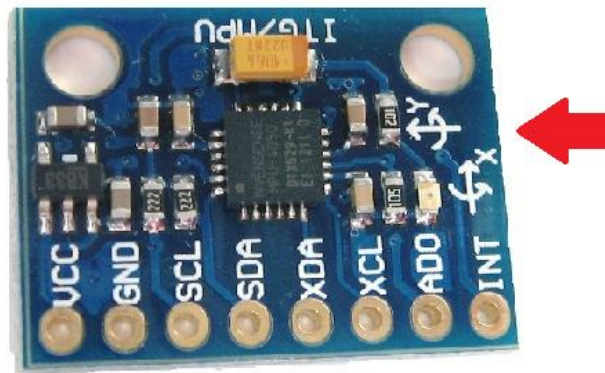


Figura 4.52. Configuración de los ejes del MPU-6050 (Arduino)

Por lo tanto es importante que las direcciones coincidan con las flechas color negro que se muestran en la figura 4.53 ya que de no ser así el control puede ser inverso.

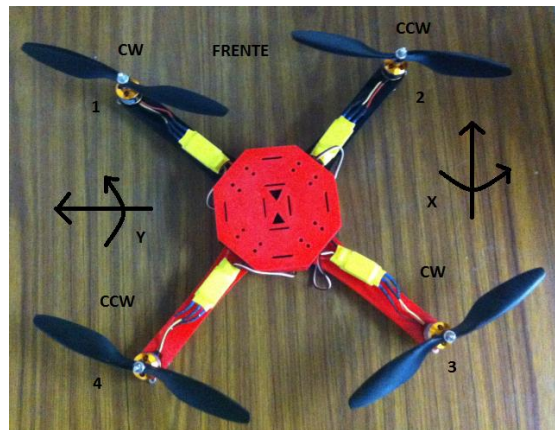


Figura 4.53. Configuración de motores y ejes de movimiento.

Una vez que se asignó el giro de los motores y el movimiento de los ejes es momento de realizar el cálculo de los ESC, para ello se comienza con la variable `ch3`, que es proporcionada por el canal receptor 3, como se aprecia a continuación:

$$\text{ch3} = \text{Canal_receptor_3}$$

Esta variable representa la velocidad de los motores, por lo tanto, haciendo uso de la figura 4.53 se aprecia que si la IMU se inclina hacia el frente resulta un signo positivo en la salida del PID correspondiente al movimiento pitch y uno negativo si este se inclina hacia atrás. Esto quiere decir que si se desea que el cuadricóptero avance hacia al frente el IMU leerá valores positivos que serán enviados a los ESC's 1 y 2 para realizar la acción y por ende la acción inversa implicara datos con signo negativos para los ESC's 3 y 4 por lo que el código quedara de la siguiente forma:

$$\text{esc_1} = \text{ch3} + \text{salida_pid_pitch}$$

$$\text{esc_2} = \text{ch3} + \text{salida_pid_pitch}$$

$$\text{esc_3} = \text{ch3} - \text{salida_pid_pitch}$$

$$\text{esc_4} = \text{ch3} - \text{salida_pid_pitch}$$

Ahora se agrega al cálculo la salida del PID correspondiente al movimiento roll, esta asignación se realiza igual que para el pitch, si la IMU se gira sobre la derecha los datos serán positivos y serán enviados a los ESC's 2 y 3 por lo tanto hacia la izquierda serán negativos que corresponden a los ESC's 1 y 4. Por lo tanto el código quedara de la siguiente manera:

$$\text{esc_1} = \text{ch3} + \text{salida_pid_pitch} - \text{salida_pid_roll}$$

$$\text{esc_2} = \text{ch3} + \text{salida_pid_pitch} + \text{salida_pid_roll}$$

$$\text{esc_3} = \text{ch3} - \text{salida_pid_pitch} + \text{salida_pid_roll}$$

$$\text{esc_4} = \text{ch3} - \text{salida_pid_pitch} - \text{salida_pid_roll}$$

Por último se asignaran dos límites para mantener un control estable; el inferior será de 1100µs, esto es para lograr que cuando los motores se inicialicen se mantengan a cierta velocidad y no se apaguen aunque se baje la palanca de arranque a su posición mínima, de esta forma cuando el pulso sea menor a 1100µs se le asigna este valor.

El límite superior es de 2000µs, esto para no perder estabilidad en el cuadricóptero. Para poder realizar esta acción se hará uso de una condición "if", la cual quedara de la siguiente forma:

$$\text{if} (\text{esc_1} < 1100) \text{esc_1} = 1100$$

Como se aprecia anteriormente si el esc_1 tiene un valor menor a 1100µs este tomara el valor de 1100µs para asignarle una cierta velocidad, esta condición se realizara para cada uno de los ESC's. Por lo tanto para el límite superior la condición será que si el esc_1 es mayor a 2000µs este tome el valor de 2000µs como se aprecia a continuación:

$$\text{If} (\text{esc_1} > 2000) \text{esc_1} = 2000$$

En el apartado “6.4.6 Cálculo De Los Pulsos Enviados A Cada ESC, Según El Movimiento Del Cuadricóptero” de anexos se encuentra el código descrito anteriormente.

- Condiciones para inicializar y detener los motores

Cuando comienza a correr el programa del lazo cerrado de control se requiere reiniciar los valores del controlador PID y a su vez agregar una condición para el encendido y el apagado de los motores. Los motores se encenderán al bajar la palanca de arranque a su posición mínima (se enciende el transmisor con la palanca de arranque en posición máxima, yaw en posición media) y la variable encender que tenía un valor cero al inicio del programa adquiere el valor de uno (encender=1).

- Para reiniciar los valores del controlador PID se establecen los siguientes valores a cero, esto se realiza de la misma forma para el movimiento pitch:

```
i_roll = 0;
```

```
ultimo_error_roll = 0;
```

- Para detener los motores se requiere una condición, ya que se colocó una condición para evitar que los motores se apaguen aun cuando la palanca de arranque se encuentre en su posición más baja. Por lo tanto, para llevar a cabo su apagado es necesario que la palanca de arranque este en su posición más baja y además de esto que la palanca correspondiente a yaw este en el extremo derecho, es decir en aproximadamente 2000µs.

```
if(encender==1 && Canal_receptor_3 < 1200 && Canal_receptor_4 > 1750)encender=0;
```

A su vez se agregó la siguiente instrucción a ejecutar cuando la variable encender sea diferente a uno, de esta manera se lleva a cabo el apagado de los motores:

```
else{
```

```
    esc_1 = 1000; esc_2 = 1000; esc_3 = 1000; esc_4 = 1000; }
```

El valor deseado o set point de los controladores PID que corresponden a cada uno de los movimientos del cuadricóptero roll, pitch y yaw debe ser establecido de acuerdo al valor enviado por el piloto mediante el transmisor de radiofrecuencia.

Por ello mediante el siguiente código se establece el valor del set point roll en función a la posición de la palanca correspondiente a este movimiento, es decir, Canal 1. Esta referencia debe ser dada en unidad de grados por segundo (dps) ya que es la unidad en la que se obtienen los datos provenientes de la IMU, que corresponden a la variable de proceso.

Se decidió establecer una banda muerta de $20\mu\text{s}$ ($\pm 10\mu\text{s}$ del valor original) para asegurar que las variaciones que se puedan presentar en la entrada no ocasionen una respuesta errónea en el PID.

Posteriormente se establece la variable correspondiente al set point roll a cero.

```
setpoint_roll = 0
```

En estas condiciones se establece la banda muerta antes mencionada de $\pm 10\mu\text{s}$ respecto al valor medio del pulso ($1500\mu\text{s}$).

```
if(Canal_receptor_1 > 1510) setpoint_roll = (Canal_receptor_1 - 1510);
```

```
else if(Canal_receptor_1 < 1490) setpoint_roll = (Canal_receptor_1 - 1490);
```

El procedimiento anterior se lleva a cabo de la misma manera para los dos movimientos restantes del cuadricóptero con la excepción de que al movimiento pitch corresponde el canal receptor 2 y al movimiento yaw corresponde el canal receptor 3.

El código del lazo cerrado de control completo se encuentra en el apartado de anexos “6.4.7 Lazo Cerrado De Control” y como se explicó anteriormente solo se llevaron a cabo los controladores correspondientes a los movimientos roll y pitch.

En la figura 4.54 se muestra el diagrama de conexión del cuadricóptero en lazo cerrado.

