CAPÍTULO IV - Desarrollo

1. ***Circuito de alimentación y control de motores***

Se desarrolló un circuito para llevar a cabo la alimentación y regulación de velocidad de los motores de corriente continua del chasis de cuadricóptero Draganflyer V utilizado en el presente trabajo. En primer lugar, se decidió utilizar una batería de Polímero de Litio (Li-Po, del inglés Lithium Polymer) de 3 celdas, ya que es la configuración de alimentación de los motores de corriente continua recomendada por el fabricante en [Draganfly Innovations 2006].Por otro lado, en [Nadales 2009], se describen las características de funcionamiento de los motores en cuestión, y la relación entre el voltaje de alimentación y la corriente consumida por los mismos al hacer girar las hélices del cuadricóptero. A continuación se detallan las características de los motores citadas previamente:

* Modelo de motores DC: Mabuchi RC 280SA.
* Voltaje de alimentación recomendado: 4,5 a 12 Voltios.
* Consumo sin carga: 0,14 Amperios.
* Consumo medio: 4 Amperios.
* Peso individual: 47 Gramos.



Ilustracion 4. Relación voltaje-corriente de los motores DC del chasis Draganflyer V.

Fuente: [Nadales 2009]

Tomando en cuenta las características de los motores anteriormente expuestas, se puede estimar un consumo promedio de 16 Amperios al alimentar los cuatro motores de forma continua, y de hasta 24 amperios considerando un pico de consumo de hasta 2 amperios por motor al realizar el encendido de los mismos, los cuales deben ser provistos por la batería de polímero de litio de forma instantánea. Para una batería de polímero de litio cualquiera, se tiene que:

Para satisfacer la necesidad de consumo de los motores del cuadricóptero se utilizó una batería Li-Po Yuntong de 3 celdas, con una capacidad de 1350 miliAmperios por Hora (mAH), y una tasa de descarga de 25C. Para la batería seleccionada se estimaron los siguientes valores de consumo y duración:

Para regular la velocidad de los motores de corriente continua del cuadricóptero, se utilizó Modulación por Ancho de Pulso (PWM, del inglés Pulse Width Modulation), enviada desde los puertos de la tarjeta Arduino Nano 3.0, para realizar una conmutación rápida entre los estados de encendido y apagado de los motores, pudiendo mantener un control de lazo abierto sobre su velocidad a partir del ciclo de trabajo del PWM emitido.

****

Ilustracion 4. Diagrama de bloques del sistema de control de velocidad de los motores DC del cuadricóptero.

Fuente: elaboración propia.

**FOTO CIRCUITO DE CONTROL DE MOTORES FINAL!!!!**

1. ***Circuito de lógica, sensores y comunicación***

Se diseñó una placa de circuito que consta de una serie de puertos pin header para situar la tarjeta Arduino y la unidad de medición inercial, además de conectar el sensor de ultrasonido, las salidas de PWM y el módulo de XBEE mediante cables “jumper”. La misma sirve de interfaz de comunicación entre la tarjeta Arduino, y los demás módulos del cuadricóptero. Además, permite la alimentación de todo el circuito haciendo uso de la etapa de regulación de voltaje embebida en la tarjeta Arduino, la cual puede ser alimentada con voltajes de entre 6 y 20 voltios. A continuación se detalla el consumo promedio de cada componente del circuito, y el consumo total del mismo:

Tabla 4. Desglose del consumo promedio por componente, y acumulado, del circuito de lógica, sensores y comunicación.

|  |  |
| --- | --- |
| **Componente** | **Consumo promedio** |
| Arduino Nano | 50 mA |
| Pololu minIMU-9 V2 | 10mA |
| XBee Series 1mW Wire Antenna – Series 1 | 50mA |
| Parallax Ping | 30mA |
| 4 Optocouplers 4N26 – Módulo de control de motores | 240 mA |
| **Total** | **380mA** |

Fuente: elaboración propia.

Para alimentación del módulo de lógica, sensores y comunicación durante la realización de pruebas de vuelo se utilizó una batería de polímero de litio Yuntong de 300mAH, con un coeficiente de descarga de 3C. A continuación se presenta el cálculo de la máxima capacidad de descarga, y duración de la carga de la misma al alimentar el circuito desarrollado:



Ilustracion 4. Placa del circuito de lógica, sensores y comunicación.

Fuente: elaboración propia.

