PLATAFORMA DE DESARROLLO PARA CONTROL COOPERATIVO Y ESTABILIDAD EN CUADROTORES.

Proyecto de Trabajo de Grado

Juan Pablo Jiménez
Michael Levkovsky

Director: Ing. Rodrigo Escobar M.S.

Pontificia Universidad Javeriana Bogotá
Facultad De Electrónica
2013

1. INTRODUCCIÓN:

Un cuadrotor es una aeronave de cuatro rotores distribuidos a igual distancia sobre un marco en forma de cruz, generalmente sin tripulación. Debido a que este tipo de artefactos tienen un bajo número de partes móviles, su construcción y mantenimiento son relativamente simples. Tienen la capacidad de moverse dentro de sitios cerrados y con obstáculos o en lugares inaccesibles para el ser humano. Además, es posible emplear múltiples vehículos para trabajar en conjunto y realizar tareas complejas. Empleado como sistema no tripulado puede ser usado en tareas de búsqueda y rescate, inspección industrial, monitoreo y evaluación de desastres naturales, entre otros [1] , [2].

Es creciente el interés de la comunidad científica en este tipo de aparatos, situación que se refleja en el incremento de artículos relacionados con el tema y en la presentación de trabajos de grado y tesis, tanto nacionales [3], [4], [5] como internacionales [5], [6], [7]. Por esta razón es importante tener en la universidad herramientas que faciliten el desarrollo de proyectos en esta área y así eliminar la necesidad de construir una plataforma nueva cada vez.

Las plataformas de desarrollo cumplen un importante rol tanto en la academia como en la industria ya que facilitan la implementación práctica de estudios teóricos y son una base solida y confiable para el desarrollo de un producto final. Gracias a esta herramienta, el desarrollador o estudiante se puede concentrar en un buen desarrollo teórico y practico del diseño en vez de tener que implementar funcionalidades de bajo nivel, ya que esto ha sido resuelto por quienes han elaborado la plataforma de desarrollo.

En este proyecto se creó una plataforma de desarrollo escalable con base en el cuadrotor de la marca "Alien Jump Jet" [28], que se puede reprogramar, interconectar, modificar y en la cual es posible implementar, parametrizar y validar diversos modelos, esquemas y algoritmos relacionados con cuadricópteros. Esta plataforma hará posible la aplicación de futuros proyectos, trabajos de grado y tesis de maestría en la universidad en los campos de control, robótica e inteligencia artificial.

En este libro el lector encontrará algunos de los desarrollos mas importantes tanto en hardware como en software, partiendo de la caracterización del equipo sobre el cual se trabajó y con base en los siguientes objetivos:

El objetivo general es: "Implementar una plataforma de desarrollo para cuadrotores con base en el sistema comercial "Alien Jump Jet".

Los objetivos específicos son: "Identificar las características requeridas por una plataforma de desarrollo para poder ser empleada en el estudio de técnicas de control local y cooperativo en cuadrotores".

"Verificar, las características mecánicas y de estabilidad del cuadrotor comercial "Alien Jump Jet".

"Seleccionar e instalar un sistema electrónico programable en el cuadrotor "Alien Jump Jet" capaz de manipular las variables de entrada y salida del vehículo, con capacidad de transmisión y recepción inalámbrica de datos."

"Elaborar un manual en donde se especifiquen las características de la plataforma y se den guías para su configuración y operación."

Este libro esta dividido en cinco secciones: Marco Teórico, Especificaciones, Desarrollos, Análisis de resultados y Conclusiones.

En el marco teórico se encontrará con los conceptos necesarios para contextualizar al lector en el problema que se va a abordar en el desarrollo del proyecto. Estos temas son principalmente la descripción

general de un cuadricóptero, qué es un sistema de navegación inercial, una breve descripción del *"Alien Jump Jet",* qué es una plataforma de desarrollo, una breve descripción de lo que es un micro-controlador y finalmente los canales de comunicaciones que se usan en el sistema (I2C y UART). En la sección de especificaciones se describirán y justificarán las razones por las cuales se escogieron los componentes que hacen parte del sistema electrónico. También se hará una descripción general del sistema a través de un diagrama en bloques.

La sección de desarrollo describe con mayor profundidad las partes que componen el sistema, se describirá con detalle el diseño del circuito, los cálculos hechos y las mediciones iniciales. En el análisis de resultados se mostrarán y discutirán las especificaciones y desempeño del nuevo sistema electrónico y se describirá el costo del sistema y su posible valor comercial. Para finalizar se hará una conclusión general de los resultados obtenidos y el posible aporte que puede presentar a la investigación relacionada con vehículos aéreos no tripulados, cuadricópteros y trabajo cooperativo. También se presentaran algunas sugerencias de posibles usos y continuaciones del trabajo.

2. MARCO TEÓRICO:

2.1. VEHÍCULO NO TRIPULADO: CUADROTOR

Un Vehículo Aéreo no Tripulado (UAV: Unmanned Aerial Vehicle, por sus siglas en ingles) [53] es un vehículo controlado desde tierra o por un programa implementado a bordo que le permita volar autónomamente empleando trayectos determinados con ayuda de un sistema de posicionamiento global o GPS. Las aplicaciones de este tipo de vehículos son cada día mayores en tareas muy variadas para disminuir riesgos como: irradiaciones peligrosas, exposición a químicos o fuego, búsqueda y rescate, entre otros, ya que estos pueden ser usados para la detección de incendios, identificación de manchas de petróleo, derrames químicos o radioactivos, la ubicación de personas en siniestros, entre otras aplicaciones [1],[2].

Este tipo de vehículos se empezaron "a desarrollar después de la Primera Guerra Mundial siendo ampliamente utilizados en operaciones militares, tanto para entrenamiento como para misiones" [53]. Los primeros equipos construidos no eran de vuelo autónomo sino prototipos de cuatro hélices que transportaban carga y equipos en cortas distancias. A pesar de tener su inicio en el campo militar tienen un gran potencial para aplicaciones civiles por lo que se han utilizado en campos como la telemetría y la vigilancia.

Imagen 1. Cuadrotor [54]

Un cuadrotor es una "estructura simétrica que consiste en dos barras dispuestas en forma de cruz (+) o equis (x) con motores en sus extremos. En lugar de usar superficies como en un avión, el control se efectúa variando la potencia de los cuatro motores instalados al final de la estructura, permitiéndole así realizar movimientos en muchas mas direcciones. Esto provoca variaciones en los momentos del cuadrotor que dan lugar al movimiento que se desea obtener"[53]. Usualmente el sentido de giro de las hélices en dos motores es horario mientras que en los otros dos es anti-horario, tal y como se observa en la Imagen 2. De esta forma se conserva el par motor y evita que gire sobre si mismo.

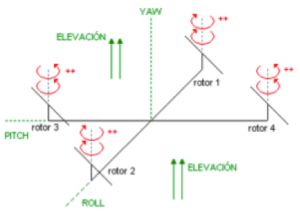


Imagen 2. Grados de libertad de un cuadrotor[19].

A diferencia de otros cuadricópteros, este sistema emplea un método diferente para anular el *"momentum"* generado por las hélices. En el "Alien Jump Jet"[28], en particular todas sus hélices giran en el mismo sentido. La anulación del efecto físico mencionado, se obtiene debido a que los cuatro rotores tienen un cierto grado de inclinación en el soporte donde se encuentran las hélices y de esta forma se cancela entre si el *"momentum"* de cada hélice.

Este sistema tiene la capacidad de moverse con tres grados de libertad angulares conocidos como YAW, PITCH y ROLL. El ángulo de YAW [53] [19] hace referencia a la rotación del sistema con respecto el eje vertical, perpendicular al equipo. Este movimiento se logra aumentando (o disminuyendo) por igual la potencia distribuida de giro de los rotores 1 y 2 y disminuyendo (o aumentando) en igual magnitud los motores 3 y 4 (ver imagen 2). Al disminuir esta potencia aumenta el par motor creando un giro contrario a las hélices que están rotando con mayor potencia.

El ángulo de PITCH [53][19] corresponde a la medida de rotación del sistema sobre su eje horizontal, perpendicular a la línea imaginaria determinada por el avance frontal del equipo por lo que los cambios en PITCH generan desplazamientos hacia adelante o hacia atrás. Con un cuadrotor en configuración de cruz es posible variar el ángulo de PITCH manteniendo los rotores 3 y 4 constantes cambiando la potencia en igual proporción pero en sentido opuesto los rotores 1 y 2 (ver imagen 2).

El ángulo de ROLL [53][19] corresponde a la medida de rotación del sistema con respecto al eje horizontal, perpendicular al eje de PITCH,.Cambios en ROLL permiten desplazamientos hacia la derecha o hacia la izquierda, empleando el mismo principio que el de inclinación (PITCH) pero esta vez dejando los rotores 1 y 2 constantes y variando 3 y 4 (ver imagen 2).

Por último, el movimiento vertical que permite ascender o descender el sistema a través del manejo distribuido de la potencia de los 4 motores, se le conoce como ELEVACIÓN.

2.2. SISTEMA DE NAVEGACIÓN INERCIAL

Los vehículos no tripulados dependen de un conjunto de sensores a bordo que le permiten al sistema de control o al operador en tierra, conocer la actitud de la aeronave e incluso otras variables de interés como la elevación, la posición y la velocidad. Al conjunto de sensores normalmente integrados por una unidad de procesamiento se le conoce como un sistema de navegación inercial [9][10].

Un sistema de navegación inercial por lo general está constituido por tres componentes: acelerómetros, giróscopos y magnetómetros. El sistema, como unidad, recolecta la información relacionada con la velocidad angular y la aceleración lineal de un objeto en movimiento que, instantes después, es enviada a un microprocesador para ser manipulada de acuerdo a las necesidades del usuario.

Acelerómetro [$\frac{\boldsymbol{m^2}}{\boldsymbol{s}}$] [9] ,[10] y [22]: Genera tres señales análogas que describen la aceleración para cada uno de los tres ejes de movimiento (X, Y y Z). La medida mas significativa de estas tres aceleraciones es causada por la gravedad. En un acelerómetro sensores capacitivos detectan la variación de desplazamiento de un número determinado de partículas de silicio o de una masa de prueba desde un punto fijo a lo largo del eje de medición debido a una aceleración en esa dirección. Cuando una aceleración actúa, la masa de partículas se mueve desde su origen y crea una variación que detecta el sensor capacitivo dando información de las medidas adquiridas.

Giróscopo [$\frac{\boldsymbol{Rad}}{\boldsymbol{s}}$] [9], [10] y [22].: Genera tres señales análogas que describen la taza de variación angular por cada uno de los tres ejes de movimiento (X, Y y Z). Este transforma la fuerza generada por un movimiento angular en una señal eléctrica proporcional producida por la resonancia de unos pequeños arcos. Los arcos generan voltajes que son detectados por una capacitancia, diferencia de potencial que luego se amplifica, filtra y desmodula.

Magnetómetro[g] [9], [11] y [22] : Este pequeño dispositivo es capaz de medir la intensidad de campo magnético que pasa a través del chip, por medio sensores de efecto Hall. Se construyen magnetómetros con sensores en los tres ejes, gracias a lo cual se puede obtener un vector de campo que da la información de la taza de variación de movimiento. El magnetómetro se utiliza para medir la dirección del campo

magnético de la tierra e indirectamente, la posición relativa del sensor en el campo por ser el campo magnético de la tierra de dirección constante.

2.3. MICRO-CONTROLADOR

Un micro-controlador es un circuito integrado que ejecuta una serie de instrucciones programadas en su memoria. Este sistema por lo general contiene una unidad central de procesamiento (CPU), memorias de almacenamiento y módulos de entradas y salidas.

1. CPU: Es el bloque encargado de ejecutar las instrucciones contenidas en memoria.

2. Memoria:. Almacena el código que elabora el usuario y a su vez los datos adquiridos durante la acción que ejecuta dicho programa. Generalmente cuentan con una memoria volátil (RAM) para datos y no-volátiles (EEPROM y Flash) para constantes y programa.

3. Unidades de entrada/salida: Son el conjunto de elementos que emplea el miro-controlador para interactuar con el entorno.

2.4. ALIEN JUMP JET

Imagen 3. Alien Jump Jet [28]

El "Alien Jump Jet" [28], elaborado por la compañía Snelflight, es un prototipo de cuadrotor diseñado especialmente para vuelos en espacios cerrados. Este, gracias a sus especificaciones y capacidades aerodinámicas, puede desplazarse en cualquier sentido y realizar giros. El equipo está constituido por una tarjeta madre sobre la cual se pueden identificar un par de giróscopos electrónicos y un sistema infrarojo de mando a distancia (de tres ejes) que le indica al bloque de potencia (Puente H) que suministre mayor o menor corriente a los cuatro motores DC Brush de 7.4 voltios con referencia 15MM LFF-N20WA-08260, alimentados por tres baterías Li-Po con una capacidad de 180 mAh..

Un vistazo a la parte mecánica y específicamente al peso del equipo puede ser resumido en los datos expuestos en la tabla 1, obtenidos de un proyecto elaborado por estudiantes de la Universidad de Lausanne en Suiza [12]. Es importante tener en cuenta que el quipo que usan es el modelo inglés y no el Chino que posee una diferencia en cuanto al método de comunicación; para el primero es por infrarrojo y para el segundo por medio de RF :

	AJJ	42g
+	Foam	10g
+	Battery	12g
	Total	64g

Tabla 1. Peso del Alien Jump Jet[12]

Algunas partes del sistema embebido del equipo original pueden ser identificadas en la Imagen 4..

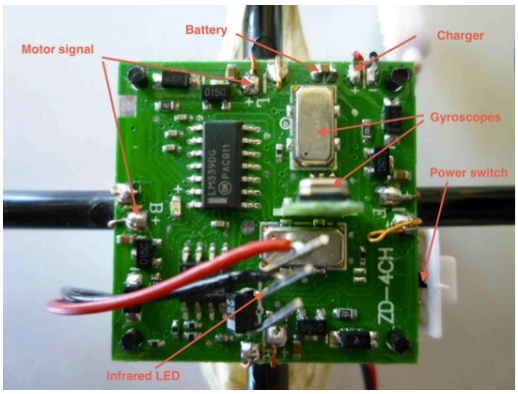


Imagen 4. Partes internas del equipo [12]

2.5. PLATAFORMA DE DESARROLLO

Las plataformas de desarrollo son de suma importancia ya que facilitan la comparación de los desarrollos que se den en la practica con los resultados teóricos y son el camino para el desarrollo de un producto final. Gracias a esta herramienta los estudiantes y desarrolladores no tendrán que desgastar recursos ni tiempo en construir un vehículo nuevo o implementar funcionalidades de bajo nivel cada vez que se quiera adelantar estudios con este tipo de vehículos. Por el contrario, podrán concentrarse en realizar un buen trabajo teórico y a su vez practico con el equipo.

Una plataforma de desarrollo está definida como *"*el conjunto de hardware y software donde se almacena, accede, recupera y estructura la información y contenidos de un sistema" [13]. Sistemas de aprendizaje e investigación como éstos ya han sido desarrollados, incluso comercialmente, permitiendo elaborar proyectos de investigación con mayor facilidad. Estos equipos, que operan como una plataforma de desarrollo, han sido elaborados por empresas como Quanser Engieneering con el "Quadrotor for Indoor Unmanned Aerial Vehicle Research" [14] o la tarjeta reprogramable de la empresa Freescale IMX53QSB: i.MX53[15]., que cuenta con la opción de implementar diversas aplicaciones.

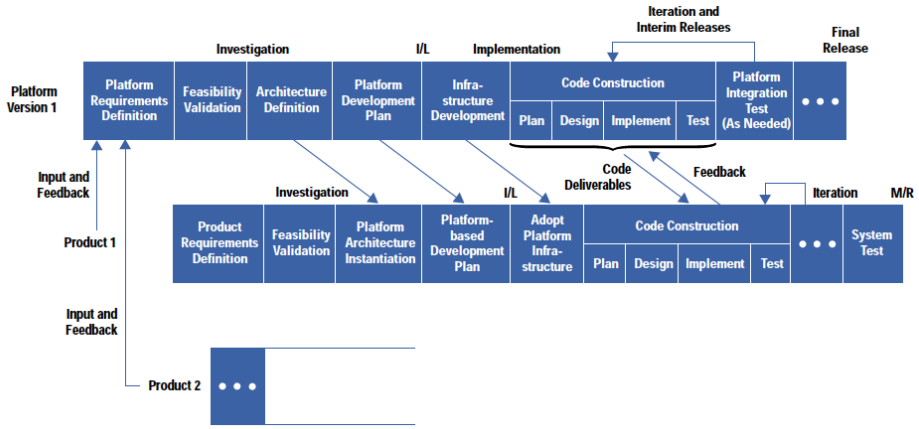


Imagen 5. Ciclo de trabajo P. Desarrollo [16]

Uno de los éxitos para la implementación de una plataforma de desarrollo, radica en la ejecución de un proceso formal compuesto por un conjunto de etapas que permitan responder principalmente las siguientes preguntas: ¿Qué hace esta plataforma?, ¿Cómo se va a usar esta plataforma?, ¿Cómo se va a presentar al usuario?, ¿Cómo se va a evaluar? y finalmente ¿Cómo se puede extender y evolucionar con el tiempo?[16].

Las etapas mas relevantes para el desarrollo de un sistema son:

Etapa de planeación y requerimientos. Relación estratégica entre la plataforma y el usuario. Se deben definir los parámetros y requerimientos de desarrollo de la plataforma.

Etapa de definición de los sub-sistemas que componen la plataforma. Definición, validación y estrategias de comprobación de la viabilidad y funcionalidad para resolver los requerimientos propuestos en la etapa anterior.

Plan de Desarrollo de la plataforma. Se definen los pasos que se deben seguir para el desarrollo de la plataforma y las fechas limites de cada etapa. Se debe documentar y evaluar cada sub-etapa.

Etapa de Desarrollo: Se ejecuta el plan anterior donde se busca entregar un modelo físico con todos sus componentes y sub-sistemas ya como parte de un producto.

Finalmente se deben definir una serie de pruebas de calidad y desempeño para poder ser ejecutadas como parte integral del proceso. Estas pruebas también operan como un modelo de validación de todos los subsistemas para su correcta ejecución y entrega final.

Una plataforma de desarrollo puede ser empleada para facilitar el desarrollo e implementación de una teoría a nivel experimental, cuando se quieren validar modelos o cuando se quiera comprobar el funcionamiento de distintas técnicas de control.

Proyectos similares:

Es importante resaltar que en la Universidad de Laussane se ha desarrollado un proyecto similar [12] pero con una diferencia fundamental: el cuadricóptero fue modificado con el objetivo de reemplazar el sistema de transmisión de datos por una unidad Bluethooth que recibe la información de un sistema de navegación inercial (adicionado), para monitorear variables importantes durante el vuelo del equipo.

Se puede evidenciar en el documento que el control y el impreso con el que viene originalmente el Alien Jump Jet no fueron extraídos sino que el nuevo sistema embebido fue puesto encima con un objetivo distinto. Esta claro que al no reemplazar el impreso original el sistema embebido con el que viene el

equipo no posee la capacidad para poder reprogramar el micro-controlador encargado del control de estabilidad, por lo tanto no existe la posibilidad de implementar un algoritmo diferente.

La plataforma de desarrollo presentada tiene un sistema embebido totalmente nuevo con el cual un micro-controlador reprogramable, a partir de los datos adquiridos por un sistema de navegación inercial, controlará la potencia entregada a cada uno de los cuatro rotores del vehículo, además de un canal de comunicaciones por medio de BlueTooth.

2.6. PROTOCOLO I2C

El protocolo I2C [38] fue desarrollado por Philips Semiconductors con el fin de interconectar dispositivos dentro de una misma tarjeta. Consta de un bus de dos líenas una para los datos y otra para el reloj del sistema, que en ingles se conocen como el "Serial Data Line (SDA)" y el "Serial Clock Line (SCLx)", los cuales llevan información entre cada uno de los dispositivos conectados al bus, bajo un esquema maestro/esclavo.

Este tipo de protocolo permite el envió serial bidireccional de datos en tramas de ocho (8) bits, siendo el primer bit enviado el mas significativo (MSB), con velocidades de 100 Kbit/s en el modo estándar, 400 Kbit/s en el modo rápido, 1 Mbit/seg en el modo rápido plus o hasta 3.4 Mbit/s en el modo de alta velocidad.

Para entender como se da el intercambio de datos entre dos dispositivos, supongamos que el micro-controlador A desea transmitir o recibir datos hacia o desde un dispositivo B. Primero, el micro-controlador A realiza un llamado, enviando una dirección especifica a los dispositivos conectados. Si la dirección enviada corresponde a la de el dispositivo B, este responde. Segundo, el micro-controlador A transmite o recibe datos hacia o desde el dispositivo B. Tercero, el micro-controlador A indica que terminó la transferencia de datos..

Ambas líneas que componen al bus del protocolo I2C son bidireccionales, lo que quiere decir que los datos pueden viajar en ambos sentidos, aunque no al mismo tiempo, ya que se pueden generan colisiones que interrumpen la transmisión/recepción entre dispositivos. Ambas líneas están conectadas a fuente a través de resistencias de *"pull-up"*. Por lo tanto, cuando el bus esta libre, ambas líneas están en "alto".

Debido a la diversidad de dispositivos que pueden ser conectadas al bus I2C, los niveles lógicos de bajo y alto ('0' y '1') no son estándar y dependen del nivel de la fuente de alimentación VDD. Sin embargo, se considera nivel '0' cuando el voltaje está por debajo del treinta porciento (30%) de VDD y nivel alto cuando está por encima del setenta porciento (70%) . Por lo tanto $\boldsymbol{V_{IL}}$ es 0.3VDD y $\boldsymbol{V_{IH}}$ es 0.7VDD.

Los datos que se transmitan deben permanecer estables durante un periodo en alto del reloj. El estado en alto o bajo del SDA puede cambiar únicamente cuando el SCL esta en bajo. Un pulso de reloj se genera por cada bit transferido.

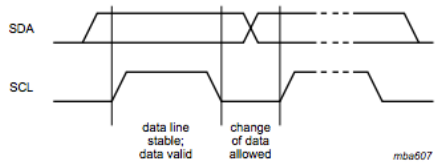


Imagen 6 a. Bit transferido en I2C [38]

El intercambio de datos se da desde una condición de inicio y finaliza con una condición de parada, ambas generadas por el maestro. La condición de inicio se da cuando en el SDA se da un cambio de alto a bajo

mientras el SCLx permanece en alto, mientras que por el contrario la condición de parada se da cuando existe un cambio de bajo a alto en el SDA, mientras el SCLx permanece en alto (ver imagen 6b). El bus se considera ocupado una vez se da la condición de inicio y libre un ciclo de reloj después de que se da la condición de parada. Si se genera nuevamente una condición de inicio en vez de parada, se considera que el bus continúa ocupado, mientras no se den los requerimientos para finalizar la transmisión.

