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.

Para realizar la conmutación a alta velocidad de cada motor en un solo sentido, se utilizó un MOSFET IRFZ44N y un diodo de rodada libre, para forzar la descarga de la inductancia del motor al cerrar el canal del MOSFET. Estos fueron seleccionados de entre todos los componentes disponibles en el mercado venezolano por su alta velocidad de conmutación, alta tolerancia y estabilidad ante valores altos de corriente y voltaje, y baja resistencia drenador-surtidor en el caso del IRFZ44N. Se realizaron pruebas para caracterizar la conductividad y resistencia interna del MOSFET ante distintos valores de voltaje aplicados en la compuerta del mismo, al encender uno de los motores del cuadricóptero de forma continua (Ciclo de trabajo de 100% con un voltaje Vgs fijo), con una alimentación de 12V -cercana al voltaje máximo de carga de la batería de polímero de litio seleccionada para alimentar el conjunto-. A continuación se presenta el diagrama del circuito de prueba utilizado, y las curvas características identificadas a partir de los datos recogidos:



Ilustracion 4. Diagrama de circuito para estimación de Vgs óptimo para conmutar MOSFET con carga de motor DC en el surtidor.

Fuente: elaboración propia

Ilustracion 4. Relación **Voltaje de compuerta** – **Corriente drenador-surtidor** del MOSFET IRFZ44N, en presencia de carga del motor.

Fuente: elaboración propia.

Ilustracion 4. Relación **Voltaje de compuerta** - **Corriente drenador-surtidor** del MOSFET IRFZ44N, en presencia de carga del motor.

Fuente: elaboración propia.

A partir del análisis realizado se identificó el rango de valores entre 4,5 y 6 voltios como el rango de valores de tensión de compuerta óptimos para el funcionamiento del MOSFET con los motores del cuadricóptero como carga en el drenador. Considerando el efecto de descarga de la batería a utilizar, se decidió utilizar un voltaje de al menos 6V, siendo éste cota superior del rango de valores óptimos identificado, como tensión de activación de la compuerta del MOSFET IRFZ44N.

Para disminuir los efectos de ruido electromagnético que pudieran ser introducidos por la rápida conmutación de los MOSFETS y los motores al módulo de lógica, sensores y comunicación, el cual envía las señales de PWM para la regulación de velocidad de los motores, se decidió utilizar opto-acopladores para separar totalmente la etapa. Se seleccionó el modelo 4N26, presente en el mercado de componentes electrónicos venezolano, por su alta velocidad de conmutación, tolerancia a altos valores de voltaje y corriente, bajo precio, y simplicidad de configuración. Para obtener la tensión de salida de 6V para conmutar la compuerta del MOSFET IRFZ44N se diseñó un circuito que hace uso de un divisor de voltaje en el emisor del fototransistor, en configuración colector común, como se presenta a continuación:

****

Ilustracion 4. Diagrama de circuito para estimación de carga total de foto-transistor del opto-acoplador 4N26 en configuración colector común.

Fuente: elaboración propia.

Para calcular la resistencia total de la carga del colector común del fototransistor se evaluó el tiempo de respuesta del mismo ante impulsos de PWM de 490 Hz, y se seleccionó un valor de 900Ω, a partir del cual se realizó el cálculo de las resistencias R1 y R2 para el circuito de conmutación con divisor de voltaje. El valor calculado para R1 fue de 390Ω, y para R2 fue de 510Ω.



Ilustracion 4. Señal de salida del opto-acoplador ante una señal de PWM con un ciclo de trabajo de 98%.

Fuente: elaboración propia.



Ilustracion 4. Señal de salida del opto-acoplador ante una señal de PWM con un ciclo de trabajo de 23%.

Fuente: elaboración propia.



Ilustracion 4. Diagrama de circuito de regulación de velocidad diseñado.

Fuente: elaboración propia.

Finalmente, se diseñó y construyó una placa de circuito impreso de una sóla capa consistente en un arreglo de cuatro módulos para regulación de velocidad desarrollados en el presente trabajo. A continuación se muestra el diseño y el acabado del circuito luego se ser ensamblado:



Ilustracion 4. Diseño de circuito impreso del módulo de alimentación y control de motores.

Fuente: elaboración propia.

****

Ilustracion 4. Tarjeta de circuito impreso del módulo de alimentación y control de motores con conector XT-60 para baterías LiPo.

Fuente: elaboración propia.

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 módulo 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 de circuito del módulo 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.

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: se utilizó la configuración por defecto para el acelerómetro (Tasa de salida de 50Hz y filtro pasa-bajos de 37Hz) y el giroscopio (Tasa de salida de 90Hz, y filtro pasa-bajo de 12,5Hz), por estar en concordancia con la frecuencia de ejecución de los algoritmos de control del cuadricóptero.