1. ***Estimación de orientación del cuadricóptero***
   1. **Convenciones respecto a los ángulos**

Para simplificar la implementación de los sistemas de control del cuadricóptero, se decidió representar la posición y velocidad angular del mismo en cada eje mediante ángulos de Euler, bajo la siguiente convención:

* Ángulo de Yaw: ángulo de giro respecto al eje Z del acelerómetro y giroscopio.
* Ángulo de Pitch: ángulo de giro respecto al eje Y del acelerómetro y giroscopio.
* Ángulo de Roll: ángulo de giro respecto al eje X del acelerómetro y giroscopio.
* Velocidad de Yaw: velocidad de giro respecto al eje Z del giroscopio.
* Velocidad de Pitch: velocidad de giro respecto al eje Y del giroscopio.
* Velocidad de Roll: velocidad de giro respecto al eje X del giroscopio.
  1. **Descripción de la Unidad de Medición Inercial (IMU)**

Se utilizó una tarjeta Pololu MinIMU-9 v2, la cual está constituida porun giroscopio de tres (3) ejes L3GD20, y por un acelerómetro de tres (3) ejes y un compás de tres (3) ejes LSM303DLHC. La misma provee de una interfaz que permite acceder a las mediciones, por eje, de cada uno de sus sensores, e incluye un regulador de voltaje y un convertidor de nivel de voltaje que permite su operación con una entrada de voltaje de 2,5V a 5,5V.



Ilustracion 4. Pololu minIMU-9 V2.

Fuente: <http://www.pololu.com/product/1268>

Para la obtención de datos de la tarjeta Pololu MinIMU-9 v2 se utilizaron las librerías diseñadas por el fabricante para su manejo desde tarjetas Arduino. A continuación se expone la configuración de los sensores utilizada:

|  |  |  |
| --- | --- | --- |
| **Sensor** | **Frecuencia de salida** | **Frecuencia corte filtro pasa-bajos** |
| **Giroscopio L3GD20** | 90Hz | 12,5Hz |
| **Acelerómetro LSM303DLHC** | 100Hz | 72,5Hz |

* 1. **Respuesta en frecuencia de la Unidad de Medición Inercial y filtrado de datos**

Se recogieron datos del giroscopio y del acelerómetro, con los motores apagados, y encendidos, y se encontró que las vibraciones del armazón se transmitían a la IMU, impidiendo obtener lecturas correctas sobre el estado del cuadricóptero.

Al realizar un estudio de sobre estos datos se encontraron que las vibraciones afectaban a todo el espectro de frecuencias y no habían modas significativas dentro de las muestras tomadas, por lo cual no se podía rechazar ninguna banda en particular.

**Gráfica que demuestre el ruido y espectro de frecuencias sin la base, en el acelerómetro y giroscopio. Motores apagados.**

**Gráfica que demuestre el ruido y espectro de frecuencias sin la base, en el acelerómetro y giroscopio. Motores encendidos**

Para disminuir las vibraciones que transfería el cuadricóptero a la IMU realizó una base compuesta por dos (2) componentes:

* **Moongel:** para reducir las vibraciones de alta frecuencia
* **Goma-espuma:** para de reducir las vibraciones de baja frecuencia.

Dicha base se coloco alrededor de la IMU para que funcionara de amortiguación entre el cuadricóptero y esta.

**Diagrama que ilustre cómo fue el montaje final de la IMU.**

Al recoger los datos con esta base se evidenció que la amplitud de las vibraciones disminuyó.

**Gráfica que demuestre el ruido y espectro de frecuencias con la base, en el acelerómetro y giroscopio.**

Al realizar el estudio se determino que la mayoría del ruido obtenido se encontraba en las altas frecuencias, por ende se implementó un filtro pasa bajo para el giroscopio y el acelerómetro de manera tal de que dejaran de pasar las señales con frecuencias superiores a la frecuencia de corte del filtro. Se pudo disminuir el error sistemático en las mediciones, reduciéndose su desviación estándar.

**Gráfica que demuestre el ruido y espectro de frecuencias con la base, y el filtro pasa bajos, en el acelerómetro y giroscopio.**

Aún después de aplicar los filtros pasa-bajos al acelerómetro y al giroscopio, se detectó una mayor cantidad de ruido en el acelerómetro, y, al realizar el análisis de datos del acelerómetro con los motores encendidos en un ángulo de inclinación cercano a cero (0) grados, se evidenció que la señal obtenida tenía una media cercana a cero, y una desviación estándar constante entre distintas pruebas. Se implementó un filtro de Kalman simple para mejorar la estimación de ángulo del cuadricóptero a partir del mismo, y se realizaron pruebas de estimación en un ángulo de inclinación de aproximadamente cero (0) grados.

**Gráfica que demuestre rendimiento de la estimación de ángulos con el Filtro de Kalman.**

En la tabla que se presenta a continuación se presentan los valores de media y desviación estándar de la señal, antes y después de implementar los filtros:

**Tabla de valores estadísticos de la señal en presencia de vibraciones, sin filtro, y con filtro.**

A partir de un análisis posterior, se determinó que el Filtro de Kalman implementado introducía una traslación en fase de \_\_\_\_ grados, por lo cual introduciría un retardo en la estimación, y reduciría el tiempo de respuesta de los sistemas de control.