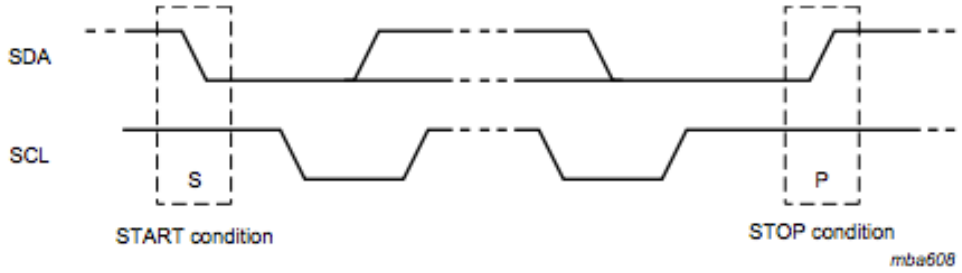


Imagen 6b. Condición de inicio y parada [38]

Por otra parte cada trama de 8 bits debe finalizar con un *"acknowledge"* o *"not-acknowledge"*. Esta condición anterior le indica al esclavo que cada dato fue o no transmitido/recibido correctamente. Si no es exitoso, el esclavo fuerza el SCLx a estar en bajo para así obligar al maestro a un estado de reposo (manteniendo el SDA en bajo por parte del transmisor). La transmisión de datos se reanuda cuando el esclavo esta listo para nuevamente recibir o transmitir datos, liberando el SCLx. El *"acknowledge"* se da cuando el SDA esta en bajo durante el noveno pulso de reloj, y por el contrario el *"not-acknowledge",* se da cuando durante el noveno pulso de reloj el SDA permanece en alto.

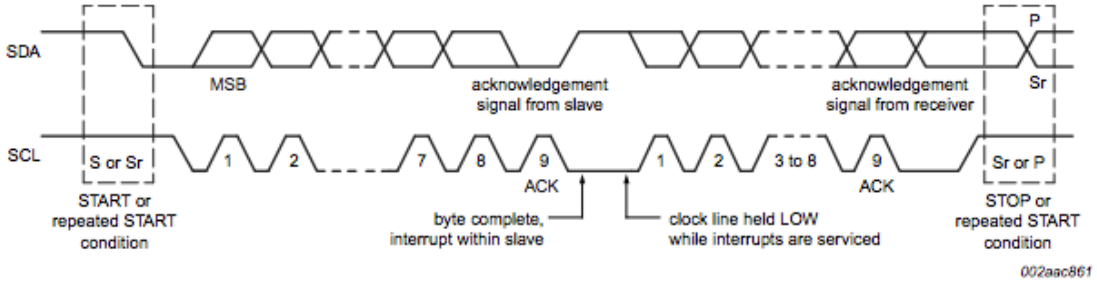


Imagen 7. Formato de transmisión [38]

El formato de la trama con el cual se efectúa la lectura de datos provenientes del esclavo, se puede observar en las imágenes 8 y 9. El siguiente es el procedimiento completo de comunicación según el protocolo I2C:

1. Se da la condición de inicio.
2. Dirección del esclavo, 7 bits (llama al esclavo)
3. Bit de escritura '0', octavo ciclo de reloj.
4. *"Acknowledge"* o *"Not-acknowledge", desde el esclavo hacia el maestro.*
5. Dirección del registro al cual se quiere acceder del esclavo, 7 bits.
6. *"Acknowledge"* o *"Not-acknowledge", desde el esclavo hacia el maestro.*
7. Nueva condición de inicio.
8. Dirección del esclavo, 7 bits (llama al esclavo).
9. Bit de lectura '1', octavo ciclo de reloj.
10. *"Acknowledge"* o *"Not-acknowledge", desde el esclavo hacia el maestro.*
11. Datos provenientes del esclavo, 8 bits.
12. *"Acknowledge"* o *"Not-acknowledge", desde el maestro hacia el esclavo.*
13. Datos provenientes del esclavo, 8 bits.

14. *"Not-acknowledge", desde el maestro hacia el esclavo.*
15. Condición de parada.

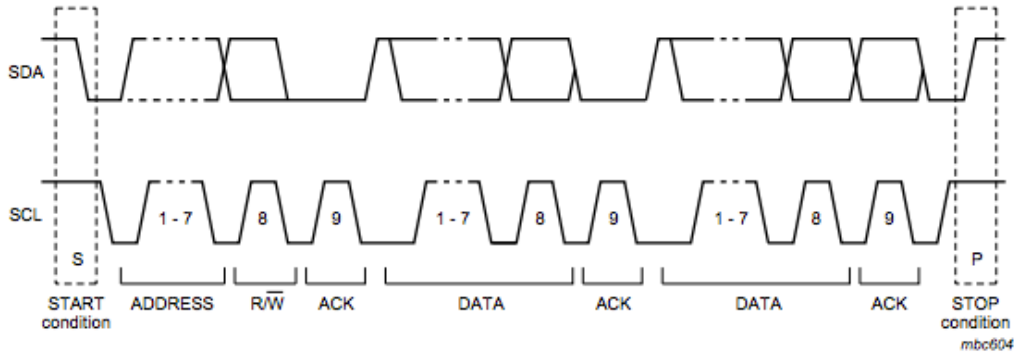


Imagen 8. Trama de datos [38]

Master	S	AD+W		RA		S	AD+R			ACK		NACK	P
Slave			ACK		ACK			ACK	DATA		DATA		

Imagen 9. Trama combinada [22]

2.7. MÓDULO UART (Universal Asynchronous Receiver Transmitter)

El módulo UART[39] consta de dos canales eTPU[1], que proveen una interfaz serial asíncrona a partir de tres líneas, TxD, RxD y GND. Ambos TxD y el RxD tienen las mismas características de tasa de transmisión, tamaño de los datos, paridad, etc. Cada uno de los canales que posee el módulo pueden ser asignados como transmisor o receptor, dependiendo la comodidad con la que se distribuyen los distintos elementos del sistema.

El módulo UART posee un transmisor, responsable de transmitir datos seriales por medio de un pin TxD, y de un receptor encargado de recibir datos seriales a través de un pin RxD. Tanto el RxD como el TxD poseen registros de conversión paralelo-serial y serial-paralelo y un registro de almacenamiento, con el fin de adquirir datos y almacenarlos temporalmente. Las interfaces de transmisión y recepción deben ser independientes debido a que cada uno de los canales eTPU controlan solamente un pin (un canal puede ser transmisor o receptor, pero no los dos al mismo tiempo). El protocolo UART permite la selección de un bit de paridad, para detectar errores durante la transmisión.

La UART no tiene un número especifico de bits por cada palabra de datos y soporta palabras entre uno (1) y veintitrés (23) bits. Normalmente se emplean palabras de 8 bits. La transmisión por UART necesita de un bit de inicio '0' (una transición de alto a bajo, lo que dura en bajo un tiempo de bit) y un bit de parada '1' (una transición de bajo a alto, lo que dura en alto un tiempo de bit).

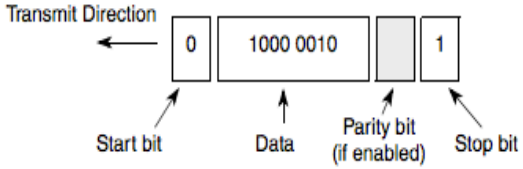


Imagen 10. Trama UART [39]

[1] Responsable de procesar las transiciones en los pines de entrada del dispositivo , y además de generar la forma de onda de salida, basado en determinadas bases de tiempo.

Los registros que posee el módulo permiten que el micro-controlador lea o escriba datos del registro de almacenamiento mientras se están recibiendo o enviando datos. Cuando se detecta que el registro de almacenamiento esta lleno, se genera una bandera para indicar que ha recibido un dato o que está listo para enviar mas.

El tiempo de bit en este protocolo hace referencia al tiempo requerido para transmitir o recibir un bit de un dato. El tiempo de bit está relacionado con el tasa de transmisión y se define como:

$$\boldsymbol{Tiempo\ de\ bit} = \frac{\mathbf{1}}{\boldsymbol{BaudRate}}$$

El receptor detecta una palabra en el buffer, comparando la pendiente de bajada siempre de un bit de inicio, esto le permite sincronizarse siempre que ocurra un evento como este, limpiando los registros y ejecutando las tareas de manera cíclica.

Siempre una palabra que es escrita en el registro de almacenamiento, debe tener el bit mas significativo (MSB) en cero. Esto le indica al protocolo UART que un nuevo dato esta disponible para ser convertido a serial y luego ser enviado a través de la línea del bus. Todos los datos recibidos son corridos hacia la derecha, enviando el bit menos significativo primero.

2.8. PWM (PULSE WIDTH MODULATION)

El PWM[36] es un método con el cual se modifica el ciclo útil de una señal cuadrada para controlar la potencia promedio de esta señal. El ciclo útil de una señal cuadrada es la relación en porcentaje del tiempo que toma en estado alto con relación al período. Específicamente en las aplicaciones en motores eléctricos, se utiliza para regular la velocidad de giro del motor, donde se mantiene el par del motor sin desaprovechar energía. Se puede utilizar tanto para motores DC como AC.

La modulación por ancho de pulso también se usa para cambiar la posición de un servomotor e incluso como método de modulación en comunicaciones

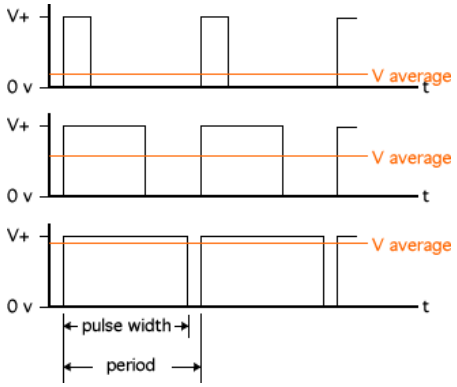


Imagen 11. Pwm para motor DC[36]

3. ESPECIFICACIONES:

Con base en el documento de la Hewelet-Packard [25] que expone un modelo para la elaboración de plataformas de desarrollo, se responde a unas preguntas básicas con respecto al objetivo y planeación del proyecto:

¿Qué hace esta plataforma?

Esta plataforma de desarrollo, es una herramienta que permite validar distintos desarrollos teóricos en el área de control, robótica y/o inteligencia artificial mediante la aplicación experimental en condiciones de laboratorio a puerta cerrada, ya que en la práctica existen diferentes comportamientos y variables que generalmente no pueden ser evaluadas en un entorno hipotético.

¿Cómo se va a usar esta plataforma?

El usuario tendrá la posibilidad de adquirir información a través de sensores de posición a bordo, para conocer la actitud de la aeronave en un instante de tiempo determinado o de recibir datos a distancia por medio de un bloque de recepción/transmisión inalámbrica. Por otra parte se podrá reprogramar un algoritmo de control a bordo que a partir de una serie de requerimientos, tomará las variables de entrada y ejecutará una serie de instrucciones, para así obtener unas salidas bien sea para variar velocidad de giro de los motores por medio de PWM y/o transmitir algún dato de acuerdo a las necesidades particulares del proyecto, bien sea por medio de I2C o UART (Bluetooth) entre el micro-controlador y algún otro dispositivo que se comunique por medio de estos protocolos.

¿Cómo se va a presentar al usuario?

La plataforma consta de dos vehículos (Cuadrotores, que contiene un sistema embebido), un programador, un manual de usuario (con los pasos mínimos que se considera son necesarios para la correcta manipulación o funcionamiento) y repuestos. Lo mencionado viene al interior de una caja que protege todos estos elementos para el momento de ser transportada. Todo lo anterior va acompañado de una base sobre la que el usuario podrá colocar el vehículo para hacer pruebas variando el movimiento en YAW y PITCH.

El cuadrotor o cuadricóptero contiene un sistema embebido para controlar la potencia entregada a los cuatro motores instalados. El criterio con el que el sistema embebido controla los motores depende de un algoritmo de estabilidad implementado en su memoria o por las instrucciones que este reciba por su canal de comunicaciones a través de Bluetooth. Las variables más importantes del sistema corresponden a los ángulos de giro del equipo en los tres ejes, conocidos por los nombres en inglés Yaw, Pitch y Roll y en conjunto se le conoce como la *"actitud"* de la aeronave.

El equipo está diseñado para operar en condiciones de laboratorio, específicamente en un espacio cerrado no muy grande (5m x 5m que es el tamaño esperado de un laboratorio de la facultad de Ingeniería Electrónica de la Universidad Javeriana, Bogotá). Por lo tanto se escogió como plataforma mecánica el *"Alien Jump Jet"*, un cuadricóptero pequeño y liviano, con autonomía media de vuelo y bajo costo en comparación con los modelos disponibles en el mercado. Es principalmente un juguete, desarrollado por la empresa inglesa Snelflight [28], del que es posible conseguir con facilidad piezas de repuesto.

El sistema embebido que se diseñó, pretende controlar la actitud del cuadricóptero a partir del trabajo conjunto de cinco bloques principales: micro-controlador, sensores, transmisión inalámbrica por medio de

Bluetooth, una memoria adicional y finalmente una etapa muy sencilla de potencia, conformada por cuatro transistores que operan como interruptores a partir del empleo de la técnica de PWM.

El sistema completo se muestra en la siguiente imagen:

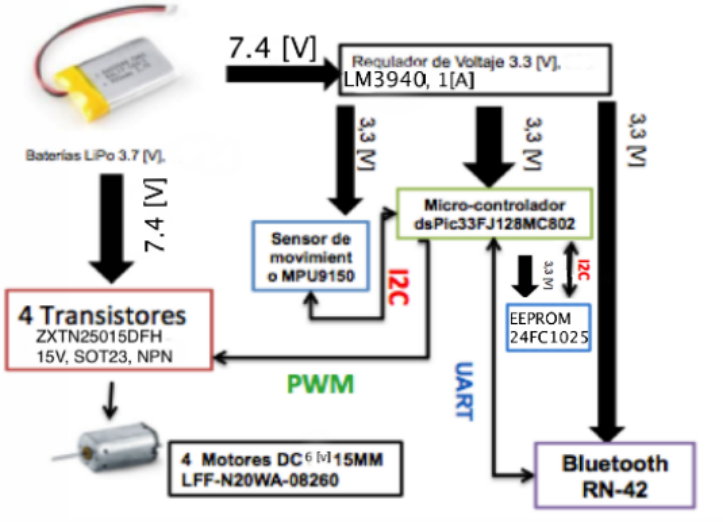


Imagen 12. Diagrama en Bloques, CERJ-1

El sistema con el cual debe contar el equipo tras el diseño del nuevo bloque de componentes debe ser lo suficientemente rápido como para recibir información proveniente de los sensores, procesarla, compararla con una orden enviada desde un control inalámbrico y modificar la potencia que se le entrega a los motores para cumplir con un requisito establecido y a su vez mantenerse en vuelo estable. Todo esto se debe hacer de una forma eficaz para que el vehículo pueda mantener un vuelo estable.

3.1. MICRO-CONTROLADOR

El micro-controlador es el eje central del sistema embebido, ya que se encarga de recibir las variables de entrada, evaluar, calcular y controlar las variables de salida del sistema. Los criterios de selección del micro-controlador fueron:

- Velocidad del procesador: Está relacionada con la velocidad con la que ejecuta una instrucción. Sin embargo, también es importante tener en cuenta los tipos de instrucciones que va a tener que ejecutar este procesador en futuras aplicaciones. Se considera que el tiempo que debe tardar en ejecutar una instrucción debe ser igual o menor a 0.1µs. Esto se soporta también mas adelante en el análisis de algunos trabajos previos que sirven como referencia para la selección de los componentes.

- La capacidad de memoria FLASH: Fue necesario hacer un análisis de trabajos anteriores que hayan usado sistemas similares para tener una referencia de la cantidad de memoria que este sistema necesita para ser funcional como plataforma de desarrollo. Una vez se tuvo este resultado, el componente seleccionado debió quedar con al menos el 50% de memoria libre para permitir la adición de algoritmos mas complejos en un futuro.

- Funciones Disponibles: Este criterio permite evaluar las funciones que se van a necesitar para poder cumplir con todas las especificaciones del diseño. Para las comunicaciones inalámbricas es necesario tener por lo menos un modulo UART y para las comunicaciones entre dispositivos es muy importante tener un puerto I2C o SPI. También es necesario que cuente con módulos de

PWM para el manejo de la potencia de los motores y temporizadores para las funciones de cálculo.

- ❖ Tamaño: Debido a que el cuadricóptero es un sistema pequeño y de empuje limitado, se deben escoger componentes lo mas pequeños y livianos posible para que estos quepan en el espacio dispuesto para ello y no contribuyan mucho al peso total del cuadricóptero. El tamaño y peso también limita la capacidad de la baterías que pueden ser empleadas y por lo tanto la autonomía de vuelo. El sistema completo debe caber en el cuadrante libre o área efectiva del Alien Jump Jet.

- ❖ Consumo de corriente: Este sistema embebido debe diseñarse para el menor consumo de corriente posible ya que con baterías limitadas es necesario que la mayor cantidad de potencia sea utilizada en los motores.

- ❖ Precio: Debe tener un precio razonable, con una buena relación de costo-beneficio. También debe ser de una marca confiable, que tenga buena documentación y fácil remplazo.

Los criterios de selección se soportan en investigaciones previas que han desarrollado o utilizado sistemas embebidos en aplicaciones similares. Esto da un criterio con respecto a cuales podrían ser los mínimos requerimientos que se necesitarían para lograr un buen desempeño en vuelo de un cuadricóptero:

- ✓ En el documento [17], implementan dos técnicas de control (PID y LQ) en un micro-controlador (REF: PIC16F876) con memoria FLASH de 8 Kbyte (arquitectura de 8 bits) y una frecuencia de operación de 20 MHz.

- ✓ En el documento [18], implementan un control LQR en un micro-controlador ATMEGA 328P el cual cuenta con una memoria FLASH de 32Kbytes (arquitectura 8 bits), una frecuencia de operación de 20 MHz,.

- ✓ En el documento [19], implementan un control PID en un micro-controlador (REF: LPC2114) con memoria FLASH de 128 Kbyte (arquitectura de 16 bits), una frecuencia de operación de 30 MHz.

Al hacer un análisis de los tipos de micro-procesadores utilizados en proyectos con aplicaciones de cuadricópteros, es posible ver las capacidades necesarias para que sea posible desarrollar proyectos de estabilidad en cuadricópteros. Con base en estos documentos y teniendo en cuenta los criterios de diseño, se escogió un micro-controlador de la marca Microchip® con referencia: dsPIC33FJ128MC802 [20].

Este micro-controlador es un DSP de 128KB de memoria flash, diseñado sobre una arquitectura Harvard de 16 bits. Cuenta con dos módulos UART, dos módulos SPI, un modulo I2C, un modulo PWM de 4 canales con HI y LOW de 16 bits de resolución, 5 “timers” de 16 bits o 2 “timers" de 32 bits, 21 pines de entrada/salida de propósito general, 0.6cmX0.6cmX0.1cm de tamaño, un consumo de corriente máximo de 300mA a una velocidad máxima de 40MIPS y un costo promedio de 3.73 USD. Un DSP es capaz de ejecutar operaciones matriciales de hasta 16X16 en un ciclo de reloj.

Adicionalmente se escogió un oscilador de cristal externo a 32MHz para operar como un oscilador primario sin emplear el PLL interno con el que cuenta el micro-controlador (“Primary (XT, HSorEC) Oscillator”).

3.2. TRANSMISIÓN INALÁMBRICA

Para el sistema de transmisión y recepción de datos la característica mas importante es que este sistema debe ser inalámbrico, ya que sistemas de comunicaciones cableados pueden afectar de gran manera la

estabilidad del vehículo volador. Para la selección de transmisión de datos se tuvieron en cuenta los siguientes criterios:

- Comunicación Bidireccional: Es de gran importancia que el cuadricóptero esté en capacidad de transmitir y recibir datos, ya que esto sirve, no solo para enviarle comandos al cuadricóptero a distancia, sino que también presenta la posibilidad de transmitir información como la de los sensores para que el usuario la pueda visualizar remotamente.

- Ancho de banda digital: Se debe establecer la capacidad de transmisión y recepción de datos del sistema o la cantidad de datos que se pueden enviar/recibir en un intervalo de tiempo, dependiendo de las prestaciones de este sistema, la velocidad y la cantidad de información que se pueda transmitir y/o recibir.

- Alcance: Como se ha dicho, el cuadricóptero está diseñado para operar en lugares cerrados y de unas dimensiones máximas de cinco metros de largo por cinco metros de ancho. Es necesario que el alcance del sistema de transmisión inalámbrico cubra al menos esta área.

- Consumo de corriente, peso, tamaño y precio: Estos criterios son los mismos que para la selección del micro-controlador, ya que se necesita que el sistema tenga el menor peso, tamaño, consumo de corriente y pecio posibles.

Los criterios de selección también se derivan de investigaciones previas:

- ✓ En el documento [18], se utilizó un modulo Xbee con un consumo de corriente de 50 mA recibiendo y 45 mA transmitiendo. La taza de transmisión de datos es de 250 Kbps; Funciona a una frecuencia de 2.4 GHz; Las dimensiones: 28mm X 24.5 mm; Su precio es de USD $23.00.

- ✓ En el documento [19], muestra un modulo Wifly GSX cuyo consumo es de 40 mA transmitiendo y 210 mA recibiendo; La taza de transmisión de datos es de 1 Mbps; El precio de este tipo de módulos es de USD$ 89.95 Las dimensiones: 37 milímetros (mm) X 20 milímetros (mm).

Con base en los criterios establecidos y en las investigaciones previas de aplicaciones similares, se escogió como sistema de transmisión remoto un modulo Bluetooth de referencia RN-42 [21], el cual es un sistema de comunicación bidireccional con una taza de transmisión de 921 Kbps a 2.4GHz y consumo promedio de 25 **mA**. Puede hacer parte de un sistema de comunicaciones complejo en el cual cada sistema transmisor/receptor tiene un identificador especifico para cada modulo Bluetooth. Sus dimensiones son 2.6 x 1.3 x 0.2 centímetros y su costo es de 16 $ USD.

3.3. SENSORES

Para el giróscopo, magnetómetro y acelerómetro se establecieron los siguientes criterios de selección:

- Resolución y Sensibilidad: Sensores con mayor resolución y sensibilidad permiten la implementación de algoritmos de control mas precisos y eficientes.

- Velocidad: Mayores tasas de muestreo permiten disminuir los efectos oscilatorios derivados de tasas bajas.

- ❖ Consumo de corriente, peso, tamaño y precio: Estos criterios son los mismos que para la selección del micro-controlador y para el modulo de comunicaciones, ya que se necesita que el sistema tenga el menor peso, tamaño, consumo de corriente y pecio.

Al analizar algunos desarrollos publicados de sistemas embebidos similares es posible extraer los requerimientos mínimos que pueden ser usados al momento de seleccionar los elementos de la plataforma de desarrollo.

Para el giróscopo, magnetómetro y acelerómetro:

- ✓ En el documento [18] se muestra un magnetómetro digital de dos ejes (REF: HMC6352); con un consumo de corriente 1 µA en *"sleep-mode"* y 1 mA en *"Steady State"*, con una resolución mínima de 0.1 Gauss y frecuencia de medición máxima de 20 Hz en *"continous mode"*.

- ✓ En el documento [18] se emplea un integrado con referencia ADXL330 con acelerómetros en tres ejes, resolución de más o menos 3g, consumo de 320 μA a un voltaje de alimentación de tres (3) voltios y tasa de muestreo de 500 Hz.

- ✓ En el documento [19] se presenta un integrado con referencia ADXL345, que consta de un acelerómetro de 3 ejes, con una resolución de mas o menos 2g, ADC máximo de 10 bits y consumo de 140 μA.