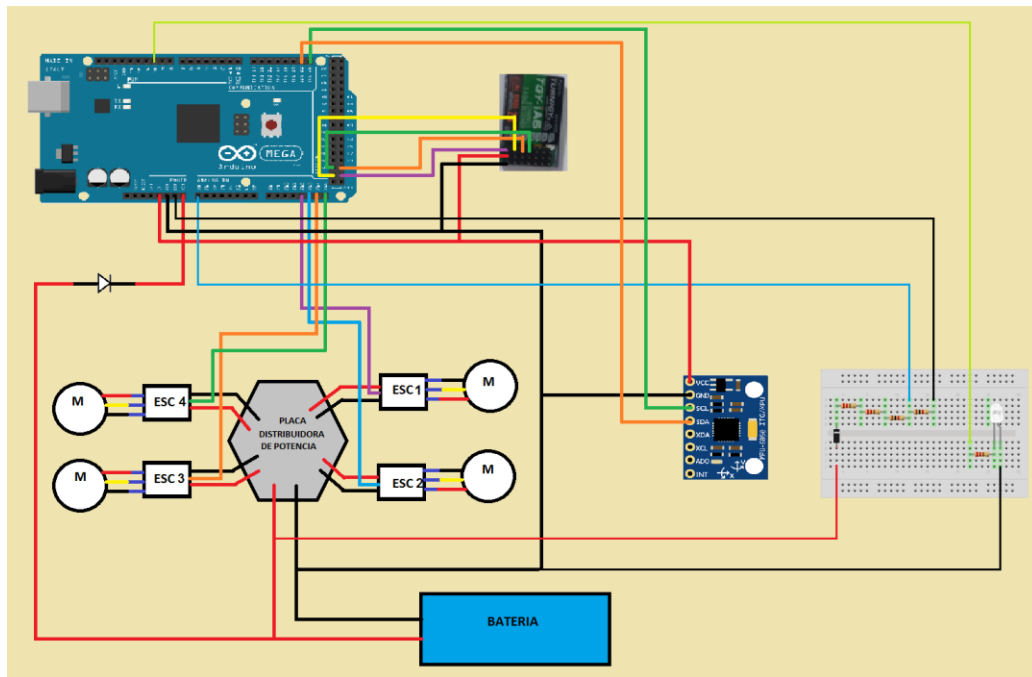


Figura 4.54. Diagrama final de conexión del cuadricóptero en lazo cerrado

En la figura 4.55 se muestra el ensamblaje completo del cuadricóptero en lazo cerrado y su conexión física.



Figura 4.55. Ensamblaje completo del cuadricóptero en lazo cerrado y conexión física

4.2 ACTIVIDADES ADICIONALES

De manera adicional se implementó un simulador de vuelo giroscópico, con el propósito de mantener al cuadricóptero a salvo de caídas y posibles daños al momento de realizar las pruebas. De esta manera los alumnos podrán realizar pruebas al controlador del cuadricóptero variando sus constantes y observar los efectos que esto tiene en su estabilidad sin el riesgo de dañar alguno de los componentes del cuadricóptero.

El simulador de vuelo consiste en un giroscopio; debido a esto el cuadricóptero tiene libertad de movimiento en todos sus ejes ya que al ejercer una fuerza sobre el giroscopio este gira sobre el eje determinado permitiendo que el cuadricóptero realice todos sus movimientos sin ninguna oposición y en condiciones similares a las que presentaría si se encontrara en el espacio libre, pero con la ventaja de que no se dañara si su controlador no es calibrado adecuadamente dando a los alumnos confianza para experimentar y llevar a cabo un análisis concreto sobre el controlador. En la figura 4.56 se muestra el simulador de vuelo giroscópico implementado para este fin.



Figura 4.56. Simulador de vuelo giroscópico

CAPÍTULO V. RESULTADOS OBTENIDOS

5.1 RESULTADOS DEL CONTROLADOR PID

Una vez implementado el código desarrollado anteriormente se obtiene un sistema de control en lazo cerrado al que se deben realizar determinadas pruebas para observar su funcionamiento y en base a esto proceder al ajuste de cada uno de los parámetros del controlador (proporcional, integral y derivativo) hasta obtener el resultado deseado.

Para realizar la calibración del controlador PID se optó por emplear el método heurístico también conocido como de prueba y error, ya que es el método que más se ajusta a las necesidades de este proyecto, esto se debe a que no se conocen las funciones de transferencia de los motores y su obtención está fuera de los alcances de este proyecto. Cabe destacar que en este proyecto no se pretende realizar la sintonización del PID, solo su calibración.

Las pruebas fueron realizadas empleando una fuente de poder en lugar de la batería, ya que esta se descarga constantemente y no sería práctico detener las pruebas para cargar la batería, de esta manera la fuente brinda la posibilidad de realizar las pruebas extensas de manera continua. Esta fuente de poder tiene especificaciones de 5V a 30A mientras que el voltaje requerido para hacer funcionar al cuadricóptero es de aproximadamente 11.1V, por lo tanto fue necesario colocar dos fuentes (con las características descritas anteriormente) en serie para de esta manera obtener una fuente de 10V a 30A.

Con lo anterior se obtiene una fuente con la suficiente potencia para realizar pruebas de funcionamiento al cuadricóptero en espacios cerrados, esto es importante ya que este proyecto tiene como finalidad servir a los alumnos para realizar pruebas y prácticas de control dentro de los laboratorios, por ello esto es una opción para no provocar un desgaste acelerado de la batería y reservarla para el vuelo en exteriores.

De acuerdo al método heurístico la calibración del controlador PID se lleva a cabo siguiendo los siguientes pasos:

- Establecer solamente la acción proporcional del controlador a un valor cercano a 1, mientras que las acciones integral y derivativa deben ser nulas.
- Incrementar la ganancia proporcional del controlador hasta lograr oscilaciones estables.
- Incrementar la acción integral hasta lograr el efecto deseado, eliminando el error de estado estacionario.
- Ajustar la acción derivativa en forma iterativa hasta lograr la acción deseada, si es necesario pueden modificarse las constantes proporcional e integral encontradas anteriormente.

A continuación se llevan a cabo los pasos realizados descritos anteriormente para calibrar el controlador PID, se colocó peso en un extremo de la base buscando con esto tener una capacidad de observación mayor para verificar que el controlador realice correcciones para mantenerse nivelado en vuelo, esto con el propósito de crear una perturbación y comprobar que el cuadricóptero se mantendrá nivelado a pesar de las turbulencias y de la habilidad del conductor del cuadricóptero. En la figura 5.1 se muestra el cuadricóptero y el peso colocado en la base.



Figura 5.1. Constantes del controlador PID, primera prueba

- La primera prueba que se realizó fue colocando la ganancia proporcional en 1 y las ganancias integral y derivativa en cero. Como se muestra en la figura 5.2.

```
//Movimiento Roll

float i_roll, setpoint_roll, salida_pid_roll, ultimo_error_roll;
float ganancia_p_roll = 1 ; //Ganancia para el control proporcional del movimiento roll
float ganancia_i_roll = 0 ; //Ganancia para el control integral del movimiento roll
float ganancia_d_roll = 0 ; //Ganancia para el control derivativo del movimiento roll
int salida_maxima_roll = 300; //Limita la salida máxima del pid roll

//Movimiento Pitch

float i_pitch, setpoint_pitch, salida_pid_pitch, ultimo_error_pitch;
float ganancia_p_pitch = 1 ; //Ganancia para el control proporcional del movimiento pitch
float ganancia_i_pitch = 0 ; //Ganancia para el control integral del movimiento pitch
float ganancia_d_pitch = 0 ; //Ganancia para el control derivativo del movimiento pitch
int salida_maxima_pitch = 300; //Limita la salida máxima del pid pitch
```

Figura 5.2. Constantes del controlador PID, primera prueba

Al realizar esta prueba no se observó corrección alguna por lo que el cuadricóptero no se niveló.

- Por ello se realizaron varias pruebas más incrementando la ganancia proporcional, se encontró que el sistema entraba en oscilación con la constante proporcional igual a 8, como se muestra en la figura 5.3.

```
//Movimiento Roll

float i_roll, setpoint_roll, salida_pid_roll, ultimo_error_roll;
float ganancia_p_roll = 8 ; //Ganancia para el control proporcional del movimiento roll
float ganancia_i_roll = 0 ; //Ganancia para el control integral del movimiento roll
float ganancia_d_roll = 0 ; //Ganancia para el control derivativo del movimiento roll
int salida_maxima_roll = 300; //Limita la salida máxima del pid roll

//Movimiento Pitch

float i_pitch, setpoint_pitch, salida_pid_pitch, ultimo_error_pitch;
float ganancia_p_pitch = 8 ; //Ganancia para el control proporcional del movimiento pitch
float ganancia_i_pitch = 0 ; //Ganancia para el control integral del movimiento pitch
float ganancia_d_pitch = 0 ; //Ganancia para el control derivativo del movimiento pitch
int salida_maxima_pitch = 300; //Limita la salida máxima del pid pitch
```

Figura 5.3. Constantes del controlador PID, segunda prueba

El sistema se aproximó bastante al set point, sin embargo tuvo un sobreimpulso y la respuesta no se mantuvo estable, por lo que es necesario eliminar el error de estado estable por medio de la adición de la acción integral. Como se dijo anteriormente el sistema entro en oscilación, por lo que se considera un sistema inestable, así mismo es necesario disminuir la magnitud del sobreimpulso ya que puede ocasionar daños a los demás componentes del cuadricóptero.

- La siguiente prueba a realizar consiste en agregar la constante integral, esta constante comúnmente tiene valores de entre 0 y 0.1, por lo que se decidió comenzar por valores pequeños, más específicamente 0.03 como se muestra en la figura 5.4.

```
//Movimiento Roll
float i_roll, setpoint_roll, salida_pid_roll, ultimo_error_roll;
float ganancia_p_roll = 8 ; //Ganancia para el control proporcional del movimiento roll
float ganancia_i_roll = 0.03 ; //Ganancia para el control integral del movimiento roll
float ganancia_d_roll = 0 ; //Ganancia para el control deriativo del movimiento roll
int salida_maxima_roll = 300; //Limita la salida máxima del pid roll

//Movimiento Pitch
float i_pitch, setpoint_pitch, salida_pid_pitch, ultimo_error_pitch;
float ganancia_p_pitch = 8 ; //Ganancia para el control proporcional del movimiento pitch
float ganancia_i_pitch = 0.03 ; //Ganancia para el control integral del movimiento pitch
float ganancia_d_pitch = 0 ; //Ganancia para el control deriativo del movimiento pitch
int salida_maxima_pitch = 300; //Limita la salida máxima del pid pitch
```

Figura 5.4. Constantes del controlador PID, tercera prueba

Con esta prueba el error de estado estable del sistema disminuyo, sin embargo no se eliminó del todo por lo que se decidió incrementar más el valor de la constante integral.