**Gráfica que demuestre traslación en fase de la estimación de ángulos con el filtro de Kalman.**

1. **Estimación de altura del cuadricóptero**
   1. **Descripción del sensor de altura utilizado**

Para la estimación de altura del cuadricóptero se utilizó un sensor Parallax Ping, el cual opera con niveles lógicos TTL, provee un rango de alcance de entre 2cm y 3m, y un peso de 9 gramos. Adicionalmente, el sensor porta un LED en su cara frontal, que indica si el proceso de estimación de altura se está realizando de forma efectiva.



Ilustracion 4. Sensor ultrasónico Parallax Ping.

Fuente: <http://www.parallax.com/product/28015>

* 1. **Caracterización y respuesta en frecuencia del sensor de altura utilizado. Filtrado de datos.**

Se sometió el cuadricóptero a distintos ángulos de inclinación para medir la robustez del algoritmo de estimación de altura desarrollado. En la tabla que se presenta a continuación se presentan los parámetros del mismo:

**Tabla de valores estadísticos de la estimación de altura función de los ángulos de inclinación en Pitch y Roll.**

A partir de la realización de pruebas emulando las condiciones teóricas de posicionamiento e inclinación con el cuadricóptero en vuelo, se pudo constatar la inestabilidad de las mediciones del sensor utilizado, las cuales poseían una alta cantidad de discontinuidades al ser éste inclinado. Para solucionar dicho problema, y con ello realizar una estimación de altura más robusta y precisa se implementó un Filtro de Kalman de una variable, con una covarianza del error del proceso físico constante, no relacionándolo directamente al modelo físico del cuadricóptero. Los parámetros del Filtro de Kalman implementado se calcularon por prueba y error, aplicando como criterio la obtención de la mayor velocidad de respuesta y robustez posible en las estimaciones de altura.

**Gráfico de respuesta en frecuencia del sensor**

**Gráfico de las mediciones del sensor ultrasónico sin filtro y con filtro**

1. ***Comunicación inalámbrica***

Para poder mover el cuadricóptero y conocer los datos de telemetría desde una computadora fue necesario definir una comunicación entre ambos. Esta comunicación debe ser ligera, rápida, sencilla, de bajo consumo y compatible tanto para la PC como para la tarjeta arduino que se instaló en el cuadricóptero, por esta razón se eligió usar el protocolo ZigBee.

* 1. ***Descripción de las características y configuración de los módulos XBEE utilizados***

Para usar el protocolo ZigBee se utilizaron dos (2) tarjetas XBee “serie 1” y dos (2) tarjetas explorer, una para conectarse vía USB a la computadora y otra para comunicarse directamente al Arduino.

Ilustracion 4. Modulo XBee serie 1

Fuente: <http://examples.digi.com/wp-content/uploads/2012/06/1x-XBee-S1-300x300.jpg>

Especificaciones técnicas de los modulos XBee:

Tabla 4. Especificaciones del modulo XBee

|  |  |
| --- | --- |
| **Caracteristicas** | **ModuloXbee** |
| **Rendimiento** | |
| Rango con obstáculos | Hasta 100 pies (30 mts) |
| Rango sin obstaculos | Hasta 300 pies (90 mts) |
| Potencia de salida transmitida (seleccionable por software) | 1mW (0 dBm) |
| Tasa de datos en radio frecuencia | 250,000 bps |
| Frecuencia de datos por interfaz serial (seleccionable por software) | 1200 bps - 250 kbps (frecuencias sin estandar tambien son soportadas) |
| Sensibilidad de recepcion | -92 dBm (1% tasa de error de 1%) |
| **Requerimientos de poder** | |
| Voltaje de Entrada | 2.8 – 3.4 V |
| Corriente para enviar data (típica) | 45mA (@ 3.3 V) |
| Corriente para recibir data (típica) | 50mA (@ 3.3 V) |
| Corriente para apagarse | < 10 µA |
| **General** | |
| Frecuencia de operación | ISM 2.4 GHz |
| Dimensiones | 0,96” x 1,087” (2,438cm x 2,761cm) |
| Temperatura de operación | -40 to 85º C (industrial) |
| Opciones de antena | Integrated Whip, Chip or U.FL Connector, RPSMA Connector |
| **Redes y Seguridad** | |
| Topologias de red soportadas | Point-to-point, Point-to-multipoint & Peer-to-peer |
| Cantidad de canales (seleccionable por software) | 16 Direct Sequence Channels |
| Opciones de direccionamiento | PAN ID, Channel and Addresses |

Fuente: <https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf>

Las tarjetas Sparkfun *Explorer USB* y *Explorer regulated* permiten usar las tarjetas de XBee como una interfaz serial, por ende, facilita la programación para la comunicacion entre la PC y el cuadricóptero.