Ilustracion 4. Pololu minIMU-9 V2.

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

* 1. **Respuesta en frecuencia de la Unidad de Medición Inercial y filtrado de datos**
  2. **Cálculo de velocidad angular**

El cálculo de velocidad angular se realizó a partir de las mediciones realizadas con el giroscopio. Tomando en cuenta la hoja de datos del sensor L3GD20, el mismo puede configurarse para obtener una sensibilidad de 8,75, 17,5 y 70 milésimas de grado por segundo por cada dígito de medición obtenido (mdps/digit – en inglés *millidegrees per second per digit*), y un rango de medición de 250, 500 y 2000 grados por segundo (dps – en inglés *degree per second*). Se decidió configurar el rango de medición a 250 grados por segundo y la senbilidad del sensor a 8,75 mdps/digit, por considerarse rango y senbilidad suficientes para medir las velocidades del cuadricóptero realizando movimientos simples en vuelo. En base a esto último se calculó la ganancia del giroscopio, para convertir todas las mediciones obtenidas mediante el mismo, en milésimas de grado por segundo, a grados por segundo, como se ilustra a continuación:

siendo:

* 1. **Estimación de ángulos de Pitch y Roll a partir del acelerómetro**

Se realizó un estimado del ángulo de inclinación del cuadricóptero a partir de las mediciones del acelerómero, el cual provee una descomposición de la fuerza de aceleración del cuadricóptero en tres (3) ejes perpendiculares (x, y, z). El acelerómetro puede detectar constantemente la fuerza de gravedad, en magnitud, dirección, y fuerza, en cada uno de sus tres (3) ejes, y en base a ello puede establecerse un marco de referencia absoluto a partir del cual calcular los ángulos de Pitch y Roll del sensor, y en consecuencia, del cuadricóptero. La estimación de ángulos se realizó siguiendo el procedimiento expuesto en [STMicroElectronics 2010], el cual es presentado a continuación:



Ilustracion 4. Modelo de estimación de angulos

Fuente: [STMicroElectronics]

siendo:

Las estimaciones de los ángulos de Pitch y Roll calculadas a partir de los datos del acelerómetro, a pesar de ser precisas y permitir mantener un marco de referencia absoluto, en base a la fuerza de gravedad de la tierra, presentan un alto porcentaje de ruido, ya que el acelerómetro es altamente sensible a perturbaciones provocadas por fuerzas externas que incidan sobre el mismo.

**Demostración del ruido del acelerómetro mediante una gráfica**

**Gráfica del espectro de frecuencias del acelerómetro**

* 1. **Estimación de posición angular a partir del giroscopio**

Se realizó un estimado del ángulo de inclinación del cuadricóptero, mediante integración numérica de las velocidades de rotación de Yaw, Pitch y Roll, como se describe a continuación:

Durante un lapso corto de tiempo, este estimado del ángulo de inclinación en cada eje puede mantenerse preciso, pero, tiende a presentar deriva constante y a alejarse de los valores reales a medir, por no realizarse la misma respecto a un marco de referencia absoluto.

**Gráfica que demuestre la deriva en el tiempo del estimado de ángulo mediante integración numérica de datos del giroscopio**

* 1. **Combinación de las estimaciones de posición angular del acelerómetro y giroscopio**

Al combinar la precisión del acelerómetro para medir inclinación respecto al marco de referencia absoluto del planeta Tierra, con la sensibilidad y estabilidad de la estimación de ángulo realizada a partir de los datos del giroscopio para medir los movimientos de rotación alrededor de cada eje del sensor, puede obtenerse una estimación de ángulo precisa, estable, y de alta sensibilidad.

Como es descrito en [Burgard 2005],si es el estado estimado de un proceso, y una observación acerca del estado del mismo, puede aplicarse la Regla de Bayes para estimar la probabilidad de que el proceso se encuentre en el estado X a partir de Z:

En base a [Sturm 2013] y [Burgard 2005], por la Ley de Probabilidad Total, el término puede ser tratado como una constante de normalización , tal que:

Por lo cual:

Para el caso particular del presente trabajo en el que se realizó estimación de posición angular mediante la combinación de datos de los sensores acelerómetro y giroscopio, por lo cual se cuenta con observaciones y , por lo cual el modelo se reduce a:

Se asume además que:

+ )

+ )

Siendo el error de estimación a partir de las observaciones de ambos sensores. Luego:

Finalmente, se decidió utilizar una aproximación del ángulo bajo la suposición de que el error de estimación supone una cantidad despreciable:

+

El modelo derivado a partir de teoría de probabilidades para la combinación de las estimaciones de ángulo del giroscopio y acelerómetro, coincide con el de un filtro digital de uso altamente extendido conocido como *Filtro Complementario* [Gaydou 2007], [Colton 2007], el cual se fundamenta en la combinación de un filtro pasa bajos y un filtro pasa altos, ambos de primer orden, para la composición de los espectros de frecuencias de dos señales lineales invariantes en el tiempo en una tercera señal de salida. El filtro complementario puede ser descrito mediante las siguientes ecuaciones en el dominio de la frecuencia [Gaydou 2007]:



Ilustracion 4. Modelo del Filtro complementario

Fuente: [Gaydou 2007]

En donde:

La ganancia del filtro pasa altos puede ser descrita a partir de la siguiente ecuación:

Donde:

constante de tiempo del filtro.

diferencial de tiempo.

Tanto el modelo probabilístico como el modelo basado en el análisis de señales del proceso de estimación de ángulo coinciden en la relación fundamental que establecen entre la estimación final de ángulo, y la estimación del giroscopio y acelerómetro, por lo cual se puede considerar que, permiten analizar la influencia de los parámetros y sobre el espectro de frecuencias de las estimaciones de ángulos como funciones de tiempo continuo. Finalmente, que se cumple la siguiente relación: Mejorar

El cálculo de la ganancia del filtro pasa altos, y con ello, de y , se realizó mediante prueba y error, teniendo como criterio de selección la estabilidad, precisión y exactitud de la estimación de ángulo realizada, y el tiempo de respuesta de la misma. Se estableció un valor de k=0,03 para la ganancia, el cual fue utilizado para las pruebas de vuelo del cuadricóptero.

**Gráfica que demuestre precisión y estabilidad del ángulo calculado con el filtro complementario**

**Gráfica que demuestre estabilidad del estimado de ángulo ante ruido, por ejemplo, con el cuadricóptero inmovilizado, pero con las hélices girando a alta velocidad.**

Es importante recalcar que por las características de funcionamiento del acelerómetro, y el rango de la función , el procedimiento de estimación de ángulos de Pitch y Roll presentado sólo permite la aproximación de los mismos en un rango entre menos noventa y noventa grados, sin poder detectar si el cuadricóptero se encuentra en un ángulo fuera de ese rango o volteado. No obstante, el algoritmo desarrollado satisface las necesidades del proyecto realizado en el presente trabajo de investigación.

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. **Obtención de datos y cálculo de altura.**

Para la obtención de datos del sensor ultrasónico de distancia se utilizó la librería NewPing V1.3 para Arduino, la cual permite el manejo del sensor mediante la encapsulación del mismo como un objeto de la clase NewPing, y basa su funcionamiento en la emisión de un pulso del sensor, y la ejecución de una rutina para detección y medición del tiempo de retorno del pulso ultrasónico emitido. La librería NewPing ofrece una mayor precisión que el método de detección de pulsos utilizado por defecto en Arduino mediante la función pulseIn(), y no interfiere en la ejecución del resto del código, ya que el proceso de detección de pulsos se ejecuta como una rutina de interrupción sobre el temporizador timer2 de las tarjetas Arduino.

La estimación de distancia a partir de las mediciones del sensor ultrasónico se realiza a partir del tiempo de retorno de los pulsos emitidos, como se expone a continuación:

Donde:

velocidad del sonido, constante, de valor 0,0343 cm/µs.

: yiempo de retorno del pulso, medido en microsegundos (µs).

* 1. **Respuesta en frecuencia del sensor de altura utilizado y filtrado de datos**

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 optimización se utilizaron los modulos de comunicación en modo AT (transparent mode), el cual permite disenar 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).

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** | **Codigo de mensaje** | **Contenido del mensaje** | | | | | | | | | | | | |
| Mensaje de estado | 6 | Posición YAW positiva | Posición YAW negativa | Posición PITCH (de 0 a 180 siendo el valor 90 igual a 0 grados) | Posición ROLL (de 0 a 180 siendo el valor 90 igual a 0 grados) | Velocidad YAW positiva | Velocidad YAW negativa | Velocidad PITCH positiva | Velocidad PITCH Negativa | | Velocidad ROLL positiva | Velocidad ROLL negativa | Altura | Estado Motores |
| Mensaje de recibido | 7 | Código mensaje recibido | | | | | | | | | | | | |
| Mensaje de motores | 1 | * Para apagar motores: 0 * Para encender motores:1 | | | | | | | | | | | | |
| Mensaje de movimiento | 2 | Movimiento Eje Roll | | | | Movimiento Eje Pitch | | | | Movimiento altura | | | | |

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”.