- ✓ En el documento [19] emplean un integrado con referencia ITG3200 con giróscopo de 3 ejes, ADC de 16 bits y consumo de 6.5 mA en operación y 5μA en standby. Tiene una sensibilidad de ±2000°/sec y una frecuencia de medición de 40 Hz.

Con base en los criterios establecidos y en las investigaciones previas de aplicaciones similares, se escogió el MPU-9150 [22] de la empresa INVENSENSE , que es un dispositivo único y de muy reciente aparición en el mercado (fecha de salida: 16 de Abril de 2012). Este dispositivo tiene varias características particulares muy importantes y beneficiosas para el desarrollo del proyecto.

El MPU-9150 [22] tiene acelerómetro, giróscopo y magnetómetro de tres ejes con sensibilidades de ±2g, ±250 grad/s y ±0.3μT y tasas de muestreo de 1KHz, 8KHz y 8Hz, respectivamente. Contiene un *"Digital Motion Processor"* (DMP) para procesamiento independiente de las variables espaciales y un sensor de temperatura para compensación de variaciones, todo esto dentro de un empaque de 4.8 mm x 4.8 mm x 1mm. Sus sensores tienen una resolución de 16 bits. Este dispositivo es un módulo multi-chip que se comunica a través de I2C a 400KHz y consume aproximadamente 6mA a máxima potencia

3.4. ETAPA DE POTENCIA

Para la etapa de potencia se seleccionaron cuatro transistores referencia ZXTN25015DFH [23], los cuales se emplean como interruptores on/off para manejar la potencia que se le entrega a los motores a través del uso de PWM, que es controlado por el micro-controlador. Estos transistores de acuerdo a un ensayo con el equipo original debían soportar una corriente de colector de por lo menos 1,62 A a máxima potencia o usando un ciclo útil del 100% y a su vez tener un beta ($\boldsymbol{\beta}$) relativamente alto para que en base la corriente no superara 8 mA, pues el micro controlador no puede tener una carga conectada por pin que exija un consumo mayor.

De la hoja de especificaciones se sabe que estos transistores soportan una corriente continua máxima de colector de 5 A. Teniendo en cuenta que la corriente por colector (la que consumen los motores) es de

1,62 A a máxima potencia, se debe analizar la curva de corriente de colector vs la ganancia beta (***β***), ver imagen 14, para calcular la corriente reflejada en base. Si el transistor trabaja a una temperatura promedio de 100ºC, este tendría un ***β*** alrededor de 600, con lo cual se asegura una corriente por base cercana a los 2.7mA, corriente mucho menor que la máxima que puede entregar por pin el micro-controlador seleccionado.

Parameter	Symbol	Limit	Unit
Collector-base voltage	V_{CBO}	40	V
Collector-emitter voltage (forward blocking)	V_{CEX}	30	V
Collector-emitter voltage	V_{CEO}	15	V
Emitter-collector voltage (reverse blocking)	V_{ECO}	4.5	V
Emitter-base voltage	V_{EBO}	7	V
Continuous collector current[c]	I_C	5	A
Peak pulse current	I_{CM}	15	A

Imagen 13. Valores máximos transistor [23]

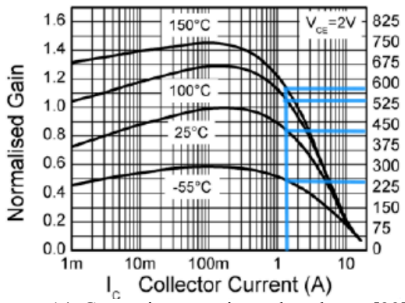


Imagen 14. Ganancia vs corriente de colector [23]

La conexión de los transistores se realiza según lo ilustrado en la Imagen 15, teniendo en cuenta que el motor es la resistencia que se observa en colector.

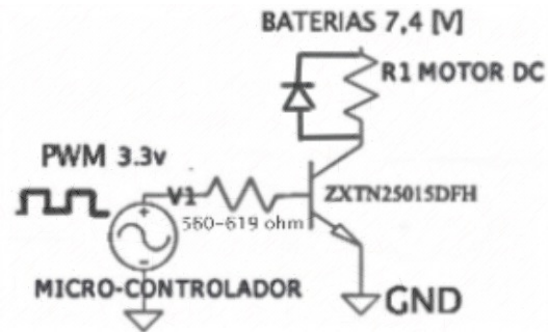


Imagen 15. Etapa de potencia

3.5. DIODO FLYWHEEL

Debido a que el motor opera como una carga inductiva cualquier cambio súbito en la corriente entregada genera altos picos de voltaje inverso. Por esta razón se selecciona un diodo, con bajo voltaje de encendido, que se coloca en paralelo al motor, de tal manera que, cuando el PWM esté en bajo, permita una ruta alterna a la corriente inversa generada por la inductancia del motor. Cuando el PWM esta en alto, el transistor se encuentra en región de saturación y el diodo en polarización inversa, la corriente pasa por el

motor. El diodo seleccionado, de referencia LSM115J [24], es un diodo rectificador Schottky, con *"Forward voltaje"* de 0.22 V @ 1 A, corriente en directa de 1A, corriente máxima de 50 A en pulsos de hasta 8.3 ms y voltaje de ruptura inversa de 15 V.

3.6. BATERÍAS

La selección de baterías estuvo sujeta a criterios con los cuales se buscaba mejorar el desempeño del equipo con respecto al vehículo original y conservar un buen tiempo de uso, alimentando el circuito y los motores simultáneamente. Por lo tanto y luego de verificar las necesidades del sistema, las baterías debían estar en capacidad de suministrar 1,62 A a cada uno de los motores y a su vez 36 mA al circuito durante aproximadamente 4 minutos (tiempo promedio de uso que dura equipo original a máxima potencia), que es considerado tiempo suficiente para ser empleado como plataforma de desarrollo en condiciones de laboratorio. Hay que tener en cuenta que dichas baterías no deben tener un peso elevado ya que limitaría el desempeño del sistema, reduciendo considerablemente la capacidad de carga útil por lo tanto la posibilidad de adicionar mas dispositivos que aplicaciones posteriores a este proyecto puedan requerir.

De acuerdo a lo anterior se seleccionaron dos baterías en serie de 3,7 voltios cada una (350 mAh, S2) y un peso de diez gramos, para un total de 7,4 V. Si los cuatro motores a máxima potencia consumen 6,4 A, la batería se agotaría en un tiempo total de 3 minutos y 30 segundos. También se tuvo en cuenta el factor de descarga de las baterías, pues estas deben estar en capacidad de suministrar una corriente instantánea de 6.4 A. Las baterías seleccionadas tienen un factor de 20C, lo que indica que están en capacidad de entregar un máximo de corriente continua de hasta 7 amperios, valor que viene de multiplicar la capacidad de la batería por 20 (0.350*20).

3.7. REGULADOR

El regulador debe estar en capacidad de alimentar con un voltaje fijo de 3.3 voltios el sistema embebido y tener capacidad suficiente para proveer alimentación a componentes que se adicionen posteriormente al desarrollo de este proyecto. Se espera que estos sean elementos como cámaras, sensores de altura o proximidad, sensores de intensidad de luz, etc. Por esta razón se seleccionó un regulador de referencia LM3940 [25], capaz de suministrar hasta 1 A de corriente, un voltaje de salida fijo de 3.3 Voltios, un bajo voltaje de *"drop-out"* de 1 V a 1A, protección de corriente inversa, protección de temperatura y capacidad de disipación de 174 °C/W. Este dispositivo recibe una entrada máxima de 7.5V y mínima de 4.5 V, suficiente para las baterías que se van a emplear.

3.8. MEMORIA ADICIONAL

Debido a que el sistema debe operar como una plataforma de desarrollo, es posible que en un futuro, para ciertas aplicaciones, sea necesario almacenar datos, información adicional o implementar códigos de control mas robustos. De acuerdo a esto, se seleccionó una memoria adicional no volátil de 128KB, EEPROM, referencia 24FC1025[26], que esta en capacidad de comunicarse a través de I2C @ 400 KHz, alimentada a 3.3 V y con un consumo de corriente máxima de 450µA.

3.9. CONECTOR DE PROGRAMACIÓN

Se selecciono un conector de programación micro-USB-B hembra, el cual es uno de los mas livianos y de fácil acceso del mercado. Sus dimensiones son relativamente pequeñas (8,8 mm X 5 mm X 2,3 mm) razón

por la cual no causan un impacto notable con relación al espacio y afectan mínimamente el tamaño del sistema.

3.10. TABLA DE COMPONENTES

Todos los componentes escogidos son polarizados a 3.3 voltios, y están resumidos en la tabla que se muestra a continuación:

COMPONENTE	REFERENCIA	DESCRIPCIÓN
Microprocesador	DSPIC33FJ128MC802	16bits, 40MIPS, 128 KB FLASH,3.3V
Unidad de Bluetooth	RN-42	clase 2, 2.4Ghz, 1Mbit/s.
Acel., Gyro. y Mag.	MPU-9150	I2C,3.3V
Transistores X 4	ZXTN25015DFH	Ic (max) 5 A, beta minimo 300
Regulador	LM3940	In: 4.5V-7.5V; Out: 3.3V, 1A
Estructura Mecánica	Alien Jump Jet	4 motores DC de 8v, 1.6A
Baterías	Lithium-ion polymer	350mAh, 7.4V, 20C, 2S
EEPROM	24FC1025	3.3V,128 Kbyte,I2C
Diodo Flywheel	LSM115J	Vfr=0,22 V @ 1A, Vrr=15 V, Ifsm=50 A x 8,3ms

Tabla 2. Componentes

4. DESARROLLOS:

Una parte fundamental en el desarrollo de este proyecto fue determinar las características mecánicas del vehículo tales como dimensiones, peso total del equipo, peso de algunas partes que lo integran y empuje. Este procedimiento se realizó con el vehículo original al cual se le midió directamente cada variable en un banco de pruebas.

Un resumen de las mediciones se muestra en la tabla 3.

SÍMBOLO	PARÁMETRO	MEDIDA	UNIDADES
A	longitud brazo	0,20	m
Ptot	Peso total	0.068039	Kg
Phel	Peso hélice	0,00746	Kg
B	Diámetro hélice	0.0683	m
Pbat	Peso batería	0.016110	Kg
D	Dimensiones Baterías	0.0323*0.0109*0.0105	m
Af	Área efectiva	0.044*0.044	m
Pc	Peso circuito original	0.006092	Kg
Atot	Altura rotor-base	0.0288	m
Emp	Empuje	**1205,3255**	N

Tabla 3. Pesos y medidas equipo original

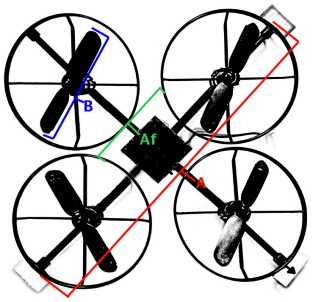


Imagen 16. Medidas sobre el equipo

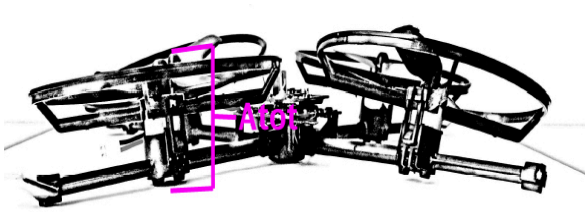


Imagen 17. Altura Aspa-base Alien Jump Jet

La medición del peso de algunas partes del equipo original, como el circuito impreso, las baterías y las hélices, se realizaron con una gramera tal y como se muestra en las imágenes 18, 19, 20 y 21.

Imagen 18. Peso equipo original

Imagen 19. Peso hélices

Imagen 20. Peso baterías

Imagen 21. Peso circuito original

4.1. MEDICIÓN DEL TIEMPO DE CARGA Y TIEMPO DE USO DEL EQUIPO ORIGINAL

El tiempo de carga es el intervalo temporal, dado en minutos, desde el inicio del proceso de carga de baterías con carga mínima hasta carga completa, indicado por medio del cambio de color de un led, de rojo a verde, en el cargador que hace parte de los accesorios con los que cuenta el equipo original. El tiempo de uso a máxima potencia es el intervalo temporal, dado en minutos, desde un inicio con baterías con carga total hasta una carga mínima, lo que se da luego de observar una perdida paulatina del empuje del vehículo y una caída lenta hasta tocar el suelo de donde ya no puede volver a elevarse.

Ambos tiempos descritos con anterioridad se muestran en las dos tablas que se encuentran en la taba 3a.

Tiempo de carga

Tiempo de carga aproximado (minutos) ($\boldsymbol{Tca}$)	20.34
Tiempo de uso a máxima potencia aproximado (minutos) Autonomía de vuelo ($\boldsymbol{Tvu}$)	4

Tabla 3a. Tiempo de carga y uso.

4.2. PRUEBA DE VUELO DEL EQUIPO ORIGINAL

Como constancia de que el equipo cuenta con las características aerodinámicas para poder elevarse, se realizaron pruebas de vuelo a puerta cerrada y al aire libre durante periodos de tiempo cortos. De igual forma se tomaron varios videos para poder demostrar que el equipo cuenta con la capacidad de mantenerse suspendido y no caer por elementos de diseño o fallas que no halla tenido en cuenta el fabricante[28].

En la imagen 22 se observa el equipo elevándose en un espacio abierto y en la imagen 23 en un espacio cerrado. Según el fabricante, la estabilidad del equipo en vuelo depende directamente del mando a distancia infra-rojo de cuatro canales y de un control proporcional.

Imagen 22. Alien Jump Jet en espacio cerrado

Imagen 23. Alien Jump Jet en espacio Abierto

Las imágenes fueron extraídas de videos tomados durante las pruebas de vuelo del equipo original y se encuentran en el CD adjunto al presente informe.

4.3. MEDICIÓN DEL EMPUJE SIN MODIFICACIÓN

El empuje se determinó a través de un procedimiento bastante sencillo. Primero se reunieron una cantidad determinada de monedas en denominaciones de 100, 200 y 500 pesos y se determinó el peso particular de acuerdo con el Banco de la República de Colombia; para la moneda de 100 pesos es de 5,31 $\pm$ 3% gramos (g) [29], para la de 200 pesos es de 7,08 $\pm$ 3% gramos (g) [30] y para la de 500 es de 7,14 $\pm$ 3% gramos (g) [31]. Segundo, se pusieron todas las monedas dentro de una bolsa plástica (cuyo peso se consideró despreciable) y se propuso elevar el equipo con el peso adicional y a máxima potencia por lo menos a mas de diez (10) centímetros desde el punto inicial. Como esto no sucedió en el primer intento, se retiraron monedas hasta lograr el objetivo puesto.

Para la prueba se despreció cualquier tipo de fricción. La resolución de este método es baja teniendo en cuenta que el mínimo cambio en la medida es de 5,31$\pm$ 3 % gramos (g) de la moneda de 100 pesos. Sin embargo, se consideró suficiente por no requerir de una precisión alta en la medida.

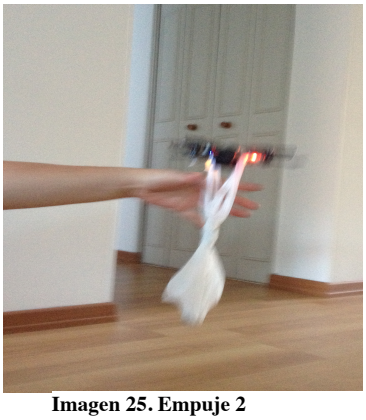


Imagen 25. Empuje 2

Con este procedimiento se encontró que el peso máximo que puede levantar el cuadricóptero es 54,87 gramos (g) o 0.05487 Kilogramos (Kg) (9 monedas de 100 pesos y 1 de 200 pesos). El empuje máximo se calcula entonces sumando este peso con el peso total del equipo y multiplicándolo por la aceleración de la gravedad que en Bogotá, que es en promedio de 9,80665 $\frac{m}{s^2}$ [32].

$$\boldsymbol{Empuje = (Peso\ adicional + Peso\ total) * aceleración\ gravedad\ Bogotá}$$

$$\boldsymbol{Empuje = (54,87g + 68.039g) * 9,80665\ \frac{m}{s^2}}$$

$$\boldsymbol{Empuje = 1205,3255\ Newtons}$$

4.4. ANÁLISIS CUALITATIVO DEL CONTROL: EQUIPO ORIGINAL

El equipo, según la pagina del fabricante, cuenta con un control proporcional con una referencia establecida por un mando a distancia y un giróscopo de tres ejes con auto-cero o corrección del "drift".

Se realizó una prueba de desempeño sobre una base de pruebas con un grado de libertad, inclinando el vehículo tal y como se muestra en la imagen 26.

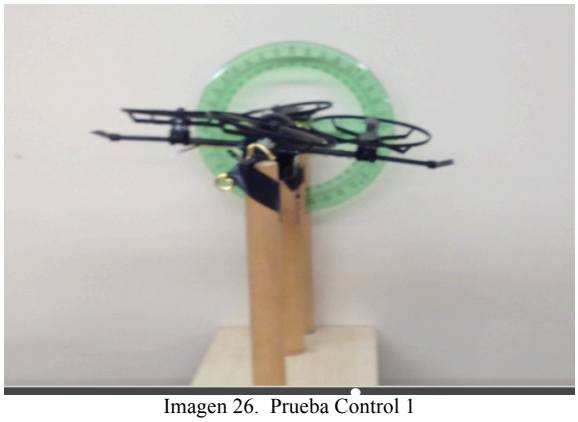


Imagen 26. Prueba Control 1

En las pruebas se observó que si el equipo se inclina mas de quince grados permanece en esa posición y no esta en capacidad de regresar al nivel cero del transportador. Cabe anotar que el análisis del control con el que cuenta el equipo original, se dio de manera cualitativa a través del análisis de imágenes y video. Para observar la respuesta del control se generó una perturbación externa o un cambio en el punto de referencia.

Esta prueba arrojo algunos resultados que se pueden observar en el video que se encuentra en el CD que se entrega con el proyecto.

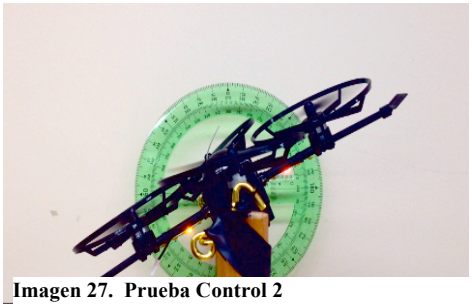


Imagen 27. Prueba Control 2

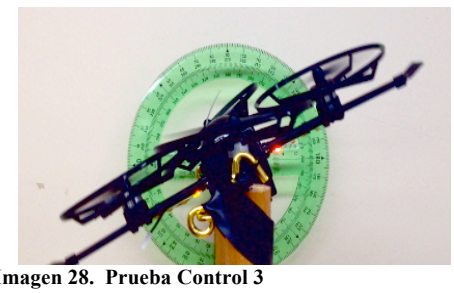


Imagen 28. Prueba Control 3

El máximo sobre-pico fue de cinco grados tras un cambio en el punto de equilibrio de diez grados en sentido anti-horario, tal y como se observa a continuación a través de las imágenes 29, 30 y 31. En la primera se alcanza a distinguir el máximo sobre-pico, en la segunda y tercera el retorno al nivel cero del transportador. El tiempo de estabilización es de 500 mili-segundos, el cual se midió a través de el análisis detallado del video (cortando el video por fragmentos y analizándolo cuadro a cuadro). Finalmente el error en estado estable es de dos grados aproximadamente, ver imagen 31.

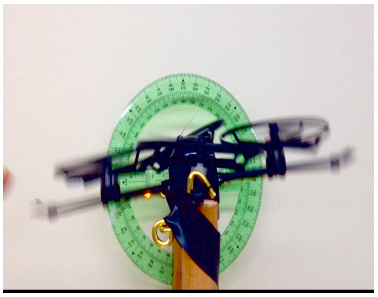


Imagen 29. Control 4

Imagen 30. Control 5

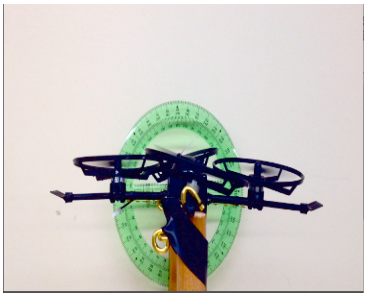


Imagen 31. Control 6

4.5. EL IMPRESO

El sistema embebido esta alojado en un impreso de dimensiones 3.922 centímetros por 3.881 centímetros. Las características del impreso son: Componente doble capa con elementos de montaje superficial en material FR4 de 0.8 milímetros de espesor (FR4: fibra de vidrio mezclada con resina epóxica y con un retardante al fuego), espacio mínimo entre caminos de 0.2 milímetros (8 mil), grosor de los caminos de una onza de cobre y ancho de los caminos mas pequeños de 0.245 milímetros.

Para que el impreso soporte la corriente que se le debe entregar a los motores, se calcula el grosor de los caminos de esta etapa. De acuerdo a la ecuación para el calcular el grosor del camino en un impreso [55] y sabiendo que el ancho del cobre es de 1 onza, se obtiene un ancho de camino de 1,09 mm

$$\boldsymbol{Area[mils^2]} = \left(\frac{\boldsymbol{Corriente[Amps]}}{\boldsymbol{k} * (\boldsymbol{Temperaturaque\ alcanza})^{\boldsymbol{b}}}\right)^{\frac{\boldsymbol{1}}{\boldsymbol{c}}}$$

Donde k, c y b son constantes que se extraen de la "Generic Standard on Printed Board Design" [33][55], donde cada una tiene un valor: k=0.048, c=0.725 y b=0.44

$$\boldsymbol{Area[mils^2]} = \left(\frac{\boldsymbol{7\ [Amps]}}{\boldsymbol{0.048} * (\boldsymbol{100}℃)^{\boldsymbol{0.44}}}\right)^{\frac{\boldsymbol{1}}{\boldsymbol{0.725}}}$$

$$\boldsymbol{Area[mils^2]} = \boldsymbol{58,997956}$$

$$\boldsymbol{Ancho[mils]} = \frac{\boldsymbol{Area[mils^2]}}{\boldsymbol{Anchoencobre[oz]} * \boldsymbol{1.378}\left[\frac{\boldsymbol{mils}}{\boldsymbol{oz}}\right]}$$

$$Ancho[mils] = \frac{58,997956}{1 * 1.378\left[\frac{mils}{oz}\right]}$$

$$Ancho[mils] = 42,81419\ mils$$

$$Ancho[mm] = 1.08747814mm$$

El camino mas ancho es el que alimenta a los cuatro transistores y el regulador, el cual se diseño con un ancho 1.42 milímetros. El montaje se realiza por medio de un "stencil", sobre el cual se aplica una pasta de soldadura libre de plomo, se adhieren los componentes por medio de una maquina "Pick and Place" y se procede a calentar el circuito a una temperatura de aproximadamente 200 grados centígrados. Los archivos "gerber" de la capa superior e inferior se pueden ver en las imágenes 32 y 33.