- Con una constante integral de 0.06 (figura 5.5) el cuadricóptero realizó la corrección adecuadamente y disminuyó el error de estado estable, sin embargo presento un sobreimpulso aun mayor y el tiempo de estabilización también incremento, por lo tanto se bajó un poco la constante integral y se agregó la constante derivativa buscando corregir estos inconvenientes.

```
//Movimiento Roll

float i_roll, setpoint_roll, salida_pid_roll, ultimo_error_roll;
float ganancia_p_roll = 8 ; //Ganancia para el control proporcional del movimiento roll
float ganancia_i_roll = 0.06 ; //Ganancia para el control integral del movimiento roll
float ganancia_d_roll = 0 ; //Ganancia para el control derivativo del movimiento roll
int salida_maxima_roll = 300; //Limita la salida máxima del pid roll

//Movimiento Pitch

float i_pitch, setpoint_pitch, salida_pid_pitch, ultimo_error_pitch;
float ganancia_p_pitch = 8 ; //Ganancia para el control proporcional del movimiento pitch
float ganancia_i_pitch = 0.06 ; //Ganancia para el control integral del movimiento pitch
float ganancia_d_pitch = 0 ; //Ganancia para el control derivativo del movimiento pitch
int salida_maxima_pitch = 300; //Limita la salida máxima del pid pitch
```

Figura 5.5. Constantes del controlador PID, cuarta prueba

- Para la siguiente prueba, como se dijo anteriormente, se optó por disminuir la constante integral a 0.04 (figura 5.6) esperando con esto disminuir el sobreimpulso y el tiempo de estabilización pero conservando un error de estado estable casi nulo. Además se agregó la constante derivativa para provocar la reducción del sobreimpulso y una aceleración del tiempo de estabilización y de respuesta.

```
//Movimiento Roll

float i_roll, setpoint_roll, salida_pid_roll, ultimo_error_roll;
float ganancia_p_roll = 8 ; //Ganancia para el control proporcional del movimiento roll
float ganancia_i_roll = 0.04 ; //Ganancia para el control integral del movimiento roll
float ganancia_d_roll = 10 ; //Ganancia para el control derivativo del movimiento roll
int salida_maxima_roll = 300; //Limita la salida máxima del pid roll

//Movimiento Pitch

float i_pitch, setpoint_pitch, salida_pid_pitch, ultimo_error_pitch;
float ganancia_p_pitch = 8 ; //Ganancia para el control proporcional del movimiento pitch
float ganancia_i_pitch = 0.04 ; //Ganancia para el control integral del movimiento pitch
float ganancia_d_pitch = 10 ; //Ganancia para el control derivativo del movimiento pitch
int salida_maxima_pitch = 300; //Limita la salida máxima del pid pitch
```

Figura 5.6. Constantes del controlador PID, quinta prueba

Se obtuvo una respuesta estable, más rápida y con menor sobreimpulso, sin embargo se desea mejorar aún más la velocidad de estabilización del sistema, por ello se incrementará la acción derivativa lo suficiente para incrementar la estabilidad pero con cautela para evitar efectos indeseables en la respuesta del sistema. Esto debido a que la acción derivativa en cantidades muy elevadas puede llegar a provocar inestabilidad en el sistema ya que es altamente sensible al ruido. A su vez para disminuir el sobreimpulso sin incrementar demasiado la acción derivativa se disminuirá la acción proporcional.

- Se estableció la acción proporcional a 6 y se incrementó la acción derivativa a 15, como se muestra en la figura 5.7. Esto provocó una disminución en el tiempo de estabilización y se eliminó casi por completo el sobreimpulso.

Sin embargo, se desea eliminar por completo el error de estado estable e incrementar aún más la rapidez del sistema.

```
//Movimiento Roll

float i_roll, setpoint_roll, salida_pid_roll, ultimo_error_roll;
float ganancia_p_roll = 6 ; //Ganancia para el control proporcional del movimiento roll
float ganancia_i_roll = 0.04 ; //Ganancia para el control integral del movimiento roll
float ganancia_d_roll = 15 ; //Ganancia para el control deriativo del movimiento roll
int salida_maxima_roll = 300; //Limita la salida máxima del pid roll

//Movimiento Pitch

float i_pitch, setpoint_pitch, salida_pid_pitch, ultimo_error_pitch;
float ganancia_p_pitch = 6 ; //Ganancia para el control proporcional del movimiento pitch
float ganancia_i_pitch = 0.04 ; //Ganancia para el control integral del movimiento pitch
float ganancia_d_pitch = 15 ; //Ganancia para el control deriativo del movimiento pitch
int salida_maxima_pitch = 300; //Limita la salida máxima del pid pitch
```

Figura 5.7. Constantes del controlador PID, sexta prueba

En la figura 5.8 se aprecia que la corrección realizada por el controlador es buena, sin embargo aún existe el error de estado estable.



Figura 5.8. Estabilización del cuadricóptero con error

Para lograr eliminar el error de estado estable se procederá a incrementar gradualmente la constante integral, mientras que para lograr incrementar la rapidez del sistema y eliminar por completo el sobreimpulso se incrementará la acción derivativa iterativamente verificando siempre que no haya consecuencias indeseables en el sistema, como inestabilidad.

- Como se muestra en la figura 5.9 las constantes que lograron el óptimo desempeño y control del sistema con acción proporcional=6, acción integral=0.08 y acción derivativa=18. Esto logro disminuir el tiempo de estabilización y se eliminó el sobreimpulso.

```
//Movimiento Roll
float i_roll, setpoint_roll, salida_pid_roll, ultimo_error_roll;
float ganancia_p_roll = 6 ; //Ganancia para el control proporcional del movimiento roll
float ganancia_i_roll = 0.08 ; //Ganancia para el control integral del movimiento roll
float ganancia_d_roll = 18 ; //Ganancia para el control deriatiivo del movimiento roll
int salida_maxima_roll = 300; //Limita la salida máxima del pid roll

//Movimiento Pitch
float i_pitch, setpoint_pitch, salida_pid_pitch, ultimo_error_pitch;
float ganancia_p_pitch = 6 ; //Ganancia para el control proporcional del movimiento pitch
float ganancia_i_pitch = 0.08 ; //Ganancia para el control integral del movimiento pitch
float ganancia_d_pitch = 18 ; //Ganancia para el control deriatiivo del movimiento pitch
int salida_maxima_pitch = 300; //Limita la salida máxima del pid pitch
```

Figura 5.9. Constantes del controlador PID, séptima prueba

De esta manera se logró que el cuadricóptero se estabilice y aplique una acción correctiva ante las perturbaciones, que en este caso fueron simuladas mediante la adición de peso en un extremo del simulador giroscópico de vuelo. Como se muestra en la figura 5.10 el controlador PID realiza una acción correctiva en proporción a la desviación de su punto estable, que fue establecido mediante programación, por lo tanto al detectar el error producido por la desviación de la variable de proceso provocada por el peso en un extremo el controlador realizó la corrección necesaria para mantener al cuadricóptero nivelado.



Figura 5.10. Cuadricóptero nivelado aun con perturbaciones

CAPÍTULO VI. CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES

Como resultado de la realización de este proyecto puede concluirse que para llevar a cabo la implementación de un cuadricóptero es necesario comprender a profundidad cada una de las diferentes etapas de las que está compuesto. Primeramente, es necesario conocer su principio de operación, es decir, que movimientos realiza y como los lleva a cabo. De acuerdo a (Álvarez Salgado & Rodríguez Lira , 2015) un cuadricóptero puede ser maniobrado mediante la combinación de sus tres movimientos: Yaw, Pitch y Roll. A su vez, cabe destacar que la elección de los materiales que componen al cuadricóptero se realiza considerando la aplicación que se dará al cuadricóptero y la relación precio-calidad, en base a ello se consideran factores como:

- Requerimientos de los motores, es decir, si es que se requiere un par elevado o una velocidad alta.
- Capacidades de corriente, esto debido a que los ESC al igual que la batería deben elegirse tomando en cuenta que su salida pueda suministrar suficiente corriente a los motores.
- Recomendaciones del fabricante de los motores, respecto al número de celdas de la batería y tipo de batería. Así como respecto al tamaño de las hélices ya que el fabricante brinda varias opciones para elegir de acuerdo al empuje que se desea para el cuadricóptero.
- Tiempo de vuelo que se desea para el cuadricóptero, ya que las baterías de mayor capacidad son también más pesadas, resultando ineficientes, por ello debe conservarse balance entre capacidad-peso.
- La aplicación, esto se refiere a que el receptor del sistema de comunicación mediante radiofrecuencia debe contar como mínimo con cuatro canales para controlar el movimiento de un cuadricóptero.

- Movimientos a controlar, ya que el sensor dinámico debe ser elegido de mínimo 6 grados de libertad (dof), ya que de esta manera se pueden controlar eficientemente al cuadricóptero mediante dos de sus movimientos y presenta una mejor relación calidad-precio. Sin embargo si se requiere controlar los tres movimientos del cuadricóptero, los sensores de 6 dof brindan la posibilidad de agregar un magnetómetro para realizar el control de los tres movimientos, o existe la posibilidad de elegir un sensor de 9 dof.
- Versatilidad, la controladora de vuelo se seleccionó tomando en cuenta la posible adición de más funciones al sistema de control del cuadricóptero, por ello se eligió una controladora con la capacidad de soportar la adición de nuevas funciones.