Ilustracion 4.  *XBee explorer para comuicacion serial*

Fuente: <https://cdn.sparkfun.com//assets/parts/7/1/1/5/11373-02.jpg>

** Ilustracion 4. *XBee explorer con puerto mini USB*

Fuente: <https://cdn.sparkfun.com//assets/parts/8/1/4/0/11812-03.jpg>

* 1. ***Descripción del protocolo de comunicación desarrollado***

Por razones de velocidad y simplicidad en la implementación se utilizaron los modulos XBEE en modo AT (transparent mode), el cual permite diseñar los paquetes de comunicación pudiendo así reducir su longitud al enviar datos en forma serial y diseñar todo el protocolo de comunicación aumentando la velocidad de transmisión de los mensajes.

Hay un error de comunicación entre el arduino nano y XBee, cuando se usa la frecuencia de 57600 baudios, de aproximadamente 3% lo que roza el máximo de tolerancia soportado por la UART(“***U****niversal****A****synchronous****R****eceiver-****T****ransmitter*” ,en español, *Transmisor-Receptor Asíncrono Universal)* por lo que genera muchos errores en la comunicación a esta frecuencia [Colton 2011 - 1], basados en este hecho se decidió utilizar una frecuencia de transmisión de 38400 (que es la tasa de transmisión estándar anterior a 57600 baudios).

Para reducir al mínimo el tamaño de los paquetes a enviar, se decidió codificar los datos a transmitirse por las interfaces inalámbricas en bytes, lo cual limitó el rango de valores a enviar mediante la interfaz diseñada, pudiendo enviarse datos de tipo entero dentro del rango numérico de cero (0) a doscientos cincuenta y cinco (255). Para los casos en que se necesitaba un mayor rango, se decidió dividir los paquetes según su signo, pudiéndose enviar, finalmente, datos dentro del rango numérico de los enteros, de cero (0) a quinientos doce (512).

Arquitectura del mensaje:

Para los mensajes se utilizó la siguiente arquitectura:

Tabla 4. Arquitectura de los mensajes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cantidad bytes** | 1 | 1 | 1-12 | 1 |
| **Contenido** | Cabecera del mensaje (255) | Código del mensaje | Contenido del mensaje | Checksum |

*Fuente: elaboración propia*

**La cabecera de mensaje:** es un identificador que indica cuando inicia un mensaje, siempre será el valor 255.

**El código de mensaje:** es el valor con el que se identifica cada tipo de mensaje.

**Contenido del mensaje:** es el que contiene la data útil dependiendo del mensaje, su longitud varia de 1 a 12 bytes dependiendo del tipo de mensaje.

**Checksum:** se utiliza a manera de validar que el mensaje llegue correctamente, se genera haciendo una operación XOR a todos los bytes anteriores de ese mensaje.

Para este TEG se diseñaron cuatro (4) tipos de mensajes diferentes, dos (2) mensajes que van del cuadricóptero a la PC y dos (2) mensajes que van desde la PC al cuadricóptero y son los siguientes:

**Mensaje de estado:** es el paquete que contiene la información de telemetría del cuadricóptero y su contenido está compuesto por 12 bytes, los cuales indican posiciones y velocidades para los tres (3) ejes del cuadricóptero con respecto a un eje fijo imaginario, además tiene información sobre la altura y si los motores se encuentran encendidos o apagados.

Debido a que tanto la posición como la velocidad pueden ser negativas o positivas y el mayor valor que se puede enviar en un byte es doscientos cincuenta y cinco (255) se decidió separar los datos cuyo rango de valores superara este valor.



Ilustracion 4. Diagrama de movimientos del cuadricóptero y signo que toman los valores de aceleración y posición.

Fuente: <http://3.bp.blogspot.com/-B8IImJJHUl0/U0cj-KWq95I/AAAAAAAAAIg/kaMEopdgCwc/s1600/Screen+Shot+2014-04-10+at+4.06.09+PM.png>

* Para el eje yaw la velocidad angular puede variar de -255 a 255 y la posición puede variar entre -180 y 180 grados
* Para el eje pitch la velocidad angular puede variar de -255 a 255 y la posición puede variar entre -90 y 90 grados.
* Para el eje roll la velocidad angular puede variar de -255 a 255 y la posición puede variar entre -90 y 90 grados.
* La altura siempre será un valor positivo debido a que representa la distancia entre el cuadricóptero y el piso, su máximo valor es de 4 metros.
* El estado de los motores indica si estos están encendidos (0) o apagados (1).

**Mensaje de recibido:** es el mensaje que envía el cuadricóptero a la pc para indicar que se ha recibido un mensaje y lo que contiene es el código del mensaje recibido.

**Mensaje de motores:** es el comando que envía la PC al cuadricóptero para apagar o encender los motores.

**Mensaje de movimiento:** es el comando que se envía para mover al cuadricóptero en el eje roll (derecha o izquierda), Pitch (adelante o atrás) o altura (para subir y bajar el cuadricóptero).