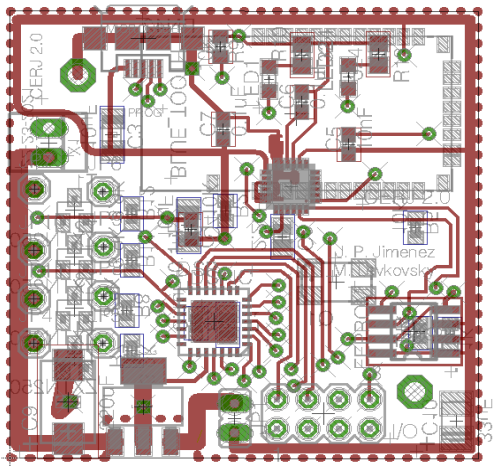


Imagen 32. Gerber capa superior

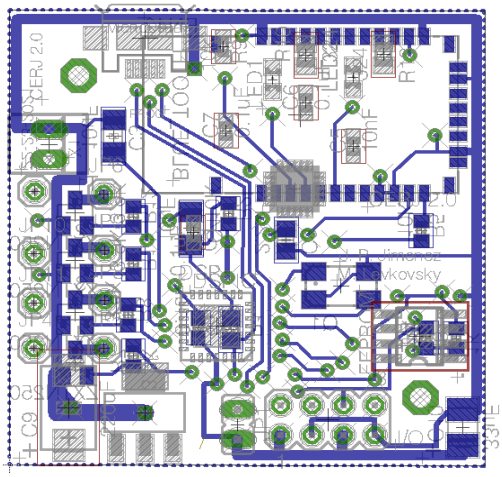


Imagen 33. Gerber capa inferior

En la imagen 34 se puede observar la ubicación de cada uno de los componentes que hacen parte del sistema embebido en ambas caras del impreso.

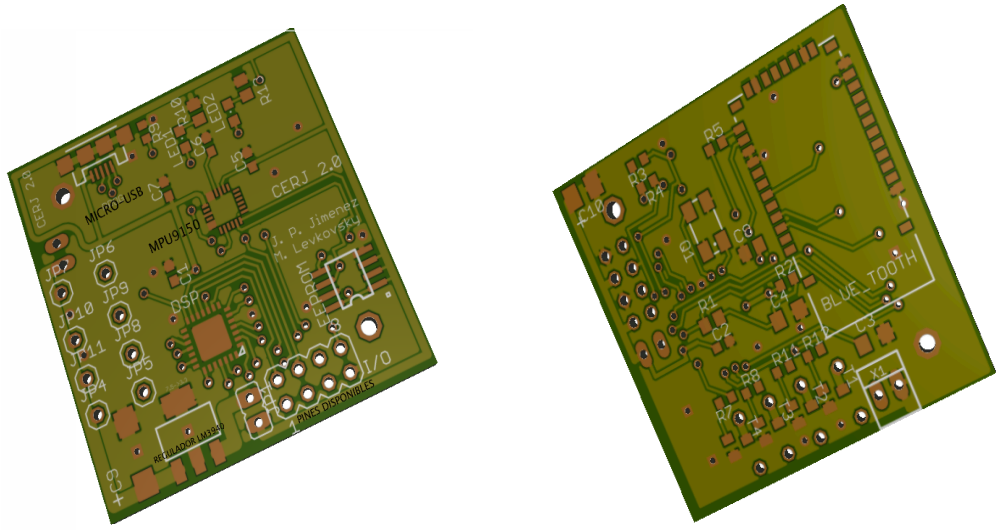


Imagen 34. Render vista Superior e Inferior Sistema Embebido

Algunos de los criterios de ubicación de los componentes en el impreso son:

- El sensor de movimiento esta en la capa superior del impreso con el fin de ubicar el eje vertical (Z) apuntando hacia arriba.

- Las baterías en la parte inferior del impreso, para no generar incomodidad y permitiendo que el centro de masa este mas bajo, mejorando la estabilidad del equipo.

- El Bluetooth en la capa inferior, ya que la necesidad de acceso manual por parte del usuario es mínima.

- El micro-controlador en la capa superior, pues éste es el elemento de mayor importancia en el sistema y debe estar ubicado lo mas cerca posible del sensor de movimiento, que también esta ubicado en esa capa. Los caminos de largas distancias operan como antenas, que pueden generar ruidos indeseados, afectando el comportamiento de la transmisión de datos.

- Es recomendado colocar condensadores de desacople en fuente para cada componente. Algunos de estos condensadores son de tantalio y otros cerámicos, con valores que oscilan entre 0.1 μF y 10 μF.

- El puerto micro-USB se ubicó en la capa superior del impreso para que el usuario pueda acceder con facilidad a conectar el programador.

- Los leds de estado en la capa superior pues estos indican cuando esta conectado el Bluetooth con el computador o algún otro dispositivo.

- El circuito se diseño con planos de tierra que no solo sirven para eliminar ruido, sino que también sirven como disipadores de calor.

- Se procuró mantener los caminos de fuentes paralelos a un camino de tierra para disminuir los efectos de ruidos eléctricos, ya que esta distribución hace las veces de un condensador.

4.6. PROGRAMACIÓN

La programación se efectúa a través de un programador de referencia Microstick II [34], que está en capacidad de programar dsPICs y algunos PICs de la empresa Microchip®. Este sistema posee un socket para poder ubicar los dispositivos con empaque “DIP”, del cual se tomaron las cinco líneas necesarias para la programación,: TARGET_DATA, TARGET_CLOCK, TARGET_MCLR, VCC y Tierra (GND).

Por razones estéticas se ubico en una caja, en donde uno de sus extremos se conecta a través de un puerto micro-USB al sistema embebido del equipo y el otro vía mini-USB al computador, como se observa en la imagen 35.

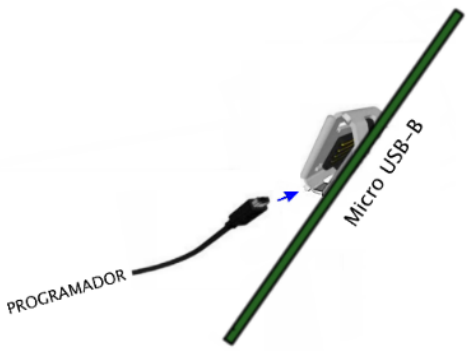


Imagen 35. Conexión física-1

El Microstick II cuenta con un puerto mini-USB, un regulador de voltaje a 3.3 V, un "debugger circuit", un dispositivo emulador (PIC por defecto) y finalmente un socket donde se ubica el dispositivo con empaque "DIP" que se desea programar (ver sección de anexos).

- El TARGET_DATA es el pin a través del cual ingresa el código convertido en un tren de pulsos seriales par ser almacenado en la memoria flash del dsPIC.

- El TARGET_CLOCK establece el "*timing"* o reloj con el cual se programa el dispositivo.

- TARGET_MCLR habilita el proceso de programación. Este pin borra la memoria del dispositivo poniéndose en bajo(lógica negada) y luego en alto durante un periodo que se realiza el proceso de programación.

- VCC y tierra (GND) polarizan el sistema durante el proceso de programación. El

El entorno de programación empleado fue MPLAB ® X IDE, el cual soporta un lenguaje C orientado al desarrollo de aplicaciones en micro-controladores. Sobre este ambiente de desarrollo integrado (IDE por sus siglas en ingles) se elaboró el código usado en la plataforma.

4.7. MEMORIA FLASH DISPONIBLE

La capacidad de memoria es limitada y varía dependiendo del tipo de micro-controlador. El micro-controlador seleccionado con referencia dsPIC33FJ128MC802 cuenta con 128 KB de memoria flash sin tener en cuenta la memoria EPPROM que se adicionó.

Con el fin de validar que el sistema esta en capacidad de almacenar el código que controla las funciones básicas del cuadricóptero y aún tener espacio libre de memoria (por lo menos el 50% libre restante) para almacenar códigos mas robustos, se compilaron dos códigos para cuadricópteros y se observó la cantidad de memoria que estos requirieron ver capitulo 5.6. También se examinaron varios documentos que utilizan sistemas electrónicos similares y emplean algoritmos de estabilidad en cuadricópteros, lo cual da una idea del tamaño que estos algoritmos ocupan ver capitulo 3.1.

4.8. VELOCIDAD DEL MICRO-CONTROLADOR

El micro-controlador cuenta con un oscilador de cristal externo de 32 MHz, pero de acuerdo a la hoja de especificaciones del dsPIC33FJ128MC802, las instrucciones trabajan a una frecuencia de 16 MHZ, si no se emplea el PLL interno del dispositivo y se colocan los bits 12, 13 y 14 en cero (DOZE, relacionados directamente con la frecuencia a la que operan las instrucciones) del CLOCK DIVISOR REGISTER (CLKDIV). Luego de configurar el registro relacionado con el reloj del sistema, la frecuencia a la que operan las instrucciones es equivalente a dividir la frecuencia del oscilador externo por dos.

Al colocar los bits DOZE del sistema en cero, significa que la CPU del micro-controlador ejecutará 40 MIPS (sin alterar la cantidad de bits por segundo que se reciban o se transmitan en las distintas interfaces de comunicación que se estén empelando), ,10 MIPS si se colocan los bits en 2 (010) y así sucesivamente. De acuerdo a como se configuren estos bits la CPU ejecutará un determinado número de instrucciones por segundo, como se puede ver en la imagen 36.

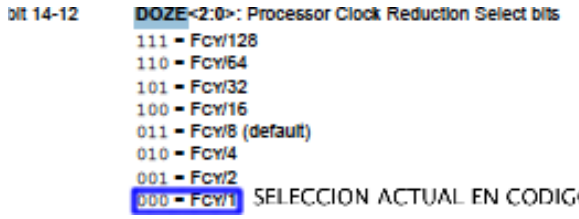


Imagen 36. DOZE bits, CLOCK DIVISOR REGISTER (CLKDIV) [20]

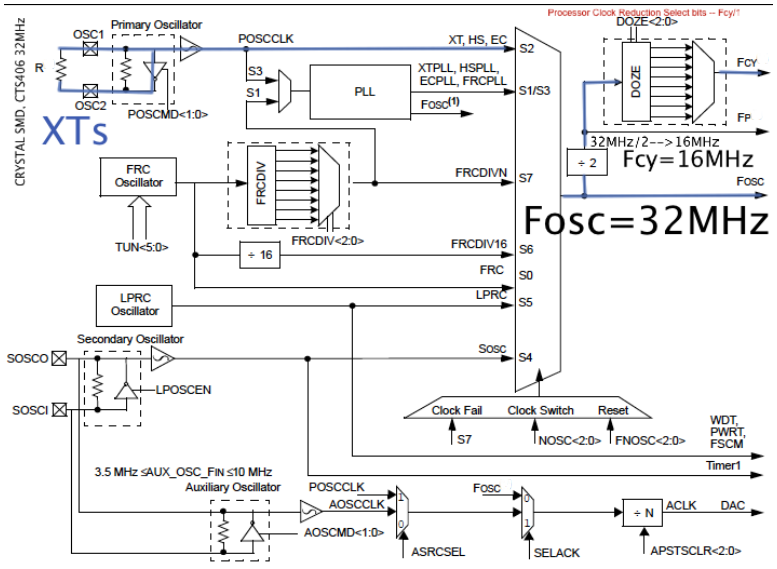


Imagen 37. CLK DIAGRAM dsPIC33 [20]

4.9. PINES EMPLEADOS

En las tablas 4, 5, 6 y 7 se muestran los pines empleados por cada dispositivo (el micro-controlador, los sensores y el Bluetooth) al igual que una breve descripción del uso del pin.

Los pines de programación del micro-controlador, son: el TARGET_DATA (pin 1), el TARGET_CLOCK (pin 2), el TARGET_MCLR (pin 26) y finalmente VCC (pin 10) y Tierra (GND) (pin5). Estos pines mencionados luego de realizarse la programación pueden ser empleados como de propósito general.

→ dsPIC33FJ128MC802

FUNCIÓN	PIN	DESCRIPCIÓN
TX/Rx UART, RB3	4	Bidireccional, I/O
TX/Rx UART, RB2	3	Bidireccional, I/O
Vss	5	GND
RA2, OSC1 (32 MHz)	6	Oscilador externo, I/O
RA3, OSC2 (32 MHz)	7	Oscilador externo, I/O
VDD	10	VDD, 3.3 [V]
ASCL1	12	I2C, Alternative Serial Clock Line
ASDA1	11	I2C, Alternative Serial Data Line
VCAP	17	Estabilidad regulador, Condensador
RB08	14	PWM
RB10	18	PMW
RB12	20	PWM
RB14	22	PWM
AVSS	24	GND
AVDD	25	VDD, 3.3[V]
$\overline{MCLR}$	26	Master clear,conectado a 3.3 [V]

Tabla 4. Conexión micro-controlador

→ MPU9150

FUNCIÓN	PIN	DESCRIPCIÓN
CLKIN	1	Conectado a GND
VLOGIC	8	Voltaje de operación 3.3[V]
AD0	9	Bit 0 de la dirección, GND
FSYNC	11	Sincronización con mas componentes, No utilizar, GND
INT	12	Interrupción, GND, No utilizar
VDD	13	VDD, 3.3[V]
GND	15,17,18	GND
SCL1	23	I2C, Serial Clock Line
SDA1	24	I2C, Serial Data Line

Tabla 5. Conexión MPU9150

→ Bluetooth, RN-42

FUNCIÓN	PIN	DESCRIPCIÓN
UART_RX	13	UART receptor, Input
UART_TX	14	UART transmisor, output
UART_RTS	15	En alto deshabilita el transmisor host, GND
UART_CTS	16	En alto deshabilita el transmisor, GND
PIO2	19	Status LED
PIO5	21	Status LED

PIO4	22	Establece los parámetros de fábrica que hay por defecto, GND
VDD	11	3.3[V]
VSS, GND	1,12,25,27,28,29	GND, 0[V]

Tabla 6. Conexión Bluetooth

→ EPPROM 24FC1025

FUNCIÓN	PIN	DESCRIPCIÓN
A0	1	Chip Address Inputs, VDD
A1	2	Chip Address Inputs, VDD
A2	3	Non-configurable Chip Select, VDD
Vss	4	GND , 0[V]
SCL1	5	I2C, Serial Clock Line
SDA1	6	I2C, Serial Data Line
WP	7	Write-Protect (WP), VSS
VDD	8	3.3[V]

Tabla 7. Conexión EEPROM

4.10. RECOMENDACIONES DE CONEXIÓN DEL SISTEMA EMBEBIDO

Las resistencias y condensadores que se identifican en el circuito se colocaron de acuerdo a una serie de sugerencias de conexión mínimas dispuestas en las hojas de especificaciones de cada componente.

Las sugerencias son las siguientes:

Para el micro-controlador:
Pin 17, $\boldsymbol{V_{CAP}}$

- Condensador conectado del pin 17 a tierra,
- Se emplea con el fin de estabilizar el voltaje de salida del regulador, reduciendo el ruido.
- Este pin no debe estar conectado a fuente (VDD).
- Según la hoja de especificaciones del componente, el valor de esta capacitancia oscila entre 4,7 µF y 10 µF.
- El condensador preferiblemente debe ser cerámico o de tantalio.
- El regulador provee energía al núcleo del componente a través de un pin VDD. Normalmente el núcleo digital de esta familia de integrados opera a 2.5 [V] y por lo tanto, para evitar incomodidades durante el diseño, este chip dispone de un regulador interno *"ON-CHIP Regulator"*.
- Condensador montaje superficial empaque C0805.

Capacitancias de desacople en fuente, Pin 10 y Pin 25

- Condensadores conectados de fuente a tierra (Pin $\boldsymbol{Vdd\ y\ AVdd}$).
- Se emplean con el fin de filtrar el ruido.
- La hoja de especificaciones recomienda un valor de 0.1 µF o 100 nF.
- Se recomienda condensadores cerámicos.
- Condensador montaje superficial empaque C0805.

Oscilador externo, pin 6 y 7

- Conectado entre el pin 6 y 7, 32 MHZ.
- Cuenta con un PLL interno, que engancha a la frecuencia de operación interna.
- Cuenta con un divisor interno, $\div \mathbf{2}$, por lo tanto $\boldsymbol{F_{CY}}$=16 MHz

- Oscilador de empaque CTS406.

Master Clear, Pin 26
- Reset del dispositivo
- La hoja de especificaciones recomienda un valor de 10 µF.
- Condensador de montaje superficial empaque C0805.
- Resistencias de 10 k**Ω** y 330 **Ω**.
- Resistencias de montaje superficial empaque R0603.

Para el Sensor, MPU9150
Capacitancia de desacople en fuente, Pin 13
- Condensador conectado de fuente a tierra.
- Se emplea con el fin de filtrar el ruido.
- La hoja de especificaciones recomienda un valor de 0.1 µF o 100 nF.
- Se recomienda condensador cerámico.
- Condensador montaje superficial empaque C0805.

Charge Pump, Pin 20
- Es el encargado de generar el voltaje alto para los osciladores micro-electromecánicos (MEMS) internos del sensor.
- Conectado a tierra por medio de un condensador CPOUT.
- La hoja de especificaciones recomienda un valor de 2.2 nF.
- Condensador montaje superficial empaque C0805

LDO, Regulador de Voltaje, Pin 10
- Mantiene el voltaje interno necesario para la correcta operación de los bloques del dispositivo.
- Cuenta con un capacitor conectado de su salida a tierra con el fin de filtrar el ruido.
- La hoja de especificaciones recomienda un valor de 0.1µF.
- Condensador montaje superficial empaque C0805.

Capacitancia de Bypass en VLOGIC, Pin 8
- Condensador conectado de fuente a tierra.
- Se emplea con el fin de filtrar el ruido.
- Se recomienda condensador cerámicos.
- La hoja de especificaciones recomienda un valor de 10 nF.
- Condensador montaje superficial empaque C0805.

Para el Bluetooth, RN42
- Status LED, Pin 21 y Pin 19, con resistencias de 330 **Ω** a tierra.
- Resistencia a tierra, Pin 20 establece los parámetros de fabrica.
- Resistencia de montaje superficial empaque R0603.
- LED de montaje superficial con empaque CHIPLED_0805.

4.11. DESARROLLOS SOBRE EL PROTOCOLO I2C

4.11.1. CALCULO DE LA RESISTENCIA DE PULL-UP

Los pines de entrada/salida de los dispositivos conectados al bus del protocolo I2C trabajan con una configuración “open drain”, por lo que es necesaria la ubicación de resistencias de “pull-up” que de acuerdo a características como la polarización y la cantidad de dispositivos conectados, se calculan

tomando como referencia la hoja de especificaciones del dsPIC33FJMC802 [20] y del sensor de movimiento MPU9150[22].

TABLE 32-6: DC CHARACTERISTICS: I/O PIN OUTPUT SPECIFICATIONS

DC CHARACTERISTICS			Standard Operating Conditions: 3.0V to 3.6V (unless otherwise stated) Operating temperature -40°C ≤TA ≤+150°C for High Temperature				
Param.	Symbol	Characteristic	Min.	Typ.	Max.	Units	Conditions
DO10	VOL	Output Low Voltage I/O Pins: 2x Sink Driver Pins - RA2, RA7-RA10, RB10, RB11, RB7, RB4, RC3-RC9	—	—	0.4	V	IOL ≤1.8 mA, VDD = 3.3V See Note 1
		Output Low Voltage I/O Pins: 4x Sink Driver Pins - RA0, RA1, RB0-RB3, RB5, RB6, RB8, RB9, RB12-RB15, RC0-RC2	—	—	0.4	V	IOL ≤3.6 mA, VDD = 3.3V See Note 1
		Output Low Voltage I/O Pins: 8x Sink Driver Pins - RA3, RA4	—	—	0.4	V	IOL ≤6 mA, VDD = 3.3V See Note 1
DO20	VOH	Output High Voltage I/O Pins: 2x Source Driver Pins - RA2, RA7-RA10, RB4, RB7, RB10, RB11, RC3-RC9	2.4	—	—	V	IOL ≥ -1.8 mA, VDD = 3.3V See Note 1
		Output High Voltage I/O Pins: 4x Source Driver Pins - RA0, RA1, RB0-RB3, RB5, RB6, RB8, RB9, RB12-RB15, RC0-RC2	2.4	—	—	V	IOL ≥ -3 mA, VDD = 3.3V See Note 1
		Output High Voltage I/O Pins: 8x Source Driver Pins - RA4, RA3	2.4	—	—	V	IOL ≥ -6 mA, VDD = 3.3V See Note 1

TABLE 31-15: CAPACITIVE LOADING REQUIREMENTS ON OUTPUT PINS

Param No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
DO50	COSCO	OSC2/SOSCO pin	—	—	15	pF	In XT and HS modes when external clock is used to drive OSC1
DO56	CIO	All I/O pins and OSC2	—	—	50	pF	EC mode
DO58	CB	SCLx, SDAx	—	—	400	pF	In I²C™ mode
		t_r, SDA and SCL Rise Time	C_b bus cap. from 10 to 400pF		20+0.1C_b		300 ns
		t_f, SDA and SCL Fall Time	C_b bus cap. from 10 to 400pF		20+0.1C_b		300 ns
		t_{LOW}, SCL Low Period				1.3	
		t_{HIGH}, SCL High Period				0.6	

Imagen 38a. Electrical characteristics [20],[22]

Resistencia de "pull-up" mínima: Para calcular la resistencia de "pull-up" mínima, se emplea la ecuación que se encuentra a continuación, donde $\boldsymbol{V_{DDMAX}}$, es el voltaje de polarización máximo 3.3[V], $\boldsymbol{V_{OLMAX}}$ es el voltaje lógico en bajo máximo.

$$R_{PMIN} = \frac{(V_{DDMAX} - V_{OLMAX})}{I_{OL}}$$

$$\boldsymbol{R_{PMIN} = \frac{3.3[V] - 0.4\,[V]}{3mA}}$$

$$\boldsymbol{R_{PMIN} = 966.66\,\Omega}$$

Resistencia de "pull-up" máxima: Para calcular la resistencia de "pull-up" máxima, se emplea la ecuación que se encuentra a continuación, donde $\boldsymbol{C_b}$ es la capacitancia equivalente del bus cuando trabaja en modo rápido o "Fast-mode" por sus siglas en ingles y $\boldsymbol{t_r}$ es el intervalo de tiempo que se encuentra entre el 30% y 70% de la pendiente de subida de la forma de onda del protocolo I2C.

$$\boldsymbol{R_{PMAX} = \frac{t_r}{0.8473 * Cb}}$$

$$\boldsymbol{R_{PMAX} = \frac{300ns}{0.8473 * 400pF}}$$

$$\boldsymbol{R_{PMAX} = 820\Omega}$$

A partir de estas dos ecuaciones y revisando los valores comerciales se colocaron dos resistencias de *"pull-up"* con un valor de $\boldsymbol{1K\Omega}$

4.11.2. FRECUENCIA DE OPERACIÓN

La frecuencia de operación máxima sobre la cual se comunican los dos dispositivos conectados mediante el protocolo I2C, se calcula de la siguiente manera:

$$\boldsymbol{F_{clk} = \frac{1}{t_{low(\min)} + t_{high(\min)} + t_{r(actual)} + t_{f(actual)}}}$$

$$\boldsymbol{F_{clk} = \frac{1}{1,3\mu s + 0,6\mu s + 300ns + 300ns}}$$

$$\boldsymbol{F_{clk} = 400\ kHz}$$

Donde: $\boldsymbol{t_{low(\min)}}$ es el intervalo de tiempo en bajo mínimo de un periodo de reloj (SCLx), $\boldsymbol{t_{high(\min)}}$ es el intervalo de tiempo en alto mínimo de un periodo de reloj (SCLx), $\boldsymbol{t_{r(actual)}}$ es el intervalo de tiempo que se encuentra entre el 30% y 70% de la pendiente de subida de la forma de onda del protocolo I2C y $\boldsymbol{t_{f(actual)}}$ es el intervalo de tiempo para el mismo rango de porcentajes de la pendiente de bajada. Cada uno de los intervalos temporales mencionados con anterioridad se pueden identificar en la imagen 38b.