El cuadricóptero se compone de un sistema mecánico, que corresponde al frame o armazón y las hélices. El frame se diseñó tomando en cuenta el aerodinamismo y eficiencia de diferentes estructuras para cuadricópteros y eligiendo la más adecuada para esta aplicación, su construcción se llevó a cabo mediante una impresora 3D en material tipo PLA debido a la ligereza del material, es importante considerar la posición de los materiales que componen al cuadricóptero de manera adecuada previamente al diseño del frame.

A su vez cuenta con un sistema electrónico que consta de los motores brushless, los variadores de velocidad (ESC), la batería o fuente de alimentación, los sensores dinámicos que en este caso particular el sensor empleado fue una IMU MPU-6050 que interiormente se compone de un acelerómetro y un giroscopio por lo que brinda 6 grados de libertad, esto significó que el control del cuadricóptero quedo limitado al movimiento roll y pitch ya que para controlar los tres movimientos es necesaria un sensor de 9 grados de libertad.

Un sensor dinámico brinda la posibilidad de implementar un sistema de control en lazo cerrado debido a que los valores enviados por este representan la variable de proceso, estos datos en tiempo real correspondientes a los grados por segundo (dps) que censa la IMU en sus ejes retroalimentan al sistema, tales valores son recibidos

por la tarjeta Arduino Mega mediante el protocolo de comunicación I²C para posteriormente ser procesados por un controlador PID y realizar una corrección respecto al error resultante entre la variable de proceso y el setpoint. Los valores provenientes de la IMU requieren de un procesamiento previo a su uso, para ello se requiere emplear un filtro complementario cuya salida provea datos confiables de la posición del cuadricóptero al controlador.

El sistema de control del cuadricóptero consta de una controladora de vuelo, existen controladoras de vuelo comerciales con algoritmos de control precargados; sin embargo, ya que este proyecto tiene fines educativos y su finalidad es que los alumnos implementen sus propios algoritmos de control y los prueben mediante el cuadricóptero se optó por emplear una tarjeta Arduino Mega 2560 y por medio de esta plataforma implementar un controlador PID para lograr el control y la estabilización del cuadricóptero ante las perturbaciones. Los controladores PID son muy eficientes y ampliamente implementados debido a su capacidad y eficiencia para controlar todo tipo de procesos mediante la combinación de sus tres acciones de control: proporcional, integral y derivativa.

Como parte del controlador PID, se requiere la obtención de un error en base al cual realizar correcciones, este error como se dijo anteriormente resulta de la diferencia entre la variable de proceso y el set point, por ello se requiere establecer el set point de alguna manera. El set point del cuadricóptero es establecido mediante el sistema de comunicación mediante radiofrecuencia, de esta manera es posible que el cuadricóptero reciba instrucciones de movimiento desde un mando a distancia o transmisor. Las señales captadas por el receptor son procesadas para ser convertidas en un set point y de esta manera, mediante el controlador PID, lograr que el cuadricóptero realice correcciones y alcance la posición deseada.

Para lograr que el cuadricóptero realice movimientos a partir de las acciones correctivas del controlador PID es necesario establecer ciertas condiciones que definirán la magnitud del ancho de pulso enviado a cada uno de los variadores de velocidad, estas diferencias entre las velocidades de cada motor provocarán que el cuadricóptero tienda a moverse hacia al frente, hacia atrás, hacia la derecha o hacia

la izquierda. Por ello, es de suma importancia prestar atención al orden de los variadores de velocidad, a la posición de la IMU y al sentido de giro de los motores ya que de estos factores depende el lograr que el cuadricóptero realice los movimientos y correcciones correctas.

A cada uno de los materiales elegidos se le realizaron pruebas individuales para corroborar su correcto funcionamiento antes de su proceder a su interconexión con los demás componentes, esto hace más fácil la detección de errores ya que se conoce previamente que partes funcionan adecuadamente. De las pruebas individuales se procedió a la interconexión de algunos de los componentes dividiendo en etapas el proceso para obtener el sistema de control en lazo cerrado.

Algunas de estas etapas son las pruebas al sistema de comunicación mediante radiofrecuencia mediante lo cual se observaron los datos obtenidos por el receptor, al conocer estos datos es posible analizar la manera de procesarlos posteriormente. La siguiente etapa consistió en calibrar la IMU y observar que los ángulos mostrados sean correctos tanto en signo como en magnitud. En la siguiente etapa se procedió al encendido simultaneo de los cuatro motores del cuadricóptero, a su vez también se implementó una alarma indicadora de bajo voltaje en la batería para evitar posibles daños a esta. Posteriormente se llevaron a cabo los cálculos para el controlador PID, se requirió de un controlador para el movimiento roll y otro controlador para el movimiento pitch.

Finalmente se unieron todas las etapas anteriores con ciertas modificaciones y adiciones al código para lograr que las correcciones y salidas del controlador PID se transformen en movimientos del cuadricóptero. Por último se realizaron pruebas para calibrar el sistema de control en lazo cerrado, tomando en cuenta para su análisis que el cuadricóptero fuera capaz de estabilizarse ante perturbaciones y de responder adecuadamente a los cambios en el set point establecido mediante el transmisor. Para realizar estas pruebas se implementó un simulador giroscópico de vuelo con el fin de brindar soporte al cuadricóptero y así llevar a cabo las pruebas al controlador sin provocar daño alguno a ninguno de sus componentes.

6.2 RECOMENDACIONES

Una de las sugerencias que harán respecto al proyecto es la utilización de diferentes estructuras de frame para el cuadricóptero ya que pueden realizarse pruebas para optimizar el vuelo del mismo, es importante mencionar que si se desea cubrir los motores debe tomarse en cuenta que estos no son como los motores de corriente directa que se conocen normalmente y si no se deja un espacio adecuado para el libre giro del mismo este podría presentar problemas.

También se sugiere antes de diseñar el frame pensar en cómo es que se ajustaran los motores a este ya que los tornillos que lo sujetaran son muy importantes para minimizar vibraciones. Otra sugerencia muy importante es la utilización de una IMU de 9 dof agregar un magnetómetro a la IMU empleada ya que así se podrá realizar un control PID en el movimiento yaw.

Otra de las recomendaciones que surgieron durante la parte de pruebas es que la duración de la batería es muy pequeña por lo que el tiempo para realizar las pruebas no era el suficiente para obtener resultados confiables así que se decidió utilizar una fuente sacrificando potencia por duración, por lo tanto si se desea utilizar una batería se tendrán que corregir las constantes del PID realizando pruebas en la plataforma hasta obtener una estabilización óptima. Si se desea hacer un cuadricóptero para pruebas al aire libre se recomienda utilizar una tarjeta Arduino mini o uno ya que tendrá más espacio y un peso menor lo que proporcionara mayor potencia.

Cuando se realicen las pruebas en la plataforma los motores deben de girar entre su capacidad media y mínima ya que de no ser así se causaran daños en el frame o puede provocar que algunas de las hélices o en el peor de los casos que un motor se desprenda del mismo causando daños irreparables en los dispositivos. Por último se sugiere que antes de realizar el frame y de comprar los materiales se piense para que se desea utilizar el cuadricóptero y que utilidad va a tener, unas vez pensadas y analizadas cuidadosamente puede procederse a comprar los materiales teniendo en cuenta que la duración de entrega de los materiales puede variar según el proveedor.

6.3 REFERENCIAS BIBLIOGRÁFICAS

Álvarez Salgado , B., & Rodríguez Lira , C. R. (21 de 10 de 2015). *Control de movimiento de un cuadricoptero mediante un controlador arduino*. Recuperado el 23 de 08 de 2016, de Tesis Institucionales: <http://tesis.ipn.mx:8080/xmlui/handle/123456789/15148>

Arduino. (s.f.). Recuperado el 20 de 10 de 2016, de Arduino: <http://playground.arduino.cc/Main/MPU-6050>

Arduino. (n.d de n.d de n.d). Recuperado el 14 de 9 de 2016, de <http://playground.arduino.cc/Main/MPU-6050>

Arduino.cl. (s.f.). Recuperado el 20 de 10 de 2016, de Arduino.cl: <http://arduino.cl/arduino-mega-2560/>

Artero, Ó. T. (2014). *ARDUINO Curso práctico de formación*. México DF: Alfaomega.

bigtronica. (s.f.). Recuperado el 20 de 10 de 2016, de bigtronica: <http://bigtronica.com/tarjetas/11-arduino-mega-2560-5053212000110.html>

Brokking, J. (s.f.). *Brokking.net*. Recuperado el 12 de 10 de 2016, de Brokking.net: <http://www.brokking.net/>

Castañeda García, F. N., Henao , R. A., & Valencia , F. A. ((n.d.) de (n.d.) de 2016). *Diseño e implementación del sistema de control de vuelo de un UAV*. Recuperado el 25 de 08 de 2016, de Repositorio Interinstitucional RI-UCM: <http://hdl.handle.net/10839/1246>

Cotte Corredor, J. M., & Moreno Pineda, A. F. (15 de 09 de 2010). *Diseño de control robusto de velocidad de motores brushless para robótica aérea* . Recuperado el 26 de 08 de 2016, de bdigital Repositorio Institucional UN: <http://www.bdigital.unal.edu.co/1896/>