Tabla 4. Contenido y codigo de cada tipo de mensaje

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tipo de mensaje** | **Estado** | **Telemetría completa** | | **Acknowledge** | **Encendido de motores** | **Movimiento** |
| **Codigo de mensaje** | 6 | 8 | | 7 | 1 | 2 |
| **Contenido del mensaje** | Posición YAW positiva | Estimado | Posición YAW positiva | Código mensaje recibido | Para apagar motores: 0           Para encender motores: 1 | Movimiento Eje Pitch |
| Posición YAW negativa | Posición YAW negativa |
| Posición PITCH (de 0 a 180 siendo el valor 90 igual a 0 grados) | Posición PITCH positiva |
| Posición ROLL (de 0 a 180 siendo el valor 90 igual a 0 grados) | Posición PITCH negativa |
| Velocidad YAW positiva | Posición ROLL positiva | Movimiento Eje Roll |
| Velocidad YAW negativa | Posición ROLL negativa |
| Velocidad PITCH positiva | Bruto | Velocidad YAW positiva |
| Velocidad PITCH Negativa | Velocidad YAW negativa |
| Velocidad ROLL positiva | Velocidad PITCH positiva | Movimiento de altura |
| Velocidad ROLL negativa | Velocidad PITCH positiva |
| Altura | Velocidad ROLL positiva |
| Estado Motores | Velocidad ROLL positiva |
|  | Filtrado | Velocidad YAW positiva |  |
| Velocidad YAW negativa |
| Velocidad PITCH positiva |
| Velocidad PITCH negativa |
| Velocidad ROLL positiva |
| Velocidad ROLL negativa |
| Bruto | Aceleracion YAW positiva |
| Aceleracion YAW positiva |
| Aceleracion PITCH positiva |
| Aceleracion PITCH positiva |
| Aceleracion ROLL positiva |
| Aceleracion ROLL positiva |
| Filtrado | Aceleracion YAW positiva |
| Aceleracion YAW negativa |
| Aceleracion PITCH positiva |
| Aceleracion PITCH negativa |
| Aceleracion ROLL positiva |
| Aceleracion ROLL negativa |
| Bruto | Altura |
| Filtrado | Altura |
| Estimado | Velocidad Z positiva |
| Velocidad Z negativa |
| PWM motor delantero | |
| PWM motor trasero | |
| PWM motor derecho | |
| PWM motor izquierdo | |

Fuente: elaboración propia

* Xbee: para poder probar los modulos XBee y configurarlos fue necesario conectarlos al “*USB XBee explorer”* el cual se conectaba la computadora por medio de un cable USB. Se descargaron los drivers para el modulo de XBee y se configuro usando el programa “XCTU” de la compañía “Digi international”.

Ilustración 4. Comunicación entre PC y Arduino

Fuente: elaboración propia

Posteriormente se probó la comunicación entre los módulos XBee conectando uno al arduino a través del “XBee explorer regulated” y el otro a la computadora usando el “XBee usb explorer”.

Se realizaron hicieron pruebas de distancia para medir la tasa de mensajes erróneos que se recibían a medida que se alejaban los modulos con la configuración de comunicación seleccionada.

Tabla 4. 5 Pruebas realizadas con el XBee a diferentes distancias, cada prueba envía un total de 200 mensajes.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| distancia | prueba 1 | prueba 2 | prueba 3 | media recibidos | % recibidos | % sin recibir |
| recibidos | recibidos | recibidos |
| 0 | 200 | 200 | 200 | 200 | 100 | 0 |
| 1 | 200 | 199 | 200 | 200 | 99.83333333 | 0.166666667 |
| 3 | 199 | 200 | 199 | 199 | 99.66666667 | 0.333333333 |
| 5 | 197 | 194 | 200 | 197 | 98.5 | 1.5 |
| 7 | 196 | 172 | 198 | 189 | 94.33333333 | 5.666666667 |
| 10 | 148 | 200 | 191 | 180 | 89.83333333 | 10.16666667 |

Fuente: elaboración propia

Ilustracion 4. Porcentaje de mentajes recibidos en distancia

Fuente: elaboración propia

1. ***Sistemas de control***
   1. **Arquitectura de control propuesta**

Se diseñó una arquitectura de control multi-capa para la regulación en vuelo de la posición angular, velocidad angular, velocidad en el eje z, y posición en el eje z (altura) del mismo, mediante algoritmos de control Proporcional-Integral-Derivativo. La arquitectura de control propuesta se describe a continuación:



Ilustración 4. Arquitectura de control propuesta

Fuente: elaboración propia.