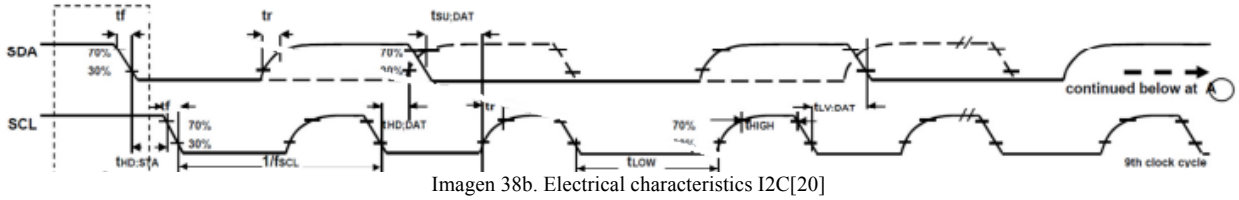


Imagen 38b. Electrical characteristics I2C[20]

Ya que el micro-controlador opera como maestro en el protocolo I2C, el modulo disponible para la comunicación entre los dispositivos conectados al bus debe estar en capacidad de generar el reloj SCLx. Normalmente los relojes del sistema están limitados a operar con frecuencias ($\boldsymbol{F_{SCL}}$) de 100 kHz, 400 kHz o 1 MHz.

Este reloj se genera con una velocidad definida por dos periodos BRG (TBRG). La tasa con la cual se genera el reloj del sistema normalmente esta dada por la suma entre el intervalo de tiempo mínimo en bajo del SCLx con el intervalo de tiempo mínimo en alto del SCLx.

El valor que se asigna en código es el que toma el registro I2CxBRG. Una vez este registro toma este valor el generador cuenta hasta cero y se detiene antes de que se cargue nuevamente de manera automática. El decremento se da de dos en dos por cada ciclo de instrucción ($\boldsymbol{T_{CY}}$).

I2CxBRG se calcula a partir de la ecuación que se encuentra enseguida. Este valor no debe estar por debajo de dos según la recomendación del fabricante[20].

$$\boldsymbol{I2CxBRG = \left[\left(\frac{1}{F_{SCL}} - PGD\right) * F_{CY}\right] - 2}$$

$\boldsymbol{F_{SCL}}$ es la frecuencia con la cual se comunican los distintos dispositivos (400 KHz según la hoja de especificaciones del sensor), PGD es un retardo que ocurre por cada pulso generado, que según la hoja de

especificaciones es de 130 ns y $\boldsymbol{F_{CY}}$ es la frecuencia de las instrucciones, que es la frecuencia del cristal externo dividida por dos.

$$\boldsymbol{I2CxBRG = \left[\left(\frac{1}{400kHz} - 130ns\right) * 16MHz\right] - 2}$$

$$\boldsymbol{I2CxBRG = 35.92 \approx 36}$$

4.11.3. I2C

La comunicación entre el micro-controlador y el sensor posee una estructura básica, bien sea para lectura o la escritura, tal y como se observa en los diagramas de flujo de la imagen 39 y 40:

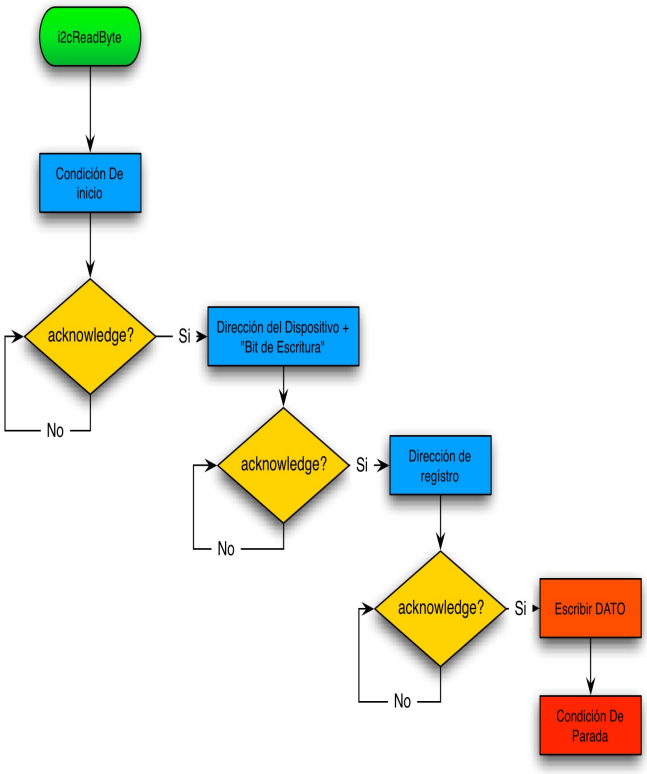


Imagen 39. Operación de lectura

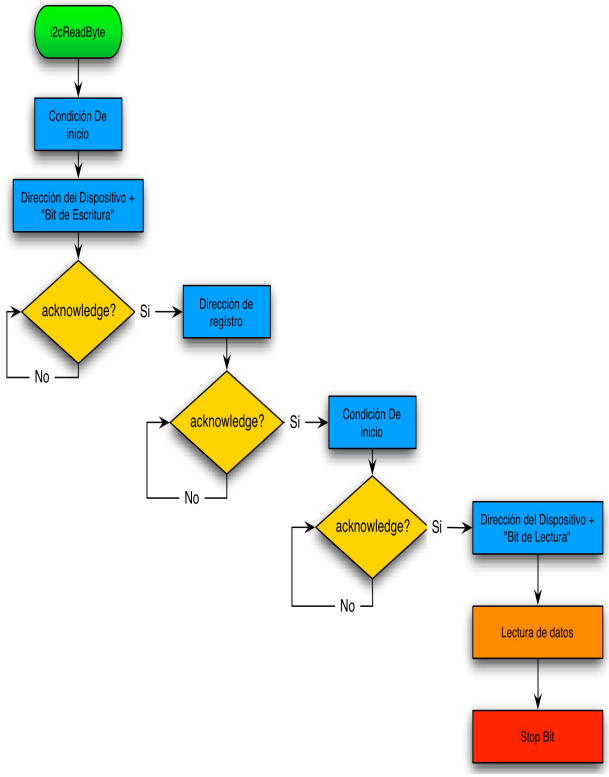


Imagen 40. Operación de escritura

Función escritura en los registros del sensor MPU9150:

```
unsigned char i2cWriteByte(unsigned char i2cADD,unsigned char ADD,unsigned char data)
        i2cStart();
                !i2cWriteAck((i2cADD<<1)|i2cWRITE);
                        !i2cWriteAck(ADD);
                                !i2cWriteAck(data);
                                        i2cStop();
};
```

```
Función de lectura de los registros del sensor MPU9150:

unsigned char i2cReadByte(unsigned char i2cADD,unsigned char ADD,unsigned char *data){
        i2cStart();
                !i2cWriteAck((i2cADD<<1)|i2cWRITE);
                        !i2cWriteAck(ADD);
                                i2cStart();
                                        !i2cWriteAck((i2cADD<<1)|i2cREAD);
                                                *data = i2cRead();
                                                        i2cNack();
                                                                i2cStop();
};
```

Estas dos funciones están compuestas por tres partes fundamentales. La primera es la dirección particular del sensor (0x68), la segunda es la dirección del registro al cual se quiere acceder y la tercera es la variable (apuntador) en donde se guarda el dato que se lee o, en el caso de la escritura, la variable que contiene el dato que se quiere escribir.

La función de escritura inicia la comunicación con un “Start-bit”, concatena los siete bits de dirección (del sensor MPU9150) con el bit de lectura ‘1’ o escritura ‘0’, espera un “acknowledge” y si la comunicación entre los dispositivos es exitosa, envía la dirección (ocho bits) del registro al cual se quiere acceder. Finalmente espera nuevamente un “acknowledge” y si la comunicación entre los dos dispositivos nuevamente es exitosa, para el caso de la escritura, se envía el paquete de ocho bits seguido de la condición de parada.

En la función de lectura se da la condición de inicio, se envía la dirección del sensor mas un bit de lectura ‘1’, se espera un “acknowledge”, se envía la dirección del registro al cual se quiere acceder, se espera otro “acknowledge” y finalmente se almacena en una variable los datos proveniente del sensor seguido de un “not-acknowledge” y una condición de parada. Se da o no un “acknowledge”, cuando el sensor no esta ocupado o cuando este responde al llamado que realiza el micro-controlador.

Teniendo en cuenta que el protocolo I2C permite tan solo el envió en tramas de ocho bits y que la resolución del sensor es de dieciséis (16) bits, se envían primero los 8 bits menos significativos (LOW) y luego los mas significativos (HIGH) los cuales se concatenan en un apuntador de enteros de dieciséis bits con los datos provenientes de cada sensor con respecto a X,Y,Z al igual que la temperatura..

Por ejemplo, si es la posición X del acelerómetro, se observa que cada dato viene en tramas de ocho bits. Por lo tanto, el apuntador toma la primer posición del vector que contiene los primeros ocho bits (LOW) de la posición X del acelerómetro, hace un corrimiento hacia la izquierda de ocho e inmediatamente concatena los otros ocho bits (HIGH) de dicha medida. Este proceso ocurre de igual forma para las medidas Y y Z, X,Y,Z del giróscopo, HIGH y LOW de la temperatura y X,Y y Z del magnetómetro.

De acuerdo a lo anterior y sabiendo que por cada medida X, Y o Z existen ocho bits para LOW y otros ocho para HIGH, es necesario un búfer de catorce posiciones para almacenar los datos provenientes del acelerómetro, giróscopo y temperatura (imagen 41) y uno de 6 posiciones para las medidas del magnetómetro (imagen 42). En la función se van concatenando los datos y a su vez incrementando el tamaño del apuntador.

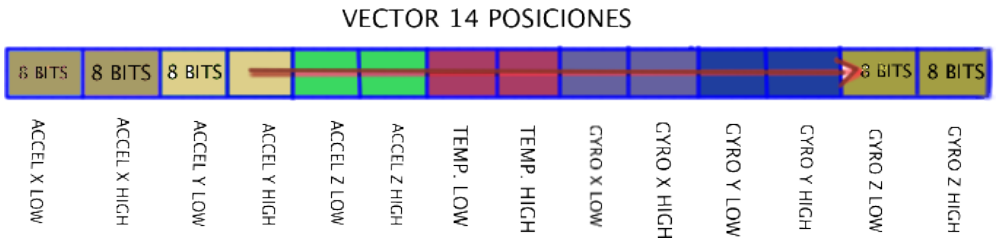


Imagen 41. Vector de posiciones medidas

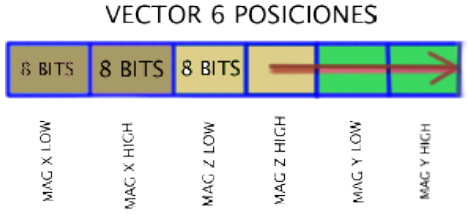


Imagen 42. Vector de posiciones medidas

```
unsigned char MPU9150_Read_RAW(int *data){
       buffer[14];
//acc X
              *data++ = ((buffer[0]<<8)|buffer[1]);
//acc Y
```

```
                *data++ = ((buffer[2]<<8)|buffer[3]);
//acc Z
                *data++ = ((buffer[4]<<8)|buffer[5]);
//Temp
                *data++ = ((buffer[6]<<8)|buffer[7]);
//Gyr X
                *data++ = ((buffer[8]<<8)|buffer[9]);
//Gyr Y
                *data++ = ((buffer[10]<<8)|buffer[11]);
//Gyr Z
                *data++ = ((buffer[12]<<8)|buffer[13]);
};

unsigned char MPU9150_Read_RAW_MAG(volatile int *data){
buffer[6];
        *data++ = (buffer[0]<<8)|buffer[1];
        *data++ = (buffer[4]<<8)|buffer[5];
        *data++ = (buffer[2]<<8)|buffer[3];
};
```

En el "main" se declara *data** como un vector de enteros de 10 posiciones (MPU[10]) en donde cada una contiene los 16 bits de las medidas X,Y y Z del acelerómetro, giróscopo y HIGH y LOW de temperatura. De igual forma se declara otro vector de enteros (MAG[3]) de tres posiciones, donde cada una contiene los 16 bits de las medidas X,Y y Z del magnetómetro. Lo anterior se relaciona en código de la siguiente manera:

```
MPU[0]= ACCEL HIGH & LOW X
MPU[1]= ACCEL HIGH & LOW Y
MPU[2]= ACCEL HIGH & LOW Z
MPU[3]= TEMP HIGH & LOW
MPU[4]= GYRO HIGH & LOW X
MPU[5]= GYRO HIGH & LOW Y
MPU[6]= GYRO HIGH & LOW Z

MAG[0]=MAG HIGH & LOW X
MAG[1]=MAG HIGH & LOW Z
MAG[2]=MAG HIGH & LOW Y
```

Por lo tanto si el usuario quiere visualizar los datos via Bluetooth debe realizarlo de la siguiente manera:

```
printf("ACCEL X \t");
printf("%f", (double) MPU[0]);  ACCEL HIGH & LOW X
```

Así para todos las medidas expuestas con anterioridad.

4.12. UART

El micro-controlador cuenta con un modulo UART que incluye un bloque que determina la tasa de transmisión del sistema (cuantos bits por segundo se transmiten) por medio de un "BaudRate Generator" (BRG) de 16 bits. El registro UxBRG controla el periodo de trabajo del temporizador de 16 bits. Las ecuaciones a continuación se emplean para encontrar el valor que se asigna a este registro en código, donde BRGH es el bit de selección entre "High speed mode" en '1' y "Low speed mode" en '0' ,Bit 3 del registro UxMode o Mode register, que configura y habilita o deshabilita el modulo UART.

$$\boldsymbol{BaudRate = \frac{F_{CY}}{16\,x(UxBRG+1)}; \; con\; BRGH = 0}$$

$$\boldsymbol{UxBRG = \frac{F_{CY}}{16x(BaudRate+1)};\; F_{CY} = \frac{F_{osc}}{2} = \frac{32MHz}{2} = 16MHz}$$

$$BaudRate - deseado = 115200$$

$$BaudRate - deseado = \frac{F_{CY}}{16\,x(UxBRG + 1)}$$

$$UxBRG = \frac{(F_{CY}/BaudRate - deseado)}{16} - 1$$

$$UxBRG = \frac{16MHz/115200}{16} - 1$$

$$UxBRG = 7,6805$$

$$BaudRate - calculado = \frac{F_{CY}}{16\,x(UxBRG + 1)}$$

$$BaudRate - calculado = \frac{16MHz}{16\,x(7,68 + 1)}$$

$$BaudRate - calculado = 115207,37$$

$$Error = \frac{BaudRatecalculado - BaudRatedeseado}{BaudRatedeseado}$$

$$Error = \frac{115207,37 - 115200}{115200} = 0.0063\%$$

$$BaudRate = \frac{F_{CY}}{4\,x(UxBRG + 1)}; con\ BRGH = 1$$

$$UxBRG = \frac{F_{CY}}{4x(BaudRate)} - 1$$

$$UxBRG = \frac{F_{CY}}{4x(BaudRatedeseado)} - 1$$

$$UxBRG = \frac{16MHz}{4x(115200)} - 1 = 33,72$$

$$BaudRate = \frac{16MHz}{4\,x(33,72 + 1)} = 115207,37$$

$$UxBRG = 33,72$$

Las funciones que se encuentran a continuación se emplean para transmitir un dato de tipo entero o un dato de tipo carácter:

```
void UART_Put_Char(int c)         // Dato de envió entero.
void UART_Put_String(char *s) // Envió cadena de caracteres.
```

4.13. PWM

La frecuencia a la que opera este PWM es de 1,2KHz y tiene como objetivo que el ruido generado por los motores pueda ser filtrado por los condensadores instalados sin que estos ocupen mucho espacio en el circuito. El registro ó PxTPER, establece la frecuencia de trabajo a la que opera PWM, a partir de la ecuación que se encuentra a continuación:

$$\boldsymbol{PxTEPER = \frac{F_{cy}}{F_{PWM} * PxTMR\ Prescaler} - 1}$$

Donde:
Fcy: Frecuencia de trabajo de las instrucciones, 16 MHz o frecuencia del oscilador externo o reloj interno dividido por dos.
Fpwm: frecuencia a la que se desea que trabaje el PWM
PxTMR Prescaler: 1:1 uno a uno.

$$\boldsymbol{PxTEPER = \frac{16MHz}{1200 * 1} - 1}$$

$$\boldsymbol{PxTEPER = 13333,3}$$

$$\boldsymbol{PxTEPER \approx 13332}$$

El modulo relacionado con el valor del ciclo útil, es el PxDCy.. El micro-controlado seleccionado tiene dos módulos PWM con salidas high" y "low". El primer módulo tiene 3 pares high/low y el segundo módulo un par.

Las salidas "high" y "low", se pueden utilizar de forma independiente, sin embargo solamente se están utilizando las salidas "high" de ambos módulos. Esto anterior se logra colocando en uno '1' del bit once al ocho del registro PWMxCON1,teniendo en cuenta el par de salidas que se desea configurar de esa manera, tal y como se muestra a continuación:

```
PWM1CON1bits.PMOD3 = 1; // PWM Modulo 1: High1 Y Low1 en modo independiente
PWM1CON1bits.PMOD2 = 1; // PWM Modulo 1: High2 Y Low2 en modo independiente
PWM1CON1bits.PMOD1 = 1; // PWM Modulo 1: High3 Y Low3 en modo independiente
PWM2CON1bits.PMOD1 = 1; // PWM Modulo 2: High1 Y Low1 en modo independiente
```

4.14. CALCULO DE RESISTENCIA DE LA ETAPA DE POTENCIA

La resistencia en base de los transistores para la configuración particular que se muestra en el capitulo 3.4, se calcula con el fin de garantizar que el transistor actúe como un interruptor, trabajando entre la región de corte y saturación. En corte, la corriente de colector debe ser mínima, el voltaje colector-emisor debe ser máximo y el voltaje base emisor debe ser mínimo. Para saturación, la corriente de colector debe ser máxima, el voltaje colector-emisor debe ser mínimo (alrededor de 200 mV) y el voltaje base-emisor debe ser máximo.

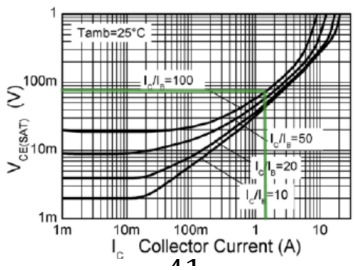


Imagen 43. Beta de 100 corriente de 1.6 A [23]

De acuerdo a lo anterior, para garantizar que el transistor esté en región de saturación, se debe obtener la relación entre la corriente máxima de colector, que es la que piden los motores a máxima potencia (permitiendo un voltaje en colector mínimo de alrededor de 200 mV), y la corriente de base, para encontrar el beta ($\boldsymbol{\beta}$) mínimo, al menos inferior a la región de operación lineal en las condiciones en las que trabaja el sistema. Calculando se obtiene:

$$\boldsymbol{I_{b(sat)} = \frac{I_{cmax}}{\beta sat}} \quad ; \quad \boldsymbol{I_{b(sat)} = \frac{1,62[A]}{376}}$$

$$\boldsymbol{I_{b(sat)} = 4,31mA}$$

Con esta corriente se debe calcular la resistencia en base observando la grafica de voltaje de saturación, $\boldsymbol{V_{BE(SAT)}}$, contra corriente de colector, $\boldsymbol{I_c}$, (imagen 44) y realizando el siguiente cálculo:

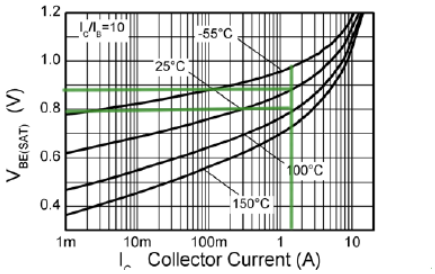


Imagen 44. Vbe (sat) vs corriente de colector [23]

$$\boldsymbol{R_B = \frac{3,3V - V_{BE(SAT)}}{I_{b(sat)}}} \quad ; \quad \boldsymbol{R_B = \frac{3,3V - 880mV}{4,31\ mA}}$$

$$\boldsymbol{R_B = 561,48\Omega}$$

El valor aproximado comercial que fácilmente se adquiere es de 560 **Ω**.

4.15. MANUAL DE USUARIO

El manual es la relación final del equipo con el usuario y está dividido en cuatro grandes capítulos con un total de 26 páginas. En el primer capítulo se hace una breve descripción del equipo y se incluye su propósito especifico, público objetivo e imágenes del sistema.

El segundo capítulo instruye sobre la manipulación de la plataforma como el tiempo de carga de las baterías, la autonomía de vuelo, la puesta en marcha del sistema, la conexión física con un computador y la programación del micro-controlador usado a través del compilador

El tercer capítulo se concentra en los parámetros mecánicos del vehículo y describe de manera resumida lo que es un cuadrotor, continuando con la tabla donde se incluyen las dimensiones mas relevantes del equipo como el peso total y el peso de algunas partes que lo integran.

El cuarto capítulo, el mas importante, se concentra en el sistema embebido, mostrando el diagrama en bloques, el esquemático general, una descripción general de los diferentes dispositivos (micro-controlador, sensor, Bluetooth, etc.), los pines disponibles después de la conexión mínima requerida, los distintos pines empleados con su función particular y finalmente el procedimiento básico para la configuración y manipulación de la plataforma, incluyendo las funciones mas importantes del protocolo I2C, UART y PWM. Además hay dos sub-capítulos donde se describe la seguridad mínima requerida al momento de manipular la plataforma y los procedimientos de mantenimiento.

5. ANÁLISIS DE RESULTADOS:

5.1. RESULTADO DEL IMPRESO

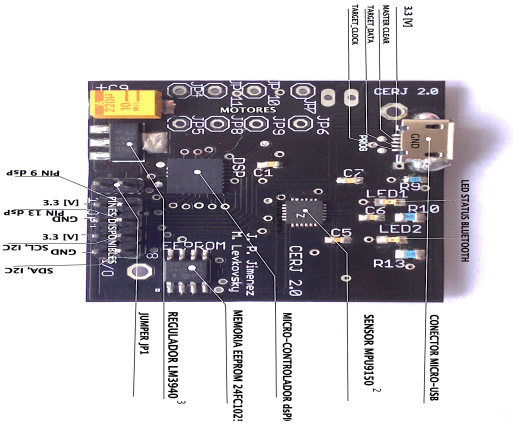
El resultado del diseño del impreso que aloja todo el sistema embebido fue el esperado y cumple con las proporciones, el peso y los requerimientos que se plantearon desde un principio. Todo el diseño estuvo sujeto a una serie de condiciones, pensando en el usuario final y en la ubicación del circuito a bordo del equipo para no afectar las características aerodinámicas del vehículo.