- Enríquez Harper, G. (2012). *Máquinas Eléctricas*. México: LIMUSA.
- Faría Hernández, C. E. ((n.d.) de (n.d.) de (n.d.)). *Construye tu drone* . Recuperado el 26 de 08 de 2016, de DRONE CENTER: <http://dronecenter.blogspot.mx/p/construye-tu-drone.html>
- Guru, B., & Hiziroglu, H. (2003). *Máquinas Eléctricas y Transformadores*. New York: Alfaomega.
- Hernández Gaviño, R. (2010). *Introducción a los sistemas de control: Conceptos, aplicaciones y simulación con MATLAB*. México: PEARSON EDUCACIÓN.
- José, E. (2015). *Implementación de un dron cuadrucóptero con Arduino*. Pamplona.
- Kosow, I. (1992). *Máquinas Eléctricas y Transformadores*. New Jersey: Reverté.
- Martín-Gil, X. L. (2012). *Cuadrucóptero Arduino por control remoto Android*. Barcelona.
- Mega Ardupilot*. (n.d de n.d de n.d). Recuperado el 16 de 10 de 2016, de Mega Ardupilot: <http://www.ardupilot.co.uk/>
- Morales Delgado, D. G. (09 de 09 de 2015). *Sistema robotico aéreo para guiar a los visitantes del campus Manuel Haz Álvarez de la Universidad Tecnica Estatal De Quevedo*. Recuperado el 27 de 08 de 2016, de Universidad Técnica Estatal de Quevedo, Repositorio Digital: <http://repositorio.uteq.edu.ec/handle/43000/632>
- Moreno, M. (2013). <https://pastranamoreno.wordpress.com/>. Recuperado el 11 de 11 de 2016, de https://pastranamoreno.files.wordpress.com/2011/03/control_procesos-valvulas.pdf
- Olea Hernández, J., & Sánchez López, A. (27 de 07 de 2015). *Diseño y construcción de una aeronave de ala rotativa para operaciones de seguridad fronteriza y*

respuesta a emergencias. Recuperado el 24 de 08 de 2016, de Tesis Institucionales: <http://tesis.ipn.mx:8080/xmlui/handle/123456789/14598>

Robologs. (s.f.). Recuperado el 20 de 10 de 2016, de Robologs: <http://robologs.net/2014/10/15/tutorial-de-arduino-y-mpu-6050/>

Tabuchi Fukuhara, R. T. (26 de 06 de 2015). *Diseño de un Vehículo Aéreo no Tripulado de cuatro rotores*. Recuperado el 27 de 08 de 2016, de Repositorio Digital de Tesis PUCP: <http://tesis.pucp.edu.pe/repositorio/handle/123456789/6105>

Vicedo, V. R. (2014). *DISEÑO E IMPLEMENTACIÓN DE UN QUADCOPTER BASADO EN MICROCONTROLADOR ARDUINO*. Valencia.

what is arduino? ((n.d.) de (n.d.) de (n.d.)). Recuperado el 25 de 8 de 2016, de Arduino USA: <https://www.arduino.cc/>

6.4 ANEXOS

6.4.1 Programa De Prueba Para El Sistema De Radiofrecuencia

```
// Declaración de variables

unsigned long timer0,timer1,timer2,timer3,timer4;
byte canal_anterior_1,canal_anterior_2,canal_anterior_3,canal_anterior_4;
int Roll,Pitch,Arranque,Yaw;

void setup() {

// Activación de las interrupciones

PCICR |= (1<< PCIE0); //Inicializa las interrupciones
PCMSKO |=(1 << PCINT0); //Selección de pin 0 para generar una interrupción
PCMSKO |=(1 << PCINT1); //Selección de pin 1 para generar una interrupción
PCMSKO |=(1 << PCINT2); //Selección de pin 2 para generar una interrupción
PCMSKO |=(1 << PCINT3); //Selección de pin 3 para generar una interrupción

Serial.begin(9600); //Inicializa la comunicación serial
}

void loop() {
print(); //Subrutina para mostrar en el monitor serial los datos
delay(250); //Retardo
}

//Esta rutina se ejecuta cada vez que las entradas 8, 9, 10 u 11 cambien de estado y calcula el ancho de pulso
//de la señal de entrada del receptor

ISR(PCINT0_vect){ //Inicia la secuencia de instrucciones para cada interrupción
//Canal 1
timer0 = micros(); //Inicia el conteo de tiempo a partir de la interrupción
if(canal_anterior_1 == 0 && PINB & B00000001){ //Condiciones para calcular el periodo
canal_anterior_1 = 1; //Si la variable cambia de estado de 0 a 1
timer1=timer0; //Asigna a la variable un estado alto (1)
}

else if(canal_anterior_1 == 1 && !(PINB & B00000001)){ //Si la variable cambia de estado de 1 a 0
canal_anterior_1 = 0; //Asigna a la variable un estado bajo (0)
Roll= timer0 - timer1; //Calcula el ancho de pulso restando el valor inicial
//Cálcula el ancho de pulso restando el valor inicial
//(cuando ocurre la interrupcion) del valor total del pulso
//(desde que inicia hasta que la variable cambia de estado nuevamente)
}

//Mismo proceso para cada canal
//Canal 2
if(canal_anterior_2 == 0 && PINB & B00000010){
canal_anterior_2 = 1;
timer2=timer0;
}

else if(canal_anterior_2 == 1 && !(PINB & B00000010)){
canal_anterior_2 = 0;
Pitch= timer0 - timer2;
}
```

```

}

//Canal 3
if(canal_anterior_3 == 0 && PINB & B00000100){
  canal_anterior_3 = 1;
  timer3=timer0;
}
else if(canal_anterior_3 == 1 && !(PINB & B00000100)){
  canal_anterior_3 = 0;
  Arranque= timer0 - timer3;
}

//Canal 2
if(canal_anterior_4 == 0 && PINB & B00001000){
  canal_anterior_4 = 1;
  timer4=timer0;
}
else if(canal_anterior_4 == 1 && !(PINB & B00001000)){
  canal_anterior_4 = 0;
  Yaw= timer0 - timer4;
}
}

//Rutina para imprimir los valores obtenidos de ancho de pulso
void print() {
  Serial.print("Roll = ");
  Serial.print(Roll);
  Serial.print(" Pitch = ");
  Serial.print(Pitch);
  Serial.print(" Arranque = ");
  Serial.print(Arranque);
  Serial.print(" Yaw = ");
  Serial.println(Yaw);
}

```

6.4.2 Programa Para Calibración De La IMU

```

#include <Wire.h> //Libreria I2C

#define MPU6050 0x68 //Direccion I2C de la IMU
#define Accl_AFS 16384.0 //Factor de escala de sensibilidad del acelerometro, constante para
//obtener datos coherentes a partir de los datos en bruto +- 2g
#define Giro_FS 131.0 //Factor de escala de sensibilidad del giroscopio, constante para
//obtener datos coherentes a partir de los datos en bruto 250dps
#define RAD_A_DEG = 57.295779 //Conversion de radianes a grados 180/PI

//Declaración de variables

int16_t AcclX, AcclY, AcclZ, GiroX, GiroY, GiroZ; //Valores de salida del MPU6050 en bruto(raw), enteros de 16 bits
float A_accl_X, A_accl_Y; //Ángulo X y Y del acelerometro
float A_giro_X, A_giro_Y, A_giro_Z; //Ángulo X, Y y Z del giroscopio
float imu_vp_roll, imu_vp_pitch, imu_vp_yaw; //Ángulos resultado del filtro, estos ángulos corresponderan a la
//variable de proceso(realimentación del lazo de control)

void setup()
{
  Wire.begin(); //Iniciar I2C como maestro
  Wire.beginTransmission(MPU6050); //Inicia la comunicación por I2C con el dispositivo MPU
  Wire.write(0x6B); //Enviar el registro de inicio solicitado
  Wire.write(0); //Establecer el registro de inicio solicitado
  Wire.endTransmission(true); //Fin de la transmisión
  Serial.begin(9600); //Inicia el monitor série para ver los resultados.
}

void loop()
{
  //Leer los valores del MPU6050//

  //Acelerometro, +- 2g
  Wire.beginTransmission(MPU6050);
  Wire.write(0x3B); //Pedir el registro 0x3B - corresponde al AcclX
  Wire.endTransmission(false);
  Wire.requestFrom(MPU6050, 6, true); //A partir del 0x3B, se piden 6 bytes
  AcclX=Wire.read()<<8|Wire.read(); //Añade el byte alto y bajo a la variable AcclX, cada valor ocupa 2 registros
  AcclY=Wire.read()<<8|Wire.read(); //Añade el byte alto y bajo a la variable AcclY
  AcclZ=Wire.read()<<8|Wire.read(); //Añade el byte alto y bajo a la variable AcclZ

  //A partir de los valores del acelerometro, se calculan los angulos X y Y respectivamente, con la formula de la tangente.
  A_accl_X = atan(AcclY/Accl_AFS)/sqrt(pow((AcclX/Accl_AFS),2) + pow((AcclZ/Accl_AFS),2))*RAD_TO_DEG;
  A_accl_Y = atan(-1*(AcclX/Accl_AFS)/sqrt(pow((AcclY/Accl_AFS),2) + pow((AcclZ/Accl_AFS),2)))*RAD_TO_DEG;

  //Giroscopio, 250 dps
  Wire.beginTransmission(MPU6050);
  Wire.write(0x43);

  Wire.endTransmission(false);
  Wire.requestFrom(MPU6050, 6, true); //Solicitar 6 bytes del giroscopio
  GiroX=Wire.read()<<8|Wire.read(); //Añade el byte alto y bajo a la variable GiroX
  GiroY=Wire.read()<<8|Wire.read(); //Añade el byte alto y bajo a la variable GiroY
  GiroZ=Wire.read()<<8|Wire.read(); //Añade el byte alto y bajo a la variable GiroZ

  //Ángulos X, Y y Z del Giroscopio
  A_giro_X = GiroX/Giro_FS; //Ángulo X del giroscopio. Valor en bruto/constante
  A_giro_Y = GiroY/Giro_FS; //Ángulo Y del giroscopio. Valor en bruto/constante
  A_giro_Z = GiroZ/Giro_FS; //Ángulo Z del giroscopio. Valor en bruto/constante

  //Aplica Filtro Complementario
  imu_vp_roll = 0.98 *(imu_vp_roll + A_giro_X*0.010) + 0.02*A_accl_X; //Toma el 98% del valor del ángulo en X, más el 1% del
//ángulo X del giroscopio, más 2% del ángulo X del acelerometro
  imu_vp_pitch = 0.98 *(imu_vp_pitch + A_giro_Y*0.010) + 0.02*A_accl_Y; //Toma el 98% del valor del ángulo en Y, más el 1% del
//ángulo Y del giroscopio, más 2% del ángulo Y del acelerometro
  imu_vp_yaw = 0.98 *(imu_vp_yaw + A_giro_Z*0.010) + 0.004; //Toma el 98% del valor del ángulo en Z, más el 1% del ángulo Z
//del giroscopio, más una constante

  //Mostrar los valores por consola
  Serial.print("Ángulo X (Roll): ");
  Serial.print( imu_vp_roll );
  Serial.print(" -- ");
  Serial.print("Ángulo Y (Pitch): ");
  Serial.println( imu_vp_pitch );

  delay(10);
}