La arquitectura de control propuesta se desglosa en:

1. Sistema de control de posición en el eje z (Altura).
   1. Entrada: estimación de altura del cuadricóptero.
   2. Referencia: altura deseada, especificada por el usuario en tiempo real.
   3. Salida: velocidad lineal en z para alcanzar la altura deseada.
   4. Intervalo de ejecución: cada 30 milisegundos.
2. Sistema de control de velocidad lineal en el eje z.
   1. Entrada: velocidad lineal del cuadricóptero.
   2. Referencia: velocidad lineal en z deseada.
   3. Salida: corrección de altura.
   4. Intervalo de ejecución: cada 30 milisegundos.
3. Sistema de control de posición angular de Pitch.
   1. Entrada: ángulo de Pitch del cuadricóptero.
   2. Referencia: ángulo de Pitch deseado, especificado por el usuario en tiempo real.
   3. Salida: velocidad angular para alcanzar la posición de Pitch deseada.
   4. Intervalo de ejecución: cada 20 milisegundos.
4. Sistema de control de posición angular de Roll.
   1. Entrada: ángulo de Roll del cuadricóptero.
   2. Referencia: ángulo de Roll deseado, especificado por el usuario en tiempo real.
   3. Salida: velocidad angular para alcanzar la posición de Roll deseada.
   4. Intervalo de ejecución: cada 20 milisegundos.
5. Sistema de control de velocidad angular de Pitch.
   1. Entrada: velocidad angular del cuadricóptero en el eje de Yaw.
   2. Referencia: velocidad angular de Pitch deseada, obtenida como salida del sistema de control de posición angular de Pitch.
   3. Salida: corrección de Pitch.
   4. Intervalo de ejecución: cada 10 milisegundos.
6. Sistema de control de velocidad angular de Roll.
   1. Entrada: velocidad angular del cuadricóptero en el eje de Yaw.
   2. Referencia: velocidad angular de Roll deseada, obtenida como salida del sistema de control de posición angular de Pitch.
   3. Salida: corrección de Pitch.
   4. Intervalo de ejecución: cada 10 milisegundos.
7. Sistema de control de velocidad angular de Yaw.
   1. Entrada: velocidad angular del cuadricóptero en el eje de Yaw.
   2. Referencia: velocidad angular = 0, constante.
   3. Salida: corrección de Yaw.
   4. Intervalo de ejecución: cada 10 milisegundos.
   5. ***Simulación***

El modelo linealizado del cuadricóptero fue implementado en el software MATLAB, y se realizaron simulaciones para verificar la factibilidad de realizar el control de posición angular, velocidad angular y altura del cuadricóptero mediante la arquitectura de control propuesta. Las simulaciones se llevaron a cabo mediante el método de Euler, ya que se partía del modelo linealizado del cuadricóptero, y el algoritmo Proporcional-Integral-Derivativo modela la retroalimentación en lazo cerrado como una ecuación diferencial lineal.

**Diagrama que ilustre cómo se realizó la simulación.**

Para ajustar el rendimiento de los sistemas de control implementados, se calcularon los autovalores de la matriz de transición del sistema:

**Gráficas que ilustren la ubicación de los autovalores de la matriz de transición de estados en cada eje.**

El cálculo de los parámetros de ejecución de los algoritmos PID se realizó de la siguiente manera:

**Diagrama que ilustre cómo se realizó el cálculo de las constantes a partir de los autovalores.**

Se obtuvo un rendimiento satisfactorio de la arquitectura de control propuesta, con lo cual se verificó la efectividad de la misma. A continuación se exponen los resultados obtenidos en la ejecución de los sistemas de control de posición angular, velocidad angular y altura (posición y velocidad lineal en el eje z):

**Gráficas que ilustren el rendimiento de los sistemas de control.**

* 1. ***Implementación en el cuadricóptero***

Se definió una velocidad base constante, para mantener homogeneidad en el comportamiento dinámico del cuadricóptero durante el proceso de calibración, pruebas, y vuelo. Las señales de Módulación por Ancho de Pulsos (PWM) a enviar a los motores se calcularon de la siguiente manera:

Para la implementación de los sistemas de control se utilizó la librería PID de Arduino, la cual permite encapsular en una clase todos los atributos y métodos suficientes para la ejecución de un sistema de control Proporcional-Integral-Derivativo. El cálculo de las constantes de los sistemas de control se realizó mediante prueba y error.

Se realizaron pruebas con la arquitectura de control propuesta, en las cuales no se obtuvo un resultado exitoso por la inestabilidad de la estimación de posición angular del cuadricóptero producida por las vibraciones del chásis. A consecuencia de ello, tampoco se pudo obtener un rendimiento adecuado de los sistemas de control de altura, ya que la estimación de altura del cuadricóptero se realiza en base a los ángulos de inclinación del mismo.

**Gráfica que demuestre rendimiento de los sistemas de control de posición angular y altura.**

Se calibró y probó el sistema de control de velocidad angular del cuadricóptero, el cual brindó un rendimiento satisfactorio, cancelando la dinámica de rotación del cuadricóptero en vuelo.