Una de las condiciones de diseño era que el circuito impreso debía caber en el área efectiva de 44 mm por 44 mm, espacio destinado para instalar el sistema embebido del equipo. Fue necesario calcular el punto donde debía ubicarse el conector de las baterías para que no quedara en el mismo punto por donde pasa la estructura mecánica del equipo. Los componentes de mayor relevancia debían ubicarse en la capa superior, al igual que los pines disponibles (de tal forma que no queden en contacto con la estructura) y el conector micro USB. El peso no superó los valores esperados ya que debía ser lo mas parecido al peso del circuito original, el cual era de 6.1g.

El impreso final posee las características presentadas en la tabla 8. Las imágenes 45 y 46 muestran dos vistas del impreso con indicando los componentes mas importantes.

Característica	Valor
Peso	7 gramos
Dimensiones	39 mm x 39 mm
Numero de componentes	35 componentes smt
Grosor	0.8 milímetros
Numero de capas	2
Ancho pista	1 oz de cobre
Resolución	8 Mil

Tabla 8 . Características de circuito



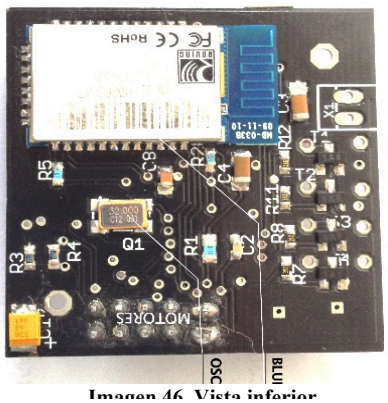


Imagen 46. Vista inferior

Sin contar las resistencias condensadores, transistores, el oscilador externo y el conector micro-USB, los componentes contenidos en el impreso son:

1	dsPIC33FJ138MC802	Micro-controlador, 128 Kbyte de memoria Flash; 16 Kbyte memoria RAM; 1 modulo I2C @ 400 KHz, '*fast mode'*; 4 módulos PWM (PWM1H1/L1, PWM1H2/L2, PWM1H3/L3, PWM2H1/L1); Oscilador externo 32 MHz; frecuencia instrucción 16 MHz.
2	Sensor MPU9150	Sensor que combina acelerómetro de tres ejes, giróscopo de tres ejes y magnetómetro de tres ejes; conectado por medio de I2C @ 400 KHz.
3	Regulador LM3940	Regulador de voltaje, voltaje de entrada 7.5 [v]; voltaje de salida 3.3 [V]; corriente máxima de salida 1 [A]; *'Dropout Voltage'* 1.0 [V] o 0.8 [V] @ Iout= 1[A]
4	Memoria EEPROM	1024 K bit de memoria no volátil adicional para el sistema; conectado I2C @ 400 KHz.
5	BLUETOOTH RN42 [3]	Bluetooth RN42 , 2,4 GHz, conectado por UART.

5.2. ESTRUCTURA MECÁNICA

Las dimensiones mecánicas y pesos del vehículo luego de las modificaciones hechas se presentan en la tabla 9.

SÍMBOLO	PARÁMETRO	MEDIDA	UNIDADES
A	Longitud brazo	0,20	m
Ptot	Peso total	0.068	Kg
Phel	Peso hélice	0,00750	Kg
B	Diámetro hélice	0.0683	m
Pbat	Peso batería	0.020108	Kg
D	Dimensiones Baterías	0.04*0.02*0.015	m
Af	Área efectiva	0.044*0.044	m
Pcn	Peso circuito nuevo	0.007	Kg
Atot	Altura rotor-base	0.0288	m
Emp	Empuje	**1309,089**	N

Tabla 9. Pesos y medidas

5.3. TIEMPO DE CARGA Y TIEMPO DE USO

Tras la instalación de las nuevas baterías se determino nuevamente el tiempo de carga y el tiempo de uso a máxima potencia por medio de un cronómetro, como se muestra en la tabla 9a.

Tiempo de carga aproximado (minutos) ($\boldsymbol{Tca}$)	20.34
Tiempo de uso a máxima potencia aproximado (minutos) Autonomía de vuelo ($\boldsymbol{Tvu}$)	3,30

Tabla 9a. Carga y Uso

5.4. MEDICIÓN DEL EMPUJE CON LA MODIFICACIÓN

A partir del mismo experimento que se realizo en el capítulo 4.3, se encontró que luego de las modificaciones hechas, esto sucedió con un peso en monedas de 65,49 gramos (g) o 0,06549 kilogramos (kg) (11 monedas de 100 pesos y 1 de 200 pesos), con lo que el empuje se calcula sumando este peso con el peso total del equipo y ese total multiplicándolo por la aceleración de la gravedad que en Bogotá es de un promedio de 9,80665 $\frac{\boldsymbol{m}}{\boldsymbol{s^2}}$ [32].

$$\boldsymbol{Empuje} = (\boldsymbol{Peso\ adicional} + \boldsymbol{Peso\ total}) * \boldsymbol{aceleración\ gravedad\ Bogotá}$$

$$\boldsymbol{Empuje} = (\boldsymbol{65,49g} + \boldsymbol{68g}) * \boldsymbol{9,80665}\ \frac{\boldsymbol{m}}{\boldsymbol{s^2}}$$

$$\boldsymbol{Empuje} = \boldsymbol{1309,089}$$

5.5. TRANSMISIÓN INALÁMBRICA

Una de las ventajas con las que cuenta la plataforma es la capacidad de transmitir datos de forma inalámbrica por medio del Bluetooth, selección que se hizo con base en algunos criterios mostrados en el capitulo 3.2 del presente libro Se implemento un código de prueba en el que, con la plataforma conectada a un computador, se visualiza en un monitor de puerto serial las variables que provienen de los sensores, como lo son las medidas X,Y,Z que detecta el acelerómetro, giróscopo y magnetómetro, tal y como lo muestra la imagen 47.

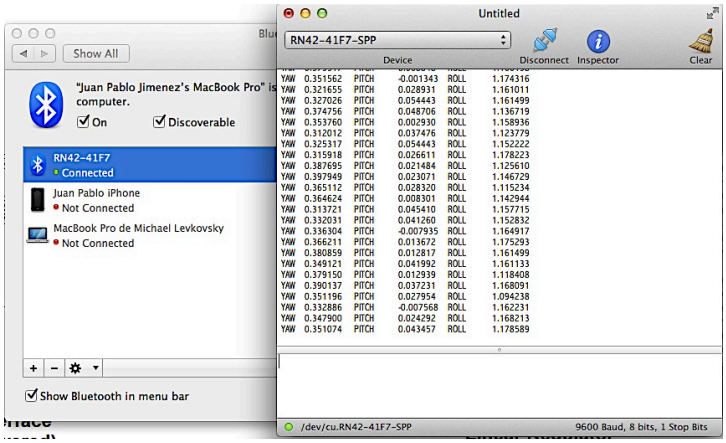


Imagen 47. Prueba transmisión inalámbrica

Para verificar la utilidad del método de transmisión seleccionado se implementó un código que envía una letra desde un ordenador hacia la plataforma y esta la retorna inmediatamente. Este código de prueba también sirve para demostrar la capacidad de tele comando del sistema. Si se envía una determinada letra ésta modifica el ciclo útil de PWM de los cuatro motores de la siguiente forma: a=0%; b=25%; c=50%; d=80%; e=90%. El soporte experimental y el código de esta prueba se encuentra en un CD que se entrega con este documento.

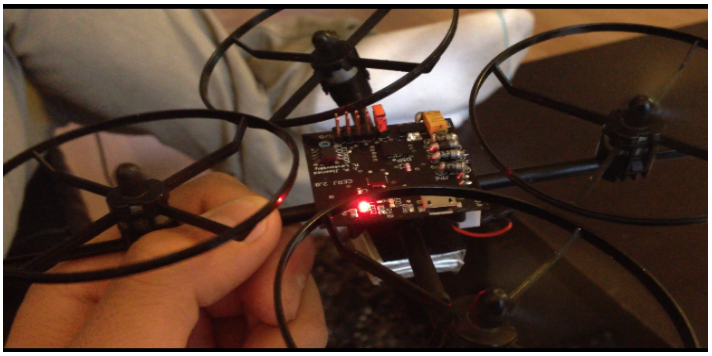


Imagen 48. Prueba de Funcionamiento

5.6. CONSUMO DE MEMORIA

Como se explicó en el capítulo 4.7, se compilaron dos algoritmos de control para cuadricópteros. El primero tiene el nombre de QUAD_1.5, que es un control PID basado en un código muy usado en la comunidad de DIY conocido como ArduIMU. Este código recibe las variables de movimiento (giróscopo, acelerómetro y magnetómetro), las convierte a ángulos de Euler y por medio del control varia la potencia entregada a los motores a partir de PWM,. Este código cuenta además con una sección que permite controlar el vehículo por medio de un mando a distancia o de programar rutas por medio de GPS.

Este código ocupa 18872 Bytes de memoria al ser compilado, lo que extrapolado a la capacidad del micro-procesador seleccionado para la plataforma de desarrollo, equivaldría al 14.74375% de la capacidad de memoria total del dispositivo.

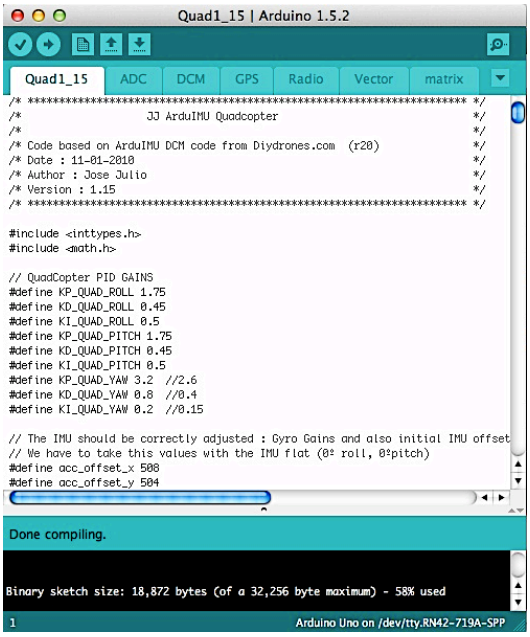


Imagen 49. compilación Quad 1.5

El segundo código que se revisó es el mas conocido y utilizado en el mudo de los multi-cópteros; el Arducopter [51]. Este es un completísimo sistema que, además de ser un control de estabilidad, incluye muchísimas características adicionales, tales como soporte para helicópteros, tricópteros, cuadricópteros, hexacópteros y octocópteros, control de altitud, programación de rutas y comportamientos por GPS, sistema de transmisión bidireccional de ordenes por protocolo Mavlink, despegue y aterrizaje automático, terminal de comandos en línea y control para estabilización de cámaras abordo, entre otras.

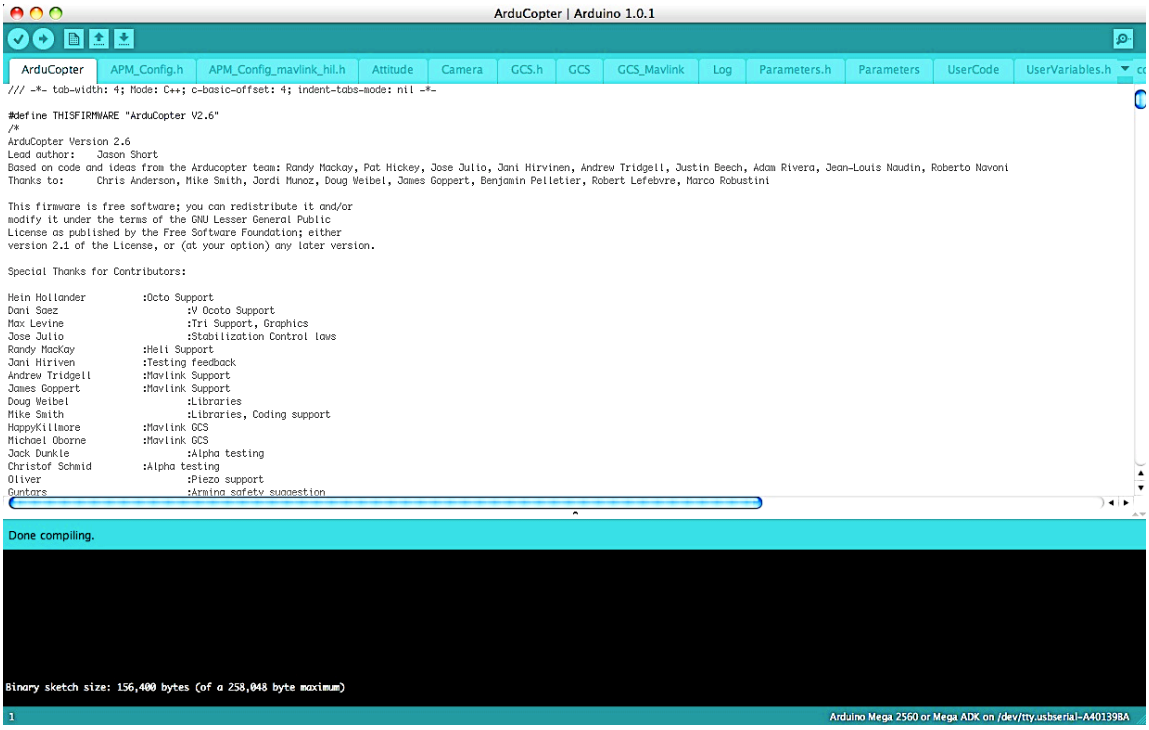


Imagen 50. Compilación Arducopter[35]

Este código se compilo y se encontró que ocupa 156400 Bytes lo que extrapolado a la capacidad del micro-procesador seleccionado para la plataforma, equivaldría al 122.1875% de la capacidad de memoria total del dispositivo. A pesar de que este código supera la capacidad de memoria de la plataforma desarrollada, también se puede ver que las funcionalidades superan ampliamente las aplicaciones básicas que se espera va a necesitar la plataforma. Aún siendo un código tan completo, el tamaño excede por poco a la capacidad de memoria de la plataforma por lo que es posible asegurar un algoritmo de características similares pueda ser implementado.

Se analizaron documentos IEEE de algunos proyectos en donde se utilizaron sistemas electrónicos similares y se implementaron algoritmos de estabilidad. Estos documentos ya fueron mencionados y tenidos en cuenta a la hora de seleccionar el micro-controlador en el capitulo 3.1 del presente libro [17],[18] y [19], en donde se determino que el algoritmo mas grande ocupaba como máximo una memoria de 128 KB y se observo que la tendencia era usar 32 KB de memoria flash.

Finalmente el código de funcionamiento mínimo relacionado con I2C, UART y PWM ocupa un 15% de la memoria flash disponible (ver imagen 50a) dejando libre un 75 %, para posibles algoritmos de control tales como los compilados anteriormente.

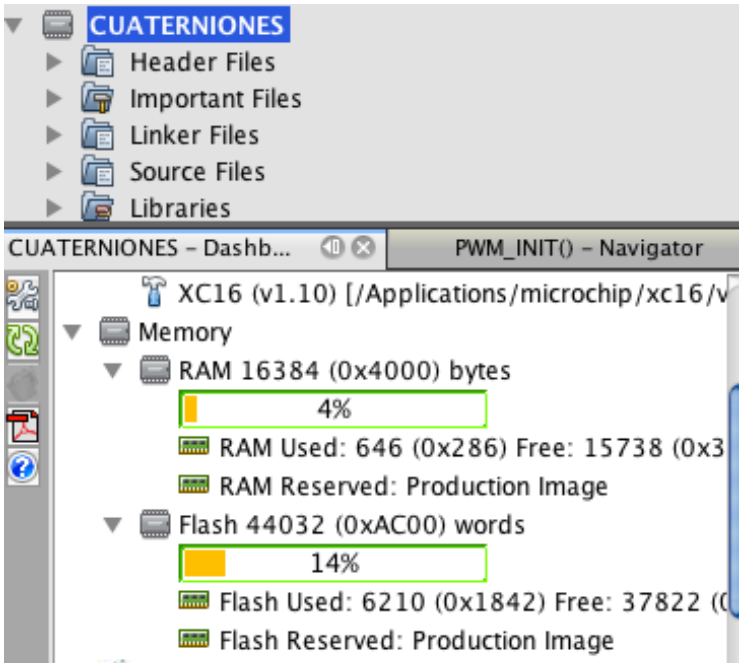


Imagen 50a. Memoria empelada

5.7. CALCULO DEL CICLO DE TRABAJO

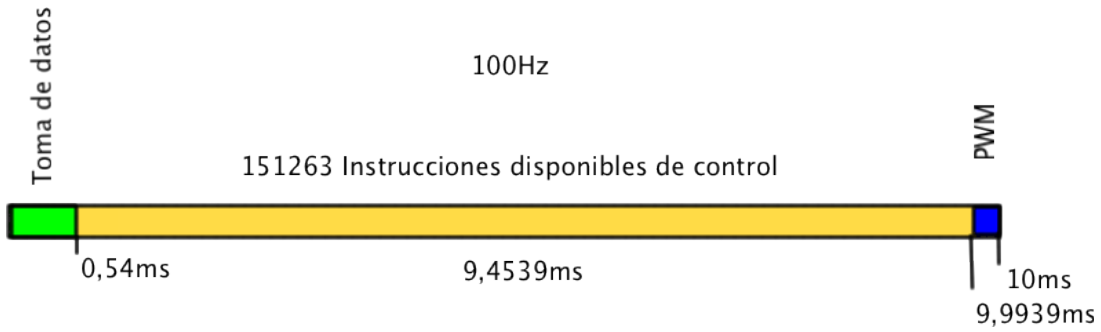


Imagen 51a. Ciclo de trabajo CERJ-1

De acuerdo con el documento *"Control of Quadrotor for robust perching and landing"*[41], de la Universidad de Pennsylvania, el sistema debe control de cuadrotóres debería estar en capacidad de realizar todo el ciclo de control en al menos 10 ms (100 Hz). No es posible medir en la práctica el tiempo de ejecución sin haber seleccionado previamente un algoritmo especifico y escrito el código para la plataforma (lo cual está fuera del alcance del proyecto). Sin embargo, para validar la capacidad de procesamiento del micro-controlador seleccionado se puede hacer un cálculo de cuanto tardaría en tomar datos, realizar las operaciones de control pertinentes y transmitir las variables.

Toma de datos:

En código, a partir del tercer acknowledge, el sensor envía todos los datos en modo continuo (X,Y y Z del acelerómetro, X,Y y Z del giróscopo, temperatura y finalmente X,Y y Z del magnetómetro) sin iniciar nuevamente la secuencia, enviando de ocho bits en ocho bits con un *"acknowledge"* entre cada byte, almacenándolos en un buffer y finalizando con el *"not-acknowledge"* y la condición de parada luego de recibir todos los datos, tal y como se muestra a continuación.

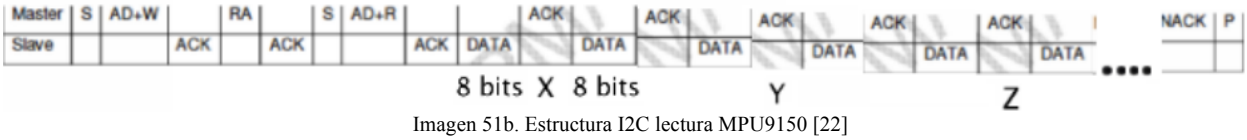


Imagen 51b. Estructura I2C lectura MPU9150 [22]

El micro-controlador tarda un tiempo determinado en enviar y recibir toda la secuencia (trama para la lectura de todos los datos de manera continua). Para calcular ese tiempo se debe multiplicar dicha trama (imagen 51b.) a partir del tercer acknowledge del esclavo (sensor) por tres (acelerómetro, giróscopo y magnetómetro), teniendo en cuenta que en modo continuo se realiza la lectura de todas las medidas de posición X (16 bits),Y (16 bits) y Z (16bits) para el acelerómetro, giróscopo y magnetómetro con un acknowledge por cada byte (8 bits), sumando además 16 bits de las medidas HIGH y LOW de temperatura mas dos acknowledge por cada 8 bits. A esto anterior se debe sumar los 36 bits que hacen parte de la secuencia inicial hasta el tercer acknowledge (tomando la condición de inicio como un bit, la dirección de 7 bits, bit de escritura o lectura, bit de acknowledge y dirección particular de registro al que se quiere acceder de 7 bits).

Si se sabe que el periodo de un solo bit (de I2C a 400KHz) es de 2,5µs, se multiplica este tiempo por el número de bits transmitidos de todas las medidas, mas los bits de la trama del protocolo I2C, como el *"Start bit", "Stop bit", "acknowledge",* etc., siendo aproximadamente 216 bits, para un tiempo total de

0,54ms. Este tiempo es lo que teóricamente tardaría la etapa de recepción de datos desde el sensor hacia micro-controlador.

$$t_{I2C} = Nobitstrasmitidos - serial * T_{bit}$$

$$t_{I2C} = 216 * 2,5\mu s$$

$$t_{I2C} = 0,54\ ms$$

PWM:

El tiempo que toma en verse reflejado un cambio del ciclo útil del PWM, depende del tiempo que toma reescribir el registro PxDCy, siendo de un ciclo de instrucción (62,5ns) y el tiempo muerto o retardo que genera el modulo de PWM, que según la imagen 51c, asumiendo que las instrucciones trabajan a 10 MHZ con un pre-escalizdor 1;1, es de 6 µs en el peor de los casos.

TCY (FCY)	Prescaler Selection	Resolution	Dead Time Range
25 ns (40 MHz)	1 TCY	25 ns	25 ns – 1.6 µs
25 ns (40 MHz)	4 TCY	100 ns	100 ns – 7 µs
50 ns (20 MHz)	4 TCY	200 ns	200 ns – 12 µs
100 ns (10 MHz)	2 TCY	200 ns	200 ns – 12 µs
100 ns (10 MHz)	1 TCY	100 ns	100 ns – 6 µs

Imagen 51c. Rango Delay PWM [20]

Algoritmo de control:

Luego de obtener ambos valores de forma teórica se deben restar al tiempo que debe tardar el ciclo de control, 10 ms y así saber el tiempo con el que cuenta el micro-controlador para poder ejecutar las demás instrucciones del control. Si se establece un tiempo promedio por instrucción, se podría determinar la cantidad máxima de instrucciones que puede ejecutar el algoritmo de control en ese tiempo.

El tiempo disponible para el algoritmo resulta ser de 9,4539 , ms, la frecuencia del procesador es de 16 MHz (T= 62,5 ns) y se puede considerar que, en promedio, una operación tarda un ciclo de reloj [20], Se divide el tiempo (9,4539 ms) entre el tiempo que tarda un ciclo de reloj y así se sabe que el número de instrucciones que se pueden ejecutar es 151263 . Si se supone que cada instrucción tarda en promedio dos ciclos de tarea, se reduciría el numero de instrucciones que se pueden ejecutar a 75632..

$$t_{restante} = 10ms - t_{UART} - t_{I2C}$$

$$t_{restante} = 10ms - (6\mu s + 65,2ns) - 0,54\ ms$$

$$t_{restante} = 9,4539\ ms$$

$$Noinstrucciones = \frac{t_{restante}}{Tinstrucciones}$$

$$Noinstrucciones = \frac{9,4539\ ms}{62,5\text{ns}} = 151263$$

Base Instr #	Assembly Mnemonic	Assembly Syntax		Description	# of Words	# of Cycles	Status Flags Affected
1	ADD	ADD	Acc	Add Accumulators	1	1	OA,OB,SA,SB
		ADD	f	f = f + WREG	1	1	C,DC,N,OV,Z
		ADD	f,WREG	WREG = f + WREG	1	1	C,DC,N,OV,Z
		ADD	#lit10,Wn	Wd = lit10 + Wd	1	1	C,DC,N,OV,Z
		ADD	Wb,Ws,Wd	Wd = Wb + Ws	1	1	C,DC,N,OV,Z
		ADD	Wb,#lit5,Wd	Wd = Wb + lit5	1	1	C,DC,N,OV,Z
		ADD	Wso,#Slit4,Acc	16-bit Signed Add to Accumulator	1	1	OA,OB,SA,SB

Imagen 52. INSTRUCTION SET OVERVIEW dsPIC33 [20]

UART:

En el caso de la transmisión de datos a través de UART solo es relevante cuando el control se haga de forma externa, donde el tiempo que toma hacer la transmisión y recepción de datos afecta el ciclo de control.