```

6.4.3 Programa Para El Encendido Y Control Simultáneo De Velocidad En Cuatro Motores

```
// Declaración de variables

unsigned long timer0,timer1,timer2,timer3,timer4, timerX;
byte canal_anterior_1,canal_anterior_2,canal_anterior_3,canal_anterior_4;
int Canal_receptor_1, Canal_receptor_2, Canal_receptor_3, Canal_receptor_4;
unsigned long timer_canal_1, timer_canal_2, timer_canal_3, timer_canal_4, esc_timer;

void setup() {

  DDRK |= B11110000; //Configura el registro del puerto digital K (bits 4, 5, 6 y 7) como salida
                    //pines 12, 13, 14 y 15
                    //Los pines de la placa Arduino Uno estan configurados como entradas por default,
                    //por ello no es necesario que se declaren como tal

  // Activación de las interrupciones

  PCICR |=(1<< PCIE0); //Inicializa las interrupciones //pines en arduino mega
  PCMSKO |=(1 << PCINT0); //Selección de pin 0 para generar una interrupción //53
  PCMSKO |=(1 << PCINT1); //Selección de pin 1 para generar una interrupción //52
  PCMSKO |=(1 << PCINT2); //Selección de pin 2 para generar una interrupción //51
  PCMSKO |=(1 << PCINT3); //Selección de pin 3 para generar una interrupción //50

  timerX = micros(); //Inicia el timerX
}

void loop() {
  //Hace que el refresh rate (frecuencia de actualización) sea siempre igual a 250Hz (1/250 Hz = 4000 us)

  while(timerX + 4000 > micros()); //Inicia el pulso después de 4000 us
  timerX = micros(); //Reestablece el timer
  PORTK |= B11110000; //Pone en alto las salidas 4, 5, 6 y 7
  timer_canal_1 = Canal_receptor_3 + timerX; //Calcula el tiempo en que la entrada digital 53 es puesta en bajo (el tiempo en que el pulso cae)
  timer_canal_2 = Canal_receptor_3 + timerX; //Calcula el tiempo en que la entrada digital 52 es puesta en bajo (el tiempo en que el pulso cae)
  timer_canal_3 = Canal_receptor_3 + timerX; //Calcula el tiempo en que la entrada digital 51 es puesta en bajo (el tiempo en que el pulso cae)
  timer_canal_4 = Canal_receptor_3 + timerX; //Calcula el tiempo en que la entrada digital 50 es puesta en bajo (el tiempo en que el pulso cae)

  while(PORTK >= 16){ //Ejecuta el ciclo hasta que los cuatro bits más altos del puerto K están en bajo
    esc_timer = micros(); //Tiempo actual
    if(timer_canal_1 <= esc_timer)PORTK &= B11101111; //Cuando se cumpla la condición la salida 4 es puesta en bajo
    if(timer_canal_2 <= esc_timer)PORTK &= B11011111; //Cuando se cumpla la condición la salida 5 es puesta en bajo
    if(timer_canal_3 <= esc_timer)PORTK &= B10111111; //Cuando se cumpla la condición la salida 6 es puesta en bajo
    if(timer_canal_4 <= esc_timer)PORTK &= B01111111; //Cuando se cumpla la condición la salida 7 es puesta en bajo
  }
}

//Esta rutina se ejecuta cada vez que las entradas 8, 9, 10 u 11 cambian de estado y calcula el ancho de pulso de la señal de entrada del receptor

ISR(PCINT0_vect){ //Inicia la secuencia de instrucciones para cada interrupción
  //Canal 1
  timer0 = micros(); //Inicia el conteo tiempo a partir de la interrupción
  if(canal_anterior_1 == 0 && PINB & B00000001){ //Si la variable cambia de estado de 0 a 1
    canal_anterior_1 = 1; //Asigna a la variable un estado alto (1)
    timer1=timer0; //Guarda el valor del timer0 en el timer1
  }
  else if(canal_anterior_1 == 1 && !(PINB & B00000001)){ //Si la variable cambia de estado de 1 a 0
    canal_anterior_1 = 0; //Asigna a la variable un estado bajo (0)
    Canal_receptor_1= timer0 - timer1; //Calcula el ancho de pulso restando el valor inicial (cuando ocurre la interrupción) del
    //valor total del pulso (desde que inicia hasta que la variable cambia de estado nuevamente)
  }
  //Mismo proceso para cada canal
  //Canal 2
  if(canal_anterior_2 == 0 && PINB & B00000010){
    canal_anterior_2 = 1;
    timer2=timer0;
  }
  else if(canal_anterior_2 == 1 && !(PINB & B00000010)){
    canal_anterior_2 = 0;
    Canal_receptor_2= timer0 - timer2;
  }
}
```

```

//Canal 3
if(canal_anterior_3 == 0 && PINB & B00000100){
  canal_anterior_3 = 1;
  timer3=timer0;
}
else if(canal_anterior_3 == 1 && !(PINB & B00000100)){
  canal_anterior_3 = 0;
  Canal_receptor_3= timer0 - timer3;
}

//Canal 2
if(canal_anterior_4 == 0 && PINB & B00001000){
  canal_anterior_4 = 1;
  timer4=timer0;
}
else if(canal_anterior_4 == 1 && !(PINB & B00001000)){
  canal_anterior_4 = 0;
  Canal_receptor_4= timer0 - timer4;
}

```

6.4.4 Prueba A La Alarma De Bajo Voltaje En La Batería

```

int voltaje_bateria;

void setup() {
  Serial.begin(9600); //Inicia la comunicación serial
}

void loop() {
  voltaje_bateria = (analogRead(A0)+70)*1.2023; //Lee el voltaje de la batería y lo almacena en la variable voltaje_bateria
  voltaje_bateria = voltaje_bateria*0.92+(analogRead(A0)+70)*0.0962; //Filtro complementario para evitar el ruido en la lectura
  // (92% + 8% = 0.92 + 0.08) //1.2023(0.08)= 0.0962
  if(voltaje_bateria < 1050) //Si el voltaje se encuentra dentro del rango especificado (se considera
  //que el voltaje de la batería no debe ser menor a 10.5V para no dañarla)
  {digitalWrite(10,HIGH);} //emite una alerta visual mediante el led
  Serial.println(voltaje_bateria); //imprime en pantalla el valor del voltaje de la batería
  delay(100); //retardo
}

```

6.4.5 Cálculo De Salidas Para Controlador proporcional-integral-derivativo (PID)

```

//Una vez que se conocen las entradas, se pueden calcular las salidas del PID
//Cálculos para el movimiento Roll

error_pid = imu_vp_roll - setpoint_roll;
i_roll += ganancia_i_roll * error_pid;
if(i_roll > salida_maxima_roll) i_roll = salida_maxima_roll;

else if(i_roll < (salida_maxima_roll * -1)) i_roll = salida_maxima_roll * -1;

salida_pid_roll = (ganancia_p_roll * error_pid) + i_roll + (ganancia_d_roll * (error_pid - ultimo_error_roll));
if(salida_pid_roll > salida_maxima_roll) salida_pid_roll = salida_maxima_roll;

else if(salida_pid_roll < (salida_maxima_roll * -1)) salida_pid_roll = salida_maxima_roll * -1;

ultimo_error_roll = error_pid;

//Cálculos para el movimiento Pitch

error_pid = imu_vp_pitch - setpoint_pitch;
i_pitch += ganancia_i_pitch * error_pid;
if(i_pitch > salida_maxima_pitch) i_pitch = salida_maxima_pitch;

else if(i_pitch < (salida_maxima_pitch * -1)) i_pitch = salida_maxima_pitch * -1;

salida_pid_pitch = (ganancia_p_pitch * error_pid) + i_pitch + (ganancia_d_pitch * (error_pid - ultimo_error_pitch));
if(salida_pid_pitch > salida_maxima_pitch) salida_pid_pitch = salida_maxima_pitch;

else if(salida_pid_pitch < (salida_maxima_pitch * -1)) salida_pid_pitch = salida_maxima_pitch * -1;

ultimo_error_pitch = error_pid;