**Gráfica que demuestre rendimiento de los sistemas de control de velocidad angular para llevar al cuadricóptero a velocidad 0.**

**Gráfica que demuestre rendimiento de los sistemas de control de velocidad angular para llevar al cuadricóptero a una velocidad deseada X.**

Para la realización de pruebas de vuelo, se modificó la arquitectura de control propuesta inicialmente, para que el usuario pudiera emitir consignas de velocidad angular y velocidad común a los cuatro motores en tiempo real (Tarea que debían realizar los sistemas de control de posición angular y altura en la arquitectura propuesta). La arquitectura final implementada se presenta a continuación:

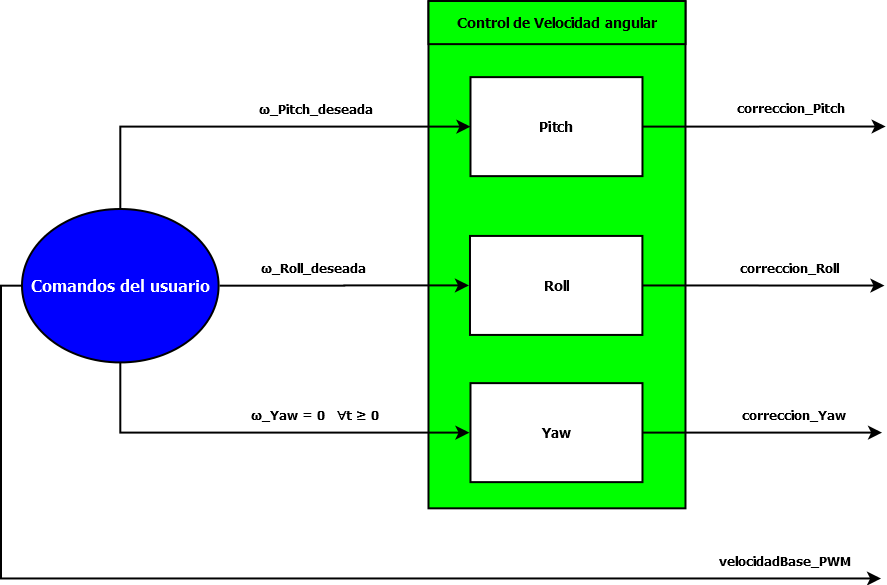


Ilustración 4. Arquitectura de control implementada

Fuente: elaboración propia.

Se obtuvo un rendimiento satisfactorio del cuadricóptero en vuelo, con la arquitectura final implementada, aunque la misma requirió de un lapso de práctica, ya que requiere una realimentación constante por parte del usuario para ajustar la inclinación y altura del cuadricóptero en vuelo.

**Gráfica que demuestre rendimiento de los sistemas de control de velocidad angular del cuadricóptero en vuelo.**

1. ***Plataforma de pruebas***
   1. ***Software de telemetría y comandos***

El software de telemetría y comandos se desarrolló como una serie de paquetes para ROS, que llevan a cabo de forma independiente la transmisión de datos entre el cuadricóptero y la PC, la obtención de comandos del usuario mediante un control Logitech Gamepad II y la exportación de los mensajes de estado del cuadricóptero a formato CSV. La codificación de los mismos se realizó en el lenguaje de programación Python, haciendo uso de la librería rospy, y siguiendo los estándares de programación de ROS.

**Captura de pantalla de rqt\_graph que muestre todos los paquetes desarrollados en ejecución, y el paso de mensajes entre ellos.**

Para realizar la comunicación entre los paquetes de software en tiempo de ejecución se definieron formatos de mensaje personalizados, los cuales se describen a continuación:

Al realizarse la comunicación entre los paquetes de forma asíncrona, como es natural en los paquetes de software desarrollados para ROS, la plataforma de software desarrollada tuvo un muy bajo acoplamiento entre sus componentes, lo cual aseguró una alta robustez del sistema ante fallas.

**Tabla con los formatos de mensaje creados, y los paquetes que los usan.**

A continuación se describen los paquetes de software desarrollados:

1. **comunicación\_serial**: Al iniciar su ejecución instancia un objeto de la clase handler\_serial, el cual encapsula el manejo del puerto serial, y permite al programa principal realizar llamados simples a funciones para llevar a cabo el envío y recepción de mensajes de encendido o apagado, estado, y comandos de movimiento. Posteriormente, son inicializadas rutinas de subscripción a los tópicos encendido\_cuadricoptero y movimientos\_cuadricoptero, y rutinas para publicación en el tópico estado\_cuadricoptero.

Durante su ejecución, el nodo del paquete publica todos los mensajes de estado recibidos por el objeto handler\_serial, y realiza el envío de los comandos de encendido o apagado, y y de movimiento. Si se finaliza su ejecución de forma abrupta, o se desconecta el cable de comunicación del puerto serial, el nodo automáticamente detiene la comunicación y libera el puerto serial como recurso del sistema.

El manejo del puerto serial en la clase handler\_serial se realiza mediante la librería pyserial, el objeto Serial, y sus métodos correspondientes.