Por lo tanto, sabiendo que UART transmite a 115200 baudios, se puede calcular el tiempo de bit:

$$\boldsymbol{Tiempo\ de\ bit} = \frac{\mathbf{1}}{\boldsymbol{BaudRate}}$$

$$\boldsymbol{Tiempo\ de\ bit} = \frac{\mathbf{1}}{\mathbf{115200}}$$

$$\boldsymbol{Tiempo\ de\ bit} = \mathbf{8,68}\mu s$$

Con este valor y asumiendo que se transmitirán 16 bits o dos bytes correspondientes a la medida HIGH Y LOW de cada una de las coordenadas o de la medición de temperatura, mas el bit de inicio y de parada por cada trama consecutivamente para el acelerómetro, giróscopo y magnetómetro, se debe multiplicar el tiempo por 180, que equivale a todas las medidas (X,Y y Z del acelerómetro, X,Y y Z del giróscopo, temperatura y finalmente X,Y y Z del magnetómetro) bien sea de movimiento o de temperatura que detecta el sensor y que se transmitirían por UART, dando como resultado 1,5624 ms.

$$t_{UART} = Nobitstransmitidos - serial * Tiempo\ de\ bit$$

$$t_{UART} = 180 * 8,68\mu s$$

$$t_{UART} = 1,5624\ ms$$

5.8. PINES DISPONIBLES

Luego de realizar las conexiones necesarias para la correcta operación del sistema, los pines disponibles para el usuario son:

- Pin13 y Pin9: Estos son pines de propósito general.
- Pin 13: Pin de propósito general y se puede utilizar como interrupción externa.
- Pines de I2C: SDA (datos del protocolo I2C) y SCLx (reloj del protocolo I2C), que permiten conectar en paralelo hasta seis dispositivos mas por medio de este protocolo.

Adicionalmente los pines empleados para la programación de micro-controlador también pueden ser empleados como pines de propósito general. Estos se pueden acceder a través de un conector de programación micro-USB. Los pines tienen la distribución que se muestra en la imagen 53a y 53b

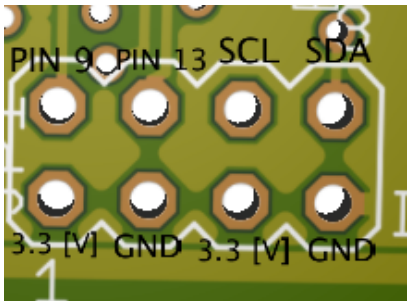


Imagen 53a. Pines disponibles

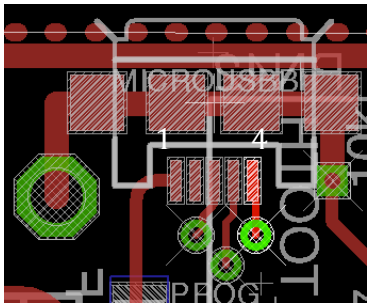


Imagen 53b. Pines disponibles micro-USB

5.9. PROTOCOLO I2C, LA UART Y EL PWM

Luego de implementar los códigos mencionados en el capitulo de desarrollos sobre I2C, UART y PWM en MPLAB® y de programar el micro-controlador, se observaron, a través de un osciloscopio, las señales de la trama de la línea de datos o SDA la línea del reloj o SCLx, UARTRx y PWM en un instante de tiempo determinado, tal y como lo demuestran las imágenes 54 y 55.

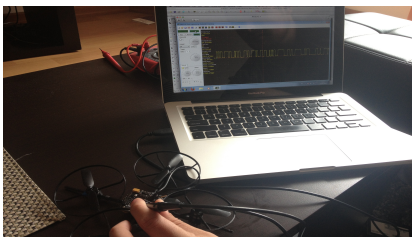


Imagen 54. Prueba I2C, UART y PWM-1

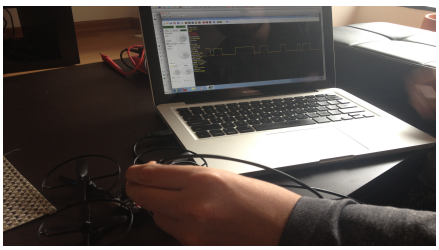


Imagen 55. Prueba I2C, UART y PWM-2

5.9.1. RESULTADOS SOBRE I2C

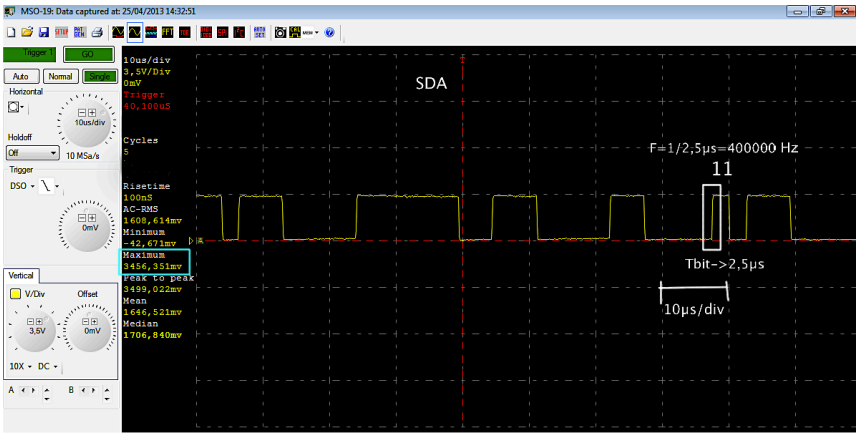


Imagen 56. SDA, Osciloscopio

En la imagen56 se observa que la comunicación entre el micro-controlador y el sensor es exitosa, trasmitiendo a una frecuencia de 400 KHz, (teniendo en cuenta que la resolución para este caso es de 10µs/div) con un voltaje lógico alto de 3,456 Voltios. Cabe anotar que es muy difícil tratar de determinar el orden de la secuencia o trama de bits, sin embargo no hay duda que se están trasmitiendo las medidas X,Y y Z, tal y como se identifica en la sección de transmisión inalámbrica, tras observar en un monitor de puerto serial lo que detecta el sensor.

A continuación se observa la línea del reloj o SCLx donde se puede ver el periodo de la secuencia y su respectivo voltaje en alto. En esta oportunidad fue mas sencillo identificar que la señal de reloj se esta comportando como se espera, transmitiendo en secuencias de 9 bits.

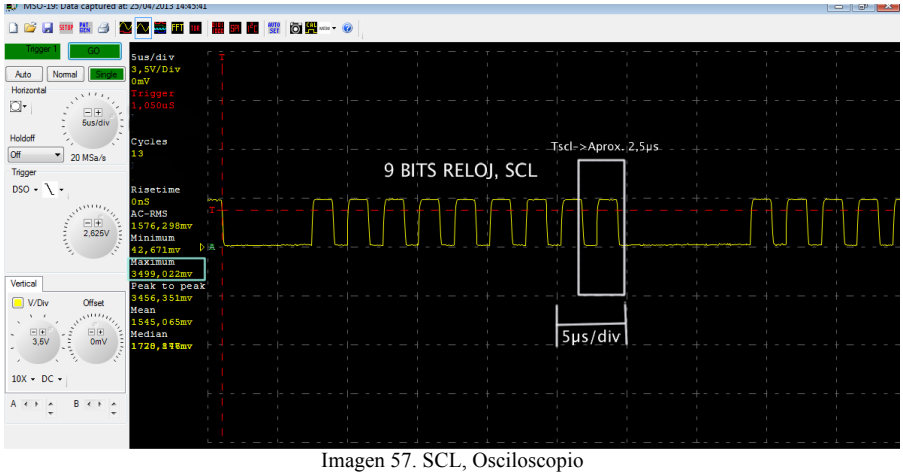


Imagen 57. SCL, Osciloscopio

Nuevamente y teniendo en cuenta que el osciloscopio tiene una resolución de 5µs/div, se calcula un periodo de bit de aproximadamente 2,5 µs (frecuencia de 400 KHz), lo que demuestra que la frecuencia para el protocolo se ajusta a las condiciones de configuración para *"fast mode"*, con un voltaje máximo de 3,499 Voltios. De acuerdo a las especificaciones del protocolo la forma de la trama del reloj se ajusta a las condiciones del protocolo transmitiendo secuencia de 9 bits por cada 8 bits trasmitidos en la línea SDA siendo el 9 bit, la condición de *"acknowledge"* o *"not-acknowledge"*.

5.9.2. RESULTADOS SOBRE UART

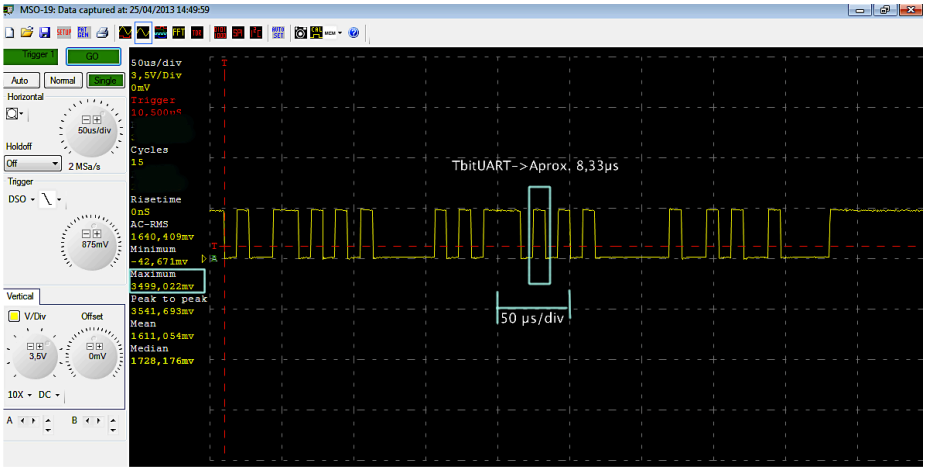


Imagen 58. UART, Osciloscopio

Teniendo en cuenta que la trasmisión se da a 115200 Baudios y que teóricamente el tiempo de bit es de 8,68 µs, la imagen 58 arroja que particularmente el tiempo de bit es de aproximadamente 8,33 µs (con una resolución de 50µs/di) y el voltaje máximo en alto de cada bit transmitido es de 3,499 Voltios.

5.9.3. RESULTADOS PWM

Luego de elaborar el código en MPLAB ®, el cual al enviar una determinada letra a través del monitor de puerto serial se puede cambiar el ciclo útil de los cuatro motores simultáneamente Si se envía la letra ‘b’ el ciclo útil cambia al 25%, como se observa en la imagen 59.

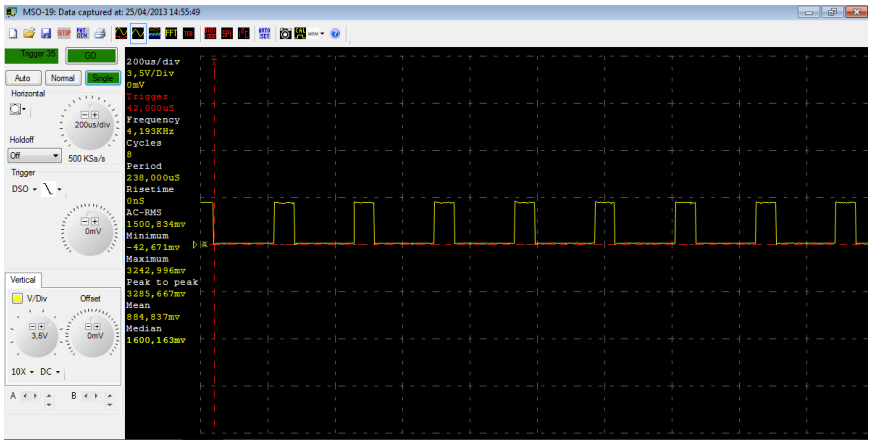


Imagen 59. PWM 25%

Si se envía la letra ‘c’ el ciclo útil cambia simultáneamente en los cuatro motores al 50% como se muestra en la imagen 60.

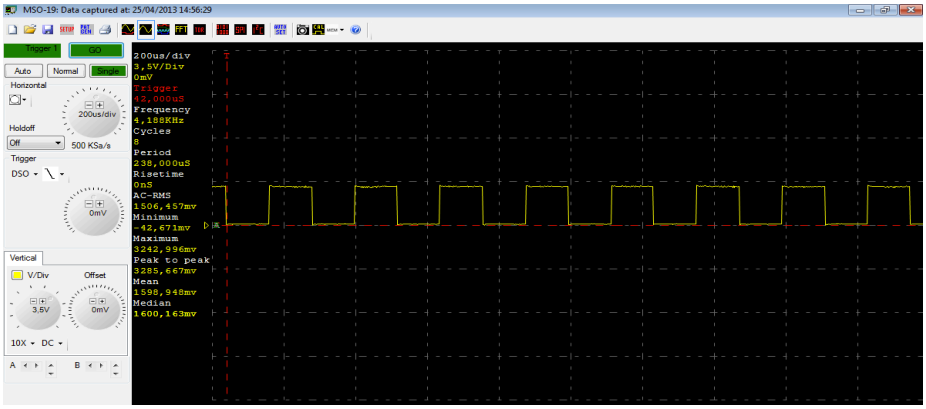


Imagen 60. PWM 50%

Si la letra enviada es ‘d’ el ciclo útil cambia al 80% como se ve en la imagen 61.

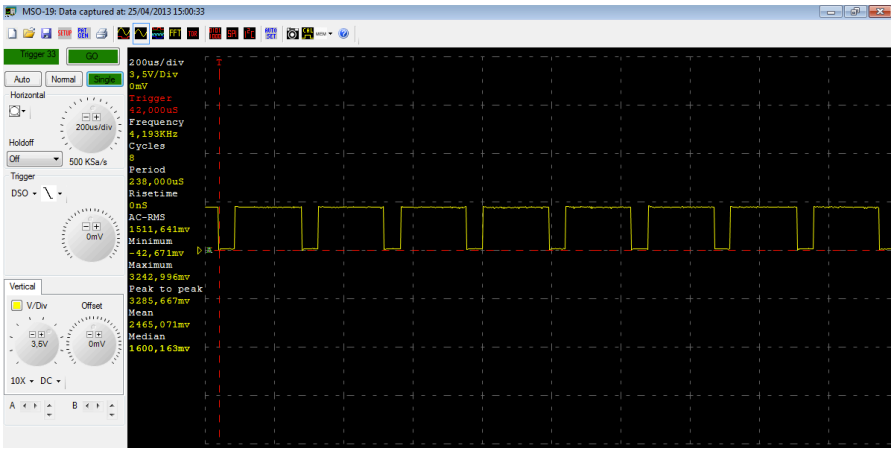


Imagen 61. PWM 80%

5.10. COSTOS Y MANUFACTURA

El costo total de fabricación es un criterio muy importante ya que de esto depende en gran medida su viabilidad y escalabilidad.

Los costos mas significativo son la fabricación del impreso y el montaje de los comoponentes debido a que estos procesos debieron cumplir con unos criterios especiales ya que el sistema embebido fue diseñado con componentes en empaques de montaje superficial que deben ser montados con una muy alta precisión, muy difícil de lograr a través de medios tradicionales o artesanales.

La empresa seleccionada para realizar la fabricación del impreso y montaje de componentes fue Microensamble, ubicada en el barrio Prado Veraniego de Bogotá, que cuenta con servicio de producción de impresos con la resolución necesaria por esta plataforma de desarrollo (8 mil), maquina de pick and place, horno de *“reflow”* y sistema de verificación de montaje a través de rayos X. El costo de este

servicio depende de la cantidad de componentes y tarjetas que se vayan a ensamblar. En el caso particular de este proyecto se montaron dos tarjetas con 35 componentes cada una.

La mayoría de los componentes fueron importados directamente desde Estados Unidos. Sin embargo, los microprocesadores, el programador y las memorias se compraron directamente con la empresa Microchip y fueron importados desde Indonesia. El modulo Bluetooth y los conectores fueron comprados en Sparkfun (www.sparfun.com), el oscilador los transistores, diodos y regulador fueron comprados en Digikey (www.digikey.com), el MPU9150 fue comprado directamente en Invensense (www.invensense.com), el Alien Jump Jet fue comprado a través de Ebay (www.ebay.com), las baterías en Hobbyking (www.hobbyking.com) y finalmente las resistencias y condensadores fueron compradas en Colombia en Tekcien. La tabla 10 resume los componentes y costos del desarrollo de esta plataforma:

COMPONENTE	VALOR ($USD)
Alien Jump Jet	60
EEPROM	2.6
Programador	30
Dspic33	3.73
Rn-42	16
Conectores	1.5
Baterías	5.5
mpu9150	17
Resistencias y condensadores	5
Manufactura y Montaje	120
Envíos	20
Transistores	2
Oscilador	1.3
Diodos	5
Regulador	1.6
TOTAL 1 EQUIPO	275

Tabla 10. Costos

El costo del programador se dividió entre 2, ya que solo es necesario uno para los dos vehículos., por lo tanto costo por plataforma de aproximadamente USD $ 260.

5.11. COMPARACIÓN DEL EQUIPO ORIGINAL CON EL MODIFICADO

Como parte de la evaluación de los cambios hechos con respecto a la parte mecánica, se compararon las características del equipo original con las medidas después de hacer las modificaciones y los resultados se muestran en la tabla 11.

SÍMBOLO	PARÁMETRO	MEDIDA MODIFICACIÓN	MEDICIÓN ORIGINAL	UNIDADES
A	Longitud brazo	0,20	0,20	m
Ptot	Peso total	0.068	0.068039	Kg
Phel	Peso hélice	0,00746	0,00746	Kg
B	Diámetro hélice	0.0683	0.0683	m
Pbat	Peso batería	0.020108	0.016110	Kg
D	Dimensiones Baterías	0.04*0.02*0.015	0.0323*0.0109*0.0105	m
Af	Área efectiva	0.044*0.044	0.044*0.044	m
Pcn	Peso circuito nuevo	0.007	0.006092	Kg
Atot	Altura rotor-base	0.0288	0.0288	m
Emp	Empuje	1309,089	1205,3255	N

Tabla 11. Comparación modificado-original

En la tabla 11 se encuentra que el peso total mejora a pesar que el peso de las baterías y del circuito aumenta, esto se debe a que se quitaron varias partes innecesarias de la estructura original, como leds y piezas plásticas.

El grosor de la lamina del impreso modificado es mayor (0,8 mm) frente a la del original que es de 0,6 mm y la mayoría de los componentes seleccionados tienen un peso mayor., lo cual hace que el peso total del circuito aumente.

A pesar de que las baterías pesan mas y su tamaño es mayor, se logro un aumento notable en el empuje, debido a que se están alimentando con mayor voltaje los motores, la corriente es mayor y la capacidad de descarga de las baterías aumentó. Los demás datos de la tabla conservaron su valor original debido a que son medidas inherentes a la estructura mecánica.

5.12. COMPARACIÓN CON UN VEHÍCULO SIMILAR

La tabla 12 presenta una comparación con la aeronave comercial AR.DRONE 2.0 [37], que es un vehículo similar que se puede encontrar en el mercado. Se hacen algunas comparaciones de las características mas relevantes con el vehículo desarrollado en el proyecto (CERJ-1).

CARACTERÍSTICA	AR.DRONE 2.0	CERJ-1
Procesador	ARM A8 1 GHz, 32 Bit	dsP 32 MHz, 16 bit
Sensores	Gyro (2000 $°/seg$), Accel ($\pm$2 g), Mag (6°), sensor de distancia y sensor de presión	Gyro (2000 $°/seg$), Accel ($\pm$2 g), Mag (6)
Video	720p,30 fps	N/A, se puede adicionar
Peso	420 g	68 g
Alcance	WI-FI (80 m)	Bluetooth 2.0 (10 m)
Dimensiones	517 mm X 517 mm	140mm X140mm
Precio	$789.000 promedio	$505.175 @ 1 USD=1837 pesos

Tabla 12. Comparación vehículo similar

El AR.DRONE 2.0, de la empresa Parrot ®, posee un tamaño y peso significativamente mayor que el CERJ-1. Posee un procesador de mayor capacidad y mayor velocidad y a bordo consta de los mismos sensores de navegación. Sin embargo, el AR.DRONE 2.0 posee en su interior un sensor de distancia, un sensor de presión y una cámara con una resolución de 720p.

En cuanto a las comunicaciones el AR.DRONE 2.0 se enlaza a través de WiFi, mientras que el CERJ-1 se comunica a través de Bluetooth, con una diferencia del alcance inherente a la tecnología (aprox. 80 m WiFi, aprox. 10 m Bluetooth). Una diferencia radical es la capacidad de programar algoritmos de control *"on-board"* en el CERJ-1, la cual es inexistente en el AR.DRONE 2.0. El equipo básico de Parrot sólo puede ser controlado a distancia, lo que también es posible en el CERJ-1

El precio del AR.DRONE 2.0 comprado en Colombia es 35,97 % mas costoso que la plataforma desarrollada. El precio del CERJ-1 incluye la importación de cada uno de los componentes y además de la producción y montaje en Colombia, donde es mas costosa que en lugares como Estados Unidos o China.

6. CONCLUSIONES:

Después de una búsqueda exhaustiva y sin mucha información con respecto a los estándares mundiales con relación a cuadrotores como plataformas de desarrollo, se llego a la conclusión que era conveniente ajustarse a los requerimientos mínimos particulares para aplicaciones alrededor de este tema, al igual que la población principal a la cual debía ir orientada la plataforma, principalmente estudiantes de ingeniería electrónica, sistemas, mecatrónica o afines.

Para que esta plataforma pueda ser usada por la población objetivo fue de gran importancia generar un manual con la información básica que se debe tener a mano, donde se encuentre la correcta manipulación del sistema, las precauciones y las generalidades de configuración.

Esta plataforma puede ser usada como herramienta para desarrollar y programar algoritmos de control de estabilidad. También se puede añadir un bloque de navegación, permitiendo la localización del vehículo con la ayuda de GPS, imágenes o por medio de sensores infrarrojos, etc., a través del bus I2C, al

que se pueden conectar hasta 6 dispositivos. Si se construye una flotilla de cuadrotores podría ser usada para investigación en trabajo cooperativo.

Las aplicaciones futuras son extensas y dependerán en gran medida de las necesidades que se generen en campos de investigación afines y de la promoción y fomento que se le de a esta herramienta.

Para concluir este trabajo se analizó además el estado actual del proyecto para así saber que tan cerca se encuentra de las metas planteadas en un principio. Los estándares de evaluación planteados se revisan y se encuentra que todos los criterios y requerimientos planteados en la etapa del anteproyecto fueron cubiertos y cumplidos satisfactoriamente:

¿Cómo se evaluó?