```

6.4.6 Cálculo De Los Pulsos Enviados A Cada ESC, Según El Movimiento Del Cuadricóptero

```

ch3 = Canal_receptor_3;

if (encoder==1) {
    if (ch3 > 1800) {ch3 = 1800;}
    esc_1 = ch3 + salida_pid_pitch - salida_pid_roll;
    esc_2 = ch3 + salida_pid_pitch + salida_pid_roll;
    esc_3 = ch3 - salida_pid_pitch + salida_pid_roll;
    esc_4 = ch3 - salida_pid_pitch - salida_pid_roll;

    if(esc_1 < 1100) esc_1 = 1100;
    if(esc_1 > 2000) esc_1 = 2000;

    if(esc_2 < 1100) esc_2 = 1100;
    if(esc_2 > 2000) esc_2 = 2000;

    if(esc_3 < 1100) esc_3 = 1100;
    if(esc_3 > 2000) esc_3 = 2000;

    if(esc_4 < 1100) esc_4 = 1100;
    if(esc_4 > 2000) esc_4 = 2000;
}

else{
    esc_1 = 1000;
    esc_2 = 1000;
    esc_3 = 1000;
    esc_4 = 1000;
}

```


6.4.7 Lazo Cerrado De Control

```
#include <Wire.h> //Libreria I2C
#define MPU6050 0x68 //Direccion I2C de la IMU
#define Acel_AFS 16384.0 //Factor de escala de sensibilidad del acelerometro, constante para obtener
//datos coherentes a partir de los datos en bruto +- 2g
#define Giro_FS 131.0 //Factor de escala de sensibilidad del giroscopio, constante para obtener
//datos coherentes a partir de los datos en bruto 250dps
#define RAD_A_DEG = 57.295779 //Conversión de radianes a grados 180/PI

//Variables para el sistema de comunicación mediante radiofrecuencia//

unsigned long timer0,timer1,timer2,timer3,timer4, timerX;
byte canal_anterior_1,canal_anterior_2,canal_anterior_3,canal_anterior_4;
int Canal_receptor_1, Canal_receptor_2, Canal_receptor_3, Canal_receptor_4;
unsigned long timer_canal_1, timer_canal_2, timer_canal_3, timer_canal_4,esc_timer;
int voltaje_bateria;
int encender,ch3;
int esc_1, esc_2, esc_3, esc_4;

//Variables para la IMU//

int16_t AcelX, AcelY, AcelZ, GiroX, GiroY, GiroZ; //Valores de salida del MPU6050 en bruto(raw), enteros de 16 bits
float A_aceL_X, A_aceL_Y ; //Ángulo X y Y del acelerometro
float A_giro_X, A_giro_Y, A_giro_Z ; //Ángulo X, Y y Z del giroscopio
float imu_vp_roll, imu_vp_pitch, imu_vp_yaw; //Ángulos resultado del filtro

//Variables para el PID//

float error_pid;

//Movimiento Roll

float i_roll, setpoint_roll, salida_pid_roll, ultimo_error_roll;
float ganancia_p_roll = 6 ; //Ganancia para el control proporcional del movimiento roll
float ganancia_i_roll = 0.08 ; //Ganancia para el control integral del movimiento roll
float ganancia_d_roll = 18 ; //Ganancia para el control deriatiivo del movimiento roll
int salida_maxima_roll = 300; //Limita la salida máxima del pid roll

//Movimiento Pitch

float i_pitch, setpoint_pitch, salida_pid_pitch, ultimo_error_pitch;
float ganancia_p_pitch = 6 ; //Ganancia para el control proporcional del movimiento pitch
float ganancia_i_pitch = 0.08 ; //Ganancia para el control integral del movimiento pitch
float ganancia_d_pitch = 18 ; //Ganancia para el control deriatiivo del movimiento pitch
int salida_maxima_pitch = 300; //Limita la salida máxima del pid pitch

void setup(){

Wire.begin(); //Iniciar I2C como maestro

DDRK |= B11110000; //Configura el registro del puerto digital K (bits 4, 5, 6 y 7) como salida
//pines 12, 13, 14 y 15
//Los pines de la placa Arduino Uno estan configurados como entradas por default,
//por ello no es necesario que se declaren como tal

DDRB |= B00010000; //Configura el registro del puerto digital B (bit 4) como salida//pin 10

Wire.beginTransmission(MPU6050); //Inicia la comunicación por I2C con el dispositivo MPU
Wire.write(0x6B); //Enviar el registro de inicio solicitado
Wire.write(0); //Establecer el registro de inicio solicitado
Wire.endTransmission(true); //Fin de la transmisión
Serial.begin(9600);
delay(250); //Esperar a que la IMU inicie
```

```

// Activación de las interrupciones

PCICR |= (1 << PCIE0); //Inicializa las interrupciones //pines en arduino mega
PCMSK0 |= (1 << PCINT0); //Selección de pin 0 para generar una interrupción //53
PCMSK0 |= (1 << PCINT1); //Selección de pin 1 para generar una interrupción //52
PCMSK0 |= (1 << PCINT2); //Selección de pin 2 para generar una interrupción //51
PCMSK0 |= (1 << PCINT3); //Selección de pin 3 para generar una interrupción //50

encender=0; //Establece la variable inicio a cero

//Lee el voltaje de la batería y emite una alerta visual cuando este se encuentre debajo de un valor especificado//

//Para compensar la pérdida de voltaje en el diodo del circuito físico se suma 29 al valor medido
//12.3V es el valor máximo de la batería y para este caso equivale a 5V que es el máximo voltaje que se mide mediante analogRead
//mide de 0 a 5V que equivalen a valores digitales de 0 a 1023. Por lo tanto, 12.3V equivale a 1023 analogRead(0).
//La variable voltaje_bateria debe mostrar un valor de 1230 cuando el voltaje de la batería es 12.3V, y un valor de 1050 cuando el voltaje es 10.5 V, por ello
//1230 / 1023 = 1.2023

voltaje_bateria = (analogRead(A0)+70)*1.2023; //Lee el voltaje de la batería y lo almacena en la variable voltaje_bateria
}

void loop(){

//Leer los valores del MPU6050

//Acelerometro
Wire.beginTransmission(MPU6050);
Wire.write(0x3B); //Pedir el registro 0x3B - corresponde al AcelX
Wire.endTransmission(false);
Wire.requestFrom(MPU6050,6,true); //A partir del 0x3B, se piden 6 bytes
AcelX=Wire.read()<<8|Wire.read(); //Añade el byte alto y bajo a la variable AcelX, cada valor ocupa 2 registros
AcelY=Wire.read()<<8|Wire.read(); //Añade el byte alto y bajo a la variable AcelY
AcelZ=Wire.read()<<8|Wire.read(); //Añade el byte alto y bajo a la variable AcelZ

//A partir de los valores del acelerometro, se calculan los angulos X y Y respectivamente, con la formula de la tangente.

A_acel_X = atan((AcelY/Acel_AFS)/sqrt(pow((AcelX/Acel_AFS),2) + pow((AcelZ/Acel_AFS),2)))*RAD_TO_DEG;
A_acel_Y = atan(-1*(AcelX/Acel_AFS)/sqrt(pow((AcelY/Acel_AFS),2) + pow((AcelZ/Acel_AFS),2)))*RAD_TO_DEG;

//Giroscopio
Wire.beginTransmission(MPU6050);
Wire.write(0x43);
Wire.endTransmission(false);
Wire.requestFrom(MPU6050,6,true); //Solicitar 6 bytes del giroscopio
GiroX=Wire.read()<<8|Wire.read(); //Añade el byte alto y bajo a la variable GiroX
GiroY=Wire.read()<<8|Wire.read(); //Añade el byte alto y bajo a la variable GiroY
GiroZ=Wire.read()<<8|Wire.read(); //Añade el byte alto y bajo a la variable GiroZ

//Ángulos X, Y y Z del Giroscopio
A_giro_X = GiroX/Giro_FS; //Ángulo X del giroscopio. Valor en bruto/constante
A_giro_Y = GiroY/Giro_FS; //Ángulo Y del giroscopio. Valor en bruto/constante
A_giro_Z = GiroZ/Giro_FS; //Ángulo Z del giroscopio. Valor en bruto/constante

//Aplicar el Filtro Complementario
imu_vp_roll = (0.98 *(imu_vp_roll + A_giro_X*0.010) + 0.02*A_acel_X); //Toma el 98% del valor del ángulo en X, más el 1%
//del ángulo X del giroscopio, más 2% del ángulo X del acelerometro
imu_vp_pitch = (0.98 *(imu_vp_pitch + A_giro_Y*0.010) + 0.02*A_acel_Y); //Toma el 98% del valor del ángulo en Y, más el 1%
//del ángulo Y del giroscopio, más 2% del ángulo Y del acelerometro
imu_vp_yaw = 0.98 *(imu_vp_yaw + A_giro_Z*0.010) + 0.004; //Toma el 98% del valor del ángulo en Z, más el 1% del ángulo Z
//del giroscopio, más una constante