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.

Ilustracion 4. Porcentaje de mentajes recibidos en distancia

Fuente: elaboración propia

Ilustración 4. Comunicación entre PC y Arduino

Fuente: Elaboración propia

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

1. ***Sistemas de control***
   1. ***Modelo físico? Simulación? Controlabilidad?***

**Modelo físico de la tesis de Ian Cowling? (Está en referencias). A partir de ello se puede simular y demostrar controlabilidad del cuadricóptero.**

* 1. **Arquitectura y descripción**

Con el propósito de estabilizar el cuadricóptero, se diseñó una arquitectura de control multi-capa para la regulación de la posición angular, velocidad angular y altura del mismo en vuelo, mediante algoritmos de control Proporcional-Integral-Derivativo.

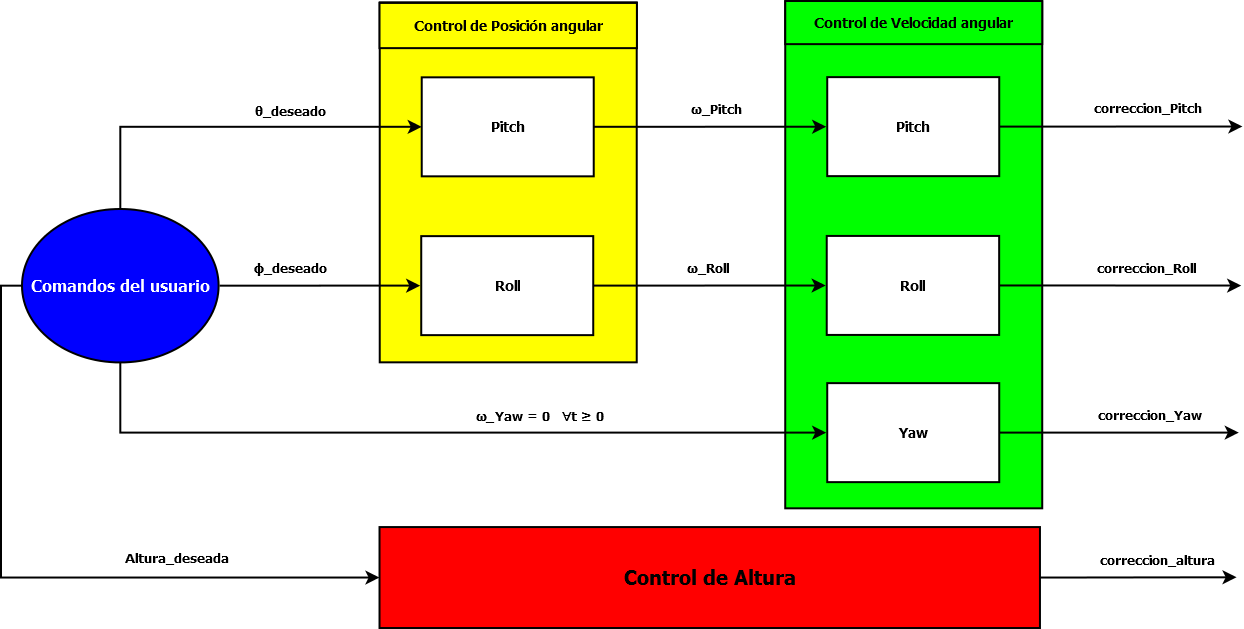


Ilustración 4. Arquitectura de control desarrollada

Fuente: elaboración propia.

Los sistemas de control desarrollados se presentan a continuación:

1. Sistema de control de altura.
   1. Entrada: estimación de altura del cuadricóptero.
   2. Referencia: altura deseada, especificada por el usuario en tiempo real.
   3. Salida: corrección de altura.
   4. Intervalo de ejecución: cada 50 milisegundos.
2. 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 de Pitch deseada.
   4. Intervalo de ejecución: cada 20 milisegundos.
3. 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 de Roll deseada.
   4. Intervalo de ejecución: cada 20 milisegundos.
4. 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 5 milisegundos.
5. 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 5 milisegundos.
6. 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 5 milisegundos.

**¿Por qué la arquitectura es así? Explicar bien. Depende del modelo?**

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.

* 1. ***Pruebas realizadas***

1. ***Plataforma de pruebas***
   1. ***Scripts para análisis de respuesta en frecuencia de los sensores***
   2. ***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.

**Tabla que describa para qué se programó cada botón del control**

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).

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. ***Montaje físico***
   * 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 por el centro de su cuerpo y se colgó en un lugar alto y amplio para que pudiera moverse sin chocar contra otro objeto.

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>