1. **logitech\_rumblepad\_ii:** Al iniciar su ejecución instancia un objeto de la clase handler\_joystick, el cual encapsula la detección y manejo de los eventos del control. Igualmente, el nodo inicializa rutinas para la publicación de los comandos del usuario en los tópicos encendido\_cuadricoptero y movimientos\_cuadricoptero.

Durante su ejecución el nodo del paquete verifica constantemente la emisión de comandos por parte del usuario, y en caso de haber sido emitido un comando desde el control, el nodo publica dicho comando al tópico que corresponda. Si se finaliza su ejecución de forma abrupta, o se desconecta el cable del control, el nodo automáticamente detiene la publicación de mensajes y libera el control como recurso del sistema.

El manejo del puerto serial en la clase handler\_joystick se realiza mediante la librería pygame, el objeto Joystick, y sus métodos correspondientes.

Ilustracion 4. Diagrama del botones para el control

Fuente: Elaboracion propia

1. **exportador\_estado\_csv:** El nodo consta de una rutina de subscripción al tópico estado\_cuadricoptero, que escribe todos los mensajes publicados en el mismo en un archivo en formato CSV (Siguiendo las pautas de codificación de archivos CSV para su lectura con MATLAB).
2. **exportador\_telemetria\_csv:** El nodo consta de una rutina de subscripción al tópico telemetría\_total, que escribe todos los mensajes publicados en el mismo en un archivo en formato CSV (Siguiendo las pautas de codificación de archivos CSV para su lectura con MATLAB).

No se desarrolló un paquete para la visualización de datos obtenidos mediante telemetría por considerarse que los paquetes para visualización de gráficas en tiempo real para ROS existentes (rxplot y rqt\_plot) brindan un rendimiento óptimo y una variedad de opciones de visualización adecuadas para la visualización del estado del cuadricóptero en tiempo real.

**Imagen que ilustre cómo es la visualización de datos con rpt\_plot, en la que se vean velocidades y posiciones angulares con distinta amplitud.**

Finalmente, en el diagrama que se presenta a continuación se describe a detalle el flujo de ejecución del software desarrollado:

**Poner un diagrama de flujo del programa, que sea sencillo pero muy explicativo.**

1. ***Scripts para análisis de respuesta en frecuencia de los sensores***

Se desarrollaron una serie de scripts de MATLAB para realizar el análisis de respuesta en frecuencia de los datos crudos y filtrados de los sensores, a partir de archivos CSV generados por los nodos exportador\_estado\_csv y exportador\_telemetria\_csv, desarrollados como parte de la plataforma de software de telemetría y comandos.

Los scripts desarrollados aplican un algoritmo de transformada de Fourier discreta sobre los datos de los sensores, realizan la traslación de los mismos en el dominio de la frecuencia, y muestran al usuario gráficas de los datos obtenidos en el dominio del tiempo y de la frecuencia, para su análisis

**Captura de las pantallas de visualización de datos de los sensores, en el dominio del tiempo y en el dominio de la frecuencia.**

* + 1. ***Montaje para la ejecución de pruebas en un solo eje del cuadricóptero***

Una vez comprobado el montaje de todos los componentes sobre el cuadricóptero se procedió a trabajar con el algoritmo de estabilización de manera independiente para los ejes roll y pitch, esto se logro utilizando una base de madera la cual restringía el movimiento del cuadricóptero a solo uno de los ejes que se iban a estabilizar.

Además se aprovecho para probar cambiar el angulo deseado sobre el cuadricóptero de manera que al volar sin restricción pueda hacer los movimientos necesarios para ir adelante, atrás, derecha o izquierda.



Ilustración 4. Ejemplo de base utilizada para restringir el movimiento del cuadricóptero en un eje

Fuente: elaboración propia



Ilustracion 4. Referencia del sistema para un solo eje

Fuente: <http://4.bp.blogspot.com/-bKrD54s3Kr4/UWPHr-gzLJI/AAAAAAAAAEA/-YTQr6I2tXk/s1600/PID+Need.bmp>

1. ***Montaje para la ejecución de pruebas en vuelo restringido***

Una vez estabilizados los ejes de roll y pitch de forma independiente se empezó a calibrar el eje yaw, para esto se sujeto al cuadricóptero de una cuerda, la cual fue amarrada al centro de la estructura, y se colgó en un lugar suficientemente alto y amplio como para que pudiera moverse sin chocar contra otros objetos.

Además se pudo probar el ajuste de roll y pitch trabajando de forma simultánea y la robustez del sistema de control frente a perturbaciones del ambiente.

Durante estas pruebas el ciclo de trabajo base estuvo por debajo del 70% para evitar que despegase y se enredara con la cuerda.

Ilustracion 4. Diagrama de movimiento sobre el eje yaw.

*Fuente:* [*http://blog.oscarliang.net/wp-content/uploads/2013/06/quadcopter-rotating.png*](http://blog.oscarliang.net/wp-content/uploads/2013/06/quadcopter-rotating.png)