```

```

//Iniciar los motores cuando la palanca yaw se encuentre en posición central.
if(Canal_receptor_3 < 1200 && Canal_receptor_4 > 1450){
  encender=1;
  //Reiniciar los controladores PID
  i_roll = 0;
  ultimo_error_roll = 0;
  i_pitch = 0;
  ultimo_error_pitch = 0;
}

//Condición para detener los motores: palanca de arranque en su posición más baja y palanca yaw a la derecha.
if(encender==1 && Canal_receptor_3 < 1200 && Canal_receptor_4 > 1750)encender=0;
//La referencia del PID roll (set point) en grados por segundo (dps) es determinada por el canal receptor 1 que corresponde al movimiento roll
setpoint_roll = 0;
//Se establecerá una banda muerta de 20us para asegurar que las variaciones en la entrada no ocasionen una actuación errónea del PID
if(Canal_receptor_1 > 1510) setpoint_roll = (Canal_receptor_1 - 1510);
else if(Canal_receptor_1 < 1490) setpoint_roll = (Canal_receptor_1 - 1490);

//La referencia del PID pitch (set point) en grados por segundo (dps) es determinada por el canal receptor 2 que corresponde al movimiento pitch
setpoint_pitch = 0;
//Se establecerá una banda muerta de 20us para asegurar que las variaciones en la entrada no ocasionen una actuación errónea del PID
if(Canal_receptor_2 > 1510) setpoint_pitch = (Canal_receptor_2 - 1510);
else if(Canal_receptor_2 < 1490) setpoint_pitch = (Canal_receptor_2 - 1490);

//Una vez que se conocen las entradas, se pueden calcular las salidas del PID
//Cálculos para el movimiento Roll

error_pid = imu_vp_roll - setpoint_roll; //Cálcula el error del PID VP-SP
i_roll += ganancia_i_roll * error_pid; //Cálcula el término integral del movimiento roll
if(i_roll > salida_maxima_roll) i_roll = salida_maxima_roll; //Limita el valor del término integral al valor //establecido como máximo (+)
else if(i_roll < (salida_maxima_roll * -1)) i_roll = salida_maxima_roll * -1; //Limita el valor del término integral al valor //establecido como máximo (-)
salida_pid_roll = (ganancia_p_roll * error_pid) + i_roll + (ganancia_d_roll * (error_pid - ultimo_error_roll)); //Cálcula la salida del PID para el movimiento roll
if(salida_pid_roll > salida_maxima_roll) salida_pid_roll = salida_maxima_roll; //Limita el valor de salida del PID para el movimiento //roll al valor establecido como máximo (+)
else if(salida_pid_roll < (salida_maxima_roll * -1)) salida_pid_roll = salida_maxima_roll * -1; //Limita el valor de salida del PID para el movimiento //roll al valor establecido como máximo (-)
ultimo_error_roll = error_pid; //Guarda el valor de la variable error_pid //en la variable ultimo_error_roll

//Cálculos para el movimiento Pitch

error_pid = imu_vp_pitch - setpoint_pitch; //Cálcula el error del PID VP-SP
i_pitch += ganancia_i_pitch * error_pid; //Cálcula el término integral del movimiento pitch
if(i_pitch > salida_maxima_pitch) i_pitch = salida_maxima_pitch; //Limita el valor del término integral al valor //establecido como máximo (+)
else if(i_pitch < (salida_maxima_pitch * -1)) i_pitch = salida_maxima_pitch * -1; //Limita el valor del término integral al valor //establecido como máximo (-)
salida_pid_pitch = (ganancia_p_pitch * error_pid) + i_pitch + (ganancia_d_pitch * (error_pid - ultimo_error_pitch)); //Cálcula la salida del PID para el movimiento pitch
if(salida_pid_pitch > salida_maxima_pitch) salida_pid_pitch = salida_maxima_pitch; //Limita el valor de salida del PID para el movimiento //pitch al valor establecido como máximo (+)
else if(salida_pid_pitch < (salida_maxima_pitch * -1)) salida_pid_pitch = salida_maxima_pitch * -1; //Limita el valor de salida del PID para el movimiento //pitch al valor establecido como máximo (-)
ultimo_error_pitch = error_pid; //Guarda el valor de la variable error_pid en la //variable ultimo_error_pitch

voltaje_bateria = voltaje_bateria*0.92+(analogRead(A0)+70)*0.0962; //Filtro complementario para evitar el ruido en la lectura
//((92% + 8% = 0.92 + 0.08) //1.2023(0.08)= 0.0962
if(voltaje_bateria < 1050) digitalWrite(10,HIGH); //Si el voltaje se encuentra dentro del rango especificado
//((se considera que el voltaje de la batería no debe ser menor a 10.5V para no dañarla)
//emite una alerta visual encendiendo el led

```

```

ch3 = Canal_receptor_3; //Se establece la señal de arranque como señal base

if (encender==1){ //Iniciar los motores
    if (ch3 > 1800) {ch3 = 1800;} //Margen
    esc_1 = ch3 + salida_pid_pitch - salida_pid_roll; //Calcula el pulso para el esc 1
    esc_2 = ch3 + salida_pid_pitch + salida_pid_roll; //Calcula el pulso para el esc 2
    esc_3 = ch3 - salida_pid_pitch + salida_pid_roll; //Calcula el pulso para el esc 3
    esc_4 = ch3 - salida_pid_pitch - salida_pid_roll; //Calcula el pulso para el esc 4

    if(esc_1 < 1100) esc_1 = 1100; //Mantiene los motores girando
    if(esc_1 > 2000)esc_1 = 2000; //Limita el pulso del esc 1 a 2000us

    if(esc_2 < 1100) esc_2 = 1100; //Mantiene los motores girando
    if(esc_2 > 2000)esc_2 = 2000; //Limita el pulso del esc 2 a 2000us

    if(esc_3 < 1100) esc_3 = 1100; //Mantiene los motores girando
    if(esc_3 > 2000)esc_3 = 2000; //Limita el pulso del esc 3 a 2000us

    if(esc_4 < 1100) esc_4 = 1100; //Mantiene los motores girando
    if(esc_4 > 2000)esc_4 = 2000; //Limita el pulso del esc 4 a 2000us
}

else{
    esc_1 = 1000; //Si la variable inicio es diferente a 2 envía un pulso de 1000us al esc 1
    esc_2 = 1000; //Si la variable inicio es diferente a 2 envía un pulso de 1000us al esc 2
    esc_3 = 1000; //Si la variable inicio es diferente a 2 envía un pulso de 1000us al esc 3
    esc_4 = 1000; //Si la variable inicio es diferente a 2 envía un pulso de 1000us al esc 4
}

//Hace que el refresh rate (frecuencia de actualización) sea siempre igual a 250Hz (1/250 Hz = 4000 us)

while(timerX + 4000 > micros()); //Inicia el pulso después de 4000 us
timerX = micros(); //Reestablece el timer
PORTK |= B11110000; //Pone en alto las salidas 4, 5, 6 y 7
timer_canal_1 = esc_1 + timerX; //Calcula el tiempo en que la entrada digital 53 es puesta en bajo (el tiempo en que el pulso cae)
timer_canal_2 = esc_2 + timerX; //Calcula el tiempo en que la entrada digital 52 es puesta en bajo (el tiempo en que el pulso cae)
timer_canal_3 = esc_3 + timerX; //Calcula el tiempo en que la entrada digital 51 es puesta en bajo (el tiempo en que el pulso cae)
timer_canal_4 = esc_4 + timerX; //Calcula el tiempo en que la entrada digital 50 es puesta en bajo (el tiempo en que el pulso cae)

while(PORTK >= 16){ //Ejecuta el ciclo hasta que los cuatro bits más altos del puerto K están en bajo
    esc_timer = micros(); //Tiempo actual
    if(timer_canal_1 <= esc_timer)PORTK &= B11101111; //Cuando se cumpla la condición la salida 4 es puesta en bajo
    if(timer_canal_2 <= esc_timer)PORTK &= B11011111; //Cuando se cumpla la condición la salida 5 es puesta en bajo
    if(timer_canal_3 <= esc_timer)PORTK &= B10111111; //Cuando se cumpla la condición la salida 6 es puesta en bajo
    if(timer_canal_4 <= esc_timer)PORTK &= B01111111; //Cuando se cumpla la condición la salida 7 es puesta en bajo
}
}

```

```

//Esta rutina se ejecuta cada vez que las entradas 8, 9, 10 u 11 cambien de estado y calcula el ancho de pulso de la señal de entrada del receptor

ISR(PCINT0_vect){ //Inicia la secuencia de instrucciones para cada interrupción
//Canal 1
timer0 = micros(); //Inicia el conteo tiempo a partir de la interrupción
if(canal_anterior_1 == 0 && PINB & B00000001){ //Si la variable cambia de estado de 0 a 1
    canal_anterior_1 = 1; //Asigna a la variable un estado alto (1)
    timer1=timer0; //Guarda el valor del timer0 en el timer1
}
else if(canal_anterior_1 == 1 && !(PINB & B00000001)){ //Si la variable cambia de estado de 1 a 0
    canal_anterior_1 = 0; //Asigna a la variable un estado bajo (0)
    Canal_receptor_1= timer0 - timer1; //Calcula el ancho de pulso restando el valor inicial (cuando ocurre la interrupcion) del
//valor total del pulso (desde que inicia hasta que la variable cambia de estado nuevamente)
}
//Mismo proceso para cada canal
//Canal 2
if(canal_anterior_2 == 0 && PINB & B00000010){
    canal_anterior_2 = 1;
    timer2=timer0;
}
else if(canal_anterior_2 == 1 && !(PINB & B00000010)){
    canal_anterior_2 = 0;
    Canal_receptor_2= timer0 - timer2;
}

//Canal 3
if(canal_anterior_3 == 0 && PINB & B00000100){
    canal_anterior_3 = 1;
    timer3=timer0;
}
else if(canal_anterior_3 == 1 && !(PINB & B00000100)){
    canal_anterior_3 = 0;
    Canal_receptor_3= timer0 - timer3;
}

//Canal 4
if(canal_anterior_4 == 0 && PINB & B00001000){
    canal_anterior_4 = 1;
    timer4=timer0;
}
else if(canal_anterior_4 == 1 && !(PINB & B00001000)){
    canal_anterior_4 = 0;
    Canal_receptor_4= timer0 - timer4;
}
}
}

```