Los estándares de evaluación planteados al inicio del proyecto fueron los siguientes:

El vehículo debía estar en capacidad de volar:
Se realizaron pruebas de vuelo a puerta cerrada comprobando así que el equipo original tenia las características aerodinámicas necesarias para desempeñarse como un vehículo aéreo no tripulado, documentándolo en video. De igual forma se midieron las características mecánicas guardando cada variable en una tabla para compararlas con el vehículo modificado.

Debía ser reprogramable:
Se realizaron las modificaciones pertinentes, seleccionando un micro-controlador con la posibilidad de cambiar el contenido en memoria. Una capacidad de 128 Kbytes se consideró suficiente, ya que se cumple con la meta planteada de dejar por lo menos el 50% de memoria libre luego de implementar el código de funcionamiento mínimo.

Por otra parte se dejaron varios pines disponibles para conectar mas dispositivos. Cabe anotar que el pin 13 del micro-controlador puede ser empleado como interrupción externa y que los pines de programación pueden ser usados como pines de propósito general. Lo anterior significa que la plataforma no solo es modificable en cuanto a software, sino también en cuanto a hardware a partir de la posibilidad de adicionar nuevos componentes.

Ser inter-comunicable:
Se conectaron con un computador los vehículos y se visualizaron las variables que provienen del sensor, por medio de un monitor de puerto serial a una taza de 115200 baudios. También se demostró la inter-comunicabilidad con la realización de un código que al enviar una determinada letra varia el ciclo útil, lo cual comprueba que el vehículo se puede comunicar y controlar a distancia.

Escalable:
Al comparar el precio de este equipo con otros ya existentes se observa que este dispositivo esta dentro del rango de costos normales para este tipo de sistemas lo cual lo hace viable económicamente. Incluso se puede observar una notable diferencia con el ARDRONE 2.0 que cuesta un 36% mas.

Se demostró que la plataforma se puede comunicar otros equipos o con un computador. Adicionalmente es posible conectar varios dispositivos bajo una misma red por medio de un Bluetooth Access Point + Router clase 1. Los componentes utilizados en el desarrollo del sistema salieron recientemente al mercado y son de alta rotación por lo que es posible adquirir repuestos, que es otro criterio que define la del sistema.

¿Cómo se puede extender y evolucionar con el tiempo?

Como primera medida en un futuro se debería desarrollar y programar algoritmos de control de estabilidad. Segundo se puede añadir un bloque de navegación, localizando el vehículo con la ayuda de GPS, imágenes o por medio de sensores infrarrojos, etc. Ya que el usuario cuenta con pines disponibles,

puede conectar hasta 6 dispositivos mas por I2C, bien sea sensores, un GPS o hasta una cámara, para poder adquirir imágenes durante el vuelo. Podrían desarrollarse estudios en trabajo cooperativo entre sistemas no tripulados.

A lo largo del desarrollo de todo el proyecto se presentaron ciertos retos y situaciones que no hacen explícitamente parte de éste, pero que se consideran importantes como parte del aprendizaje, es por esto que se describe el proceso y algunas experiencias que se presentaron a la hora de la realización del trabajo:

Durante la etapa de planeación y requerimientos, se analizaron los puntos críticos de diseño como la población a la cual va orientada este tipo de plataforma. Una vez se tuvieron estos criterios claros se procedió a revisar documentos e información relacionada con cuadricópteros y plataformas de desarrollo, se buscaron componentes que cumplieran con el estándar y finalmente se procedió a definir los sistemas y sub sistemas que integrarían la plataforma.

Diseño del impreso: Para esta etapa fue necesario tener en cuenta las recomendaciones de conexiones mínimas para el correcto funcionamiento de cada componente y así poder generar un esquemático en un software de diseño de circuitos(EAGLE®) sobre el cual fue necesario profundizar los conocimientos con respecto a este programa, en temas tales como la creación de footprints específicos para los componentes usados los cuales en algunos casos eran demasiado nuevos y no tenían librerías. En general fue necesario profundizar en todo lo relacionado con la creación de circuitos con componentes de montaje superficial.

Compra de Componentes: En este punto fue necesario una herramienta de compra de componentes seria y confiable, para esto se seleccionaron tiendas de componentes electrónicos en línea tales como MICROCHIP®, SPARKFUN, DIGIKEY® y HOBBYKING®, ya que en Colombia no existe tanta variedad ni disponibilidad como en Estados Unidos. Fue necesario tener un método de importación donde se considerara costos y tiempos los cuales varían dependiendo de la forma como estos sean importados, para esto se busco la opción mas rápida y económica a través del método de casillero en Estados Unidos.

Producción y Ensamblaje: Se debió empezar con la búsqueda de alguna empresa que estuviera en la capacidad de producir y ensamblar circuitos con la resolución y precisión necesaria, preferiblemente en Bogotá donde la posibilidad de comunicación y transporte fuera lo mas sencillo posible. Tras una búsqueda exhaustiva se encontró la empresa Microensambe S.A.S.

El proceso de producción con esta empresa no fue tan sencillo, fácil ni confiable como se esperaba, esto se debe a que para comenzar solicitaron documentos inesperados tales como el RUT, también hubo una desinformación y contradicción con respecto a los costos. Fue necesario tener en cuenta el tiempo de producción que fue alrededor de dos semanas, al igual se debió aprender a generar y trabajar con archivos tipo GERBER los cuales aparentemente son el estándar de la industria para la producción de circuitos.

Validación: fue necesario aprender MPLAB y profundizar conceptos en lenguaje C, para así generar y programar un código que permitiera validar el correcto funcionamiento de los bloques que integran el sistema, observando en el osciloscopio variables de entrada/salida.

Diseño del Manual: Teniendo como referencia manuales de productos de empresas reconocidas se elaboró el manual de la plataforma CERJ-1, pensando siempre en el grupo de personas al cual esta orientado, y la información que se debe tener a mano como es la correcta manipulación, las precauciones y configuraciones necesarias.

7. BIBLIOGRAFÍA:

[1] 2012 Junio, “Con Cuadricóptero salvavidas, colombianos llegan a la final del Imagine Cup”,[En línea],Disponible en: http://www.enter.co/moviles/con-cuadricoptero-salvavidas-colombianos-llegan-a-la-final-del-imagine-cup/.

[2] Julio de 2012, “La policía japonesa interesada en el uso de Cuadricóptero para misiones de rescate”, [Online], disponible en: http://alt1040.com/2012/07/policia-japonesa-cuadricopteros.

[3] S. Gomez y P. Rafael “Sistema de Control de estabilización de altura para un vehículo Cuadrotor” B.S. tesis, Universidad Javeriana, Bogota,2012.

[4] M. Cheguini “Sistema de navegación inercial”, B.S. tesis, Universidad Javeriana, Bogotá, 2011.

[5] P. Pounds, R Mahony y P. Corke, “Modeling and Control SimulationforAutonomousQuadrotor”, AustralianNationalUniversity, Canberra, Australia.

[6] “DynamicAnalysis and PID Control for a Quadrotor” Coll. of Mech. &Electr. Eng., China Jiliang Univ., Hangzhou, China.

[7] Niazi, Y.A.K. (2010). “Modeling and neural control of quadrotorhelicopter”. UnitedStates: LAP LAMBERT AcademicpublishingGmbH&Co.

[8] Alien Jump Jet instruction manual, [En línea]. Disponible: http://www.snelflight.co.uk/downloads/JJmanual.pdf, Londres, Inglaterra

[9] Valenzuela. R, “Sistema de navegación autonomo. Navegación inercial.Errores”, Universidad de Sevilla, España, 10 de Marzo de 2010

[10] Ferrer. G , “Integración Kalman de sensores inerciales INS con GPS en un UAV”, Abril de 2009

[11] Torres. J, Angarita. P & Siegwart. R, “Definición De Criterios De Diseño De Un Magnetómetro Multifuncional Que Integra Diferentes Técnicas, Para Caracterización Magnética De Micro Y Nano Estructuras”, Revista Colombana de Física. Vol 43, Manizales, Colombia, 2011.

[12] SAINT-RAYMOND. L (2011), “Hacking and controlling toy flyers” . EPFL, Biorobotics Laboratory, Laussane, Suiza.

[13] J. V. Gárate , A. R. Muñoz y M. Melénde, “Plataformas Tecnológicas de Tele-educación”, Santo Domingo, Febrero 2006.

[14] QuanserEngineering®. “QuadrotorforIndoorUnmannedAerialVehicleResearch”, Markham, Ontario, Canada.

[15] IMX53QSB: i.MX53 Quick Start Board, Freescale, [En línea]. Disponible: http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=IMX53QSB.

[16] Jandourek. E, “A model for platform development”. Hewelet-Packard Journal, United States. 1996

[17] Bouabdallah. S, North. A & Siegwart. R, “PID vs LQ Control techniques applied to an indoor Micro Quadrotor”, Instituto tecnologico Sueco., Lussane, Suiza , 2004

[18] J. M. B. Dominguez “Quadrotorprototype”, Instituto técnico superior de Lisboa, Portugal., Octubre 2009

[19] J. A. Redolfi y A. Henze, “Cuadricóptero Autónomo de Arquitectura Abierta, QA3”,13 Marzo de 2011

[20] Micro-controlador, dsPIC33FJ128MC802, Hoja de especificaciones, MICROCHIP®, [En línea].Disponible: http://ww1.microchip.com/downloads/en/DeviceDoc/70291G.pdf

[21] Bluetooth RN-42, Hoja de especificaciones, ROVING NETWORKS, [En línea]. Disponible: https://www.sparkfun.com/datasheets/Wireless/Bluetooth/rn-42-ds.pdf

[22] Sensor de movimiento, MPU9150, Hoja de especificaciones, INVENSENSE®, [En línea].Disponible: http://www.invensense.com/mems/gyro/documents/PS-MPU-9150A.pdf

[23] Transistores, ZXTN25015DFH, ZETEX SEMICONDUCTORS®, Hoja de especificaciones, [En línea]. Disponible: http://www.diodes.com/datasheets/ZXTN25015DFH.pdf

[24] Rectificador Schottky, LSM115J, Hoja de especificaciones, MICROSEMI®, Hoja de especificaciones, [En línea]. Disponible: http://media.digikey.com/pdf/Data%20Sheets/Microsemi%20PDFs/LSM115J.pdf

[25] REGULADOR LM3940, TEXAS INSTRUMENTS®, [En línea]. Disponible: http://www.ti.com/lit/ds/symlink/lm3940.pdf.

[26] EEPROM 24FC1025, Hoja de especificaciones, MICROCHIP®, [En línea]. Disponible: http://ww1.microchip.com/downloads/en/DeviceDoc/21941B.pdf.

[28] Alien Jump Jet, Snelflight® , [En línea]. Disponible: http://www.snelflight.co.uk/jumpjet.htm.

[29] Banco de la Republica, peso estándar de monedas, Billetes y monedas, Bogotá, Colombia, [En línea]. Disponible: http://www.banrep.gov.co/billetes_monedas/bm_mon_100.htm.

[30] Banco de la Republica, peso estándar de monedas, Billetes y monedas, Bogotá, Colombia, [En línea]. Disponible: http://www.banrep.gov.co/billetes_monedas/bm_mon_200.htm.

[31]Banco de la Republica, peso estándar de monedas, Billetes y monedas, Bogotá, Colombia, [En línea]. Disponible: http://www.banrep.gov.co/billetes_monedas/bm_mon_500.html.

[32] Ministerio de Cultura, Recreación y Deporte, Gravedad promedio en Bogotá, Bogotá, Colombia, [En línea]. Disponible: http://www.culturarecreacionydeporte.gov.co/portal/node/158

[33]IPC-2221, “ Generic Standard onPrinted Board Design”, Institute for Interconnecting and Packaging Electronic Circuits, [En línea]. Disponible: http://classes.soe.ucsc.edu/cmpe174/Fall12/References/AppNotes/IPC-2221.pdf

[34] Microstick II, MICROCHIP® [En línea]. Disponible http://ww1.microchip.com/downloads/en/DeviceDoc/51951B.pdf

[35] CARGADOR-BALANCEADOR, IMAX B8, IMAX, [En línea]. Disponible: http://helicorc.org/telechargement/ImaxB8manual.pdf.

[36] PWM, meteorología y electrónica [En línea]. Disponible: http://meteoyelectronica.blogspot.com/2010/12/pwm-una-manera-sencilla-de-controlar-un.html

[37] AR.DRONE 2.0, Parrot®, Especificaciones, [En línea]. Disponible: http://ardrone2.parrot.com/

[38] NXP Semiconductors®, "I2C Bus specification and user manual", [En línea]. Disponible: http://www.nxp.com/documents/user_manual/UM10204.pdf

[39] FREESCALE SEMICONDUCTOR, "Using the Universal Asynchronous Receiver Transmitter (UART) eTPU Function", [En línea]. Disponible: http://www.freescale.com/files/32bit/doc/app_note/AN2853.pdf

[40] Robótica, Manipuladores y Robots Móviles. Anibal Ollero Baturone, "¿Qué es un sistema de navegación inercial?". Abril 2011.[En línea]. Disponible: http://www.xatakaciencia.com/sabias-que/que-es-un-sistema-de-navegacion-inercial

[41]D.Mellinger, M. Shomin, M. Nathan y V. Kumar, "Control of Quadrotor for robust perching and landing", Universidad de Pennsylvania, Philadelphia, PA 19104, USA.

[42] García, J (2003) Terminología aeronáutica. Madrid: Ediciones Díaz Santos.

[43] F.G. Irving, "AnIntroductiontothe Longitudinal StaticStability of Low-SpeedAircraft", PergamonPress, Oxford, UK, 1966.

[44] R-Luis, "Sistemas micro-controlados". [En línea]. Disponible:http://r-luis.xbot.es/pic1/pic01.html

[45] Pounds, P. &Mahony, R. &Corke, P. (Agosto 2010). "Modelling and Control of a Quad-Rotor Robot". Brisbane: AustralianNationalUniversity.

[46] Luukkonen, T. (Agosto 2011). "Modelling and control of quadcopter". Finlandia: Aalto University.

[47] Brito, J.M.(Octubre 2009). QuadrotorPrototype. Lisboa: Universidad técnica de Lisboa.

[48] Putro, I.E. (2011). "Modeling and Control SimulationforautonomousQuadrotor". UnitedStates: LAP LAMBERT AcademicpublishingGmbH&Co.

[49] Niazi, Y.A.K. (2010). "Modeling and neural control of quadrotorhelicopter".UnitedStates: LAP LAMBERT AcademicpublishingGmbH&Co.

[50] P. Giraldo y M. Vladimir Noviembre 2009 "Modelamiento, simulación y hallazgo de modelos linealizados a partir de técnicas de identificación de un Cuatrirrotor". Bogotá: Universidad Nacional de Colombia.

[51] Código en línea, ARDUCOPTER, "*opensource*" [En línea]. Disponible: https://code.google.com/p/arducopter/

[52] D. Verdejo , J. Pérez & I. Estrada, "Control de un vehículo aéreo no tripulado", Universidad Complutense de Madrid, Facultad de informática, 2009, [En línea]. Disponible: http://eprints.ucm.es/9477/1/documentacion.pdf

[53] J. Manzanares, "Interfaz LabView para programar el sistema de control de quadrotores", Universidad Politécnica de Cataluña, 2010, [En línea]. Disponible: http://upcommons.upc.edu/pfc/bitstream/2099.1/10490/1/memoria.pdf

[54] "Cuadricopteros voladores aprenden a botar pelotas al aire", Todo Interesante (Temas Curiosos y Sorprendentes, Ingenios, Actualidad, Utilidades), [En línea]. Disponible: http://www.todointeresante.com/2010/12/cuadricopteros-voladores-aprenden-botar.html

[55] The CircuitCalculator, PCB Trace Width Calculator, [En línea]. Disponible: http://circuitcalculator.com/wordpress/?p=25/

ANEXOS:

Código 1 configuración de i2c y UART:

Código modificado del original de Ing. Mazeyar Cheguini[4]

```
SOURCE FILES
DELAY.C

#include "DELAY.h"
void Delayms( unsigned count){
T1CONbits.TON = 1;
      while (count--){
            TMR1 = 0;
                  while (TMR1<10000);
    };
};

void Delayus( unsigned count){
    T1CONbits.TON = 1;
            while (count--){
                  TMR1 = 0;
                  while (TMR1<10);
            };
};

DELAY.H

#if !defined(__DELAY_H)
#define     __DELAY_H
#include <p33FJ128MC802.h>
//Function Prototype
void Delayms(unsigned count);
void Delayus(unsigned count);
#endif

I2C.C

#include "I2C.h"
#define i2cWRITE 0
#define i2cREAD  1

void i2cStart(void){
I2C1CONbits.SEN = 1;
      while(I2C1CONbits.SEN);
};

void i2cRestart(void){
I2C1CONbits.RSEN = 1;
      while(I2C1CONbits.RSEN);
};

void i2cStop(void){
I2C1CONbits.PEN = 1;
      while(I2C1CONbits.PEN);
};

void i2cNack(void){
I2C1CONbits.ACKDT=1;
I2C1CONbits.ACKEN=1;
      while(I2C1CONbits.ACKEN);
```

```
I2C1CONbits.ACKDT=0;
};

void i2cAck(void){
I2C1CONbits.ACKDT=0;
I2C1CONbits.ACKEN=1;
      while(I2C1CONbits.ACKEN);
};

unsigned char i2cGetAck(void){
return (!I2C1STATbits.ACKSTAT);
};

void i2cWrite(unsigned char byte){
I2C1TRN=byte;
      while(I2C1STATbits.TBF);
};

unsigned char i2cWriteAck(unsigned char byte){
I2C1TRN=byte;
      while(I2C1STATbits.TRSTAT==1);
return (!I2C1STATbits.ACKSTAT);
};


unsigned char i2cRead(void){
I2C1CONbits.RCEN=1;
      while(!I2C1STATbits.RBF);
return(I2C1RCV);
};

void initI2C(void){
I2C1CONbits.A10M=0;
I2C1CONbits.SCLREL=0;
I2C1ADD=0;
I2C1MSK=0;
I2C1CONbits.SMEN = 0;
I2C1BRG = MYI2CBRG;
I2C1CONbits.I2CEN = 1;
};

unsigned char i2cWriteByte(unsigned char i2cADD,unsigned char ADD,unsigned char data){
unsigned char error = 0;
i2cStart();
error += !i2cWriteAck((i2cADD<<1)|i2cWRITE);
error += !i2cWriteAck(ADD);
error += !i2cWriteAck(data);
i2cStop();
return error;
};

unsigned char i2cReadByte(unsigned char i2cADD,unsigned char ADD,unsigned char *data){
unsigned char error = 0;
i2cStart();
error += !i2cWriteAck((i2cADD<<1)|i2cWRITE);
error += !i2cWriteAck(ADD);
i2cStart();
error += !i2cWriteAck((i2cADD<<1)|i2cREAD);
      if(error){
            i2cStop();
            return error;
      };
*data = i2cRead();
i2cNack();
i2cStop();
return error;
};

unsigned char i2cReadArray(unsigned char i2cADD,unsigned char ADD,unsigned char *data,unsigned
char length){
unsigned char error = 0;
i2cStart();
error += !i2cWriteAck((i2cADD<<1)|i2cWRITE);
error += !i2cWriteAck(ADD);
i2cStart();
error += !i2cWriteAck((i2cADD<<1)|i2cREAD);
```

```
      if(error){
            i2cStop();
            return error;
      };
            while(length-->1){
                  *data++ = i2cRead();
                  i2cAck();
            };
*data = i2cRead();
i2cNack();
i2cStop();
return error;
};
```

I2C.H

```
#if !defined(__I2C_H)
#define __I2C_H

#include <p33FJ128MC802.h>
#define MYI2CBRG 7
#define ACK       0
#define NACK    1

// --------------- Function Prototypes ---------------------
void              i2cStart(void);                              // Send Start bit
void              i2cRestart(void);                        // Send Restart bit
void              i2cStop(void);                               // Send Stop bit
void              i2cAck(void);                                // Send Acknolegment
void              i2cNack(void);                               // Send Not Acknolegment
unsigned char     i2cGetAck(void);                         // Receive Acknolegment
void              i2cWrite(unsigned char c);               // Write a byte
unsigned char     i2cWriteAck(unsigned char c);      // Write a byte, and receive Acknolegment
unsigned char     i2cRead(void);                               // Read a byte
void              initI2C(void);                               // Initialize I2C module
unsigned char i2cWriteByte(unsigned char i2cADD,unsigned char ADD,unsigned char data);
unsigned char i2cReadByte(unsigned char i2cADD,unsigned char ADD,unsigned char *data);
unsigned char i2cReadArray(unsigned char i2cADD,unsigned char ADD,unsigned char *data,unsigned
char length);
#endif
```

MPU9150.C

```
#include "MPU9150.h"
//I2C library
#include "../I2C/I2C.h"
//Delay Library
#include "../DELAY/DELAY.h"
unsigned char MPU9150_Read[1];
float MPU9150_X_Scale=1370, MPU9150_Y_Scale=1370, MPU9150_Z_Scale=1370;
void MPU9150_Init(void){
//    ------------------- Enable write to EEPROM -------------------------------
    Delayms(50);

i2cReadByte(MPU9150_I2C, MPU9150_PWR_MGM_1, MPU9150_Read);
MPU9150_Read[0] &= (0<<6);
MPU9150_Read[0] |= (1<<0);
MPU9150_Read[0] |= (1<<1);

i2cWriteByte(MPU9150_I2C, MPU9150_PWR_MGM_1, MPU9150_Read[0]);
Delayms(1);

// CONFIGURACION Sample Rate Divider
i2cWriteByte(MPU9150_I2C, MPU9150_SMPRT_DIV , 0x09);
Delayms(1);

//CONFIGURACION power management 2
i2cReadByte(MPU9150_I2C, MPU9150_PWR_MGM_2, MPU9150_Read);
MPU9150_Read[0] = 0xC0;
i2cWriteByte(MPU9150_I2C, MPU9150_PWR_MGM_2, MPU9150_Read[0]);
Delayms(1);


//CONFIGURACION CONFIG
i2cWriteByte(MPU9150_I2C, MPU9150_CONFIG, 0x01);
Delayms(1);
```

```
//CONFIGURACION  GYRO
i2cReadByte(MPU9150_I2C, MPU9150_GYRO_CONFIG, MPU9150_Read);
MPU9150_Read[0] = 0x00;
i2cWriteByte(MPU9150_I2C, MPU9150_GYRO_CONFIG, MPU9150_Read[0]);
Delayms(1);

//CONFIGURACION ACCEL
i2cReadByte(MPU9150_I2C, MPU9150_ACCEL_CONFIG, MPU9150_Read);
MPU9150_Read[0] = 0x00;
i2cWriteByte(MPU9150_I2C, MPU9150_ACCEL_CONFIG, MPU9150_Read[0]);
Delayms(1);

//CONFIGURACION MAGN.
i2cReadByte( MPU9150_MAG_ADD, MPU9150_MAG_CNTL, MPU9150_Read);
MPU9150_Read[0] |= (1<<0);
MPU9150_Read[0] |= (1<<1);
MPU9150_Read[0] |= (1<<2);
MPU9150_Read[0] |= (1<<3);
i2cWriteByte( MPU9150_MAG_ADD, MPU9150_MAG_CNTL, MPU9150_Read[0]);
Delayms(1);

i2cReadByte( MPU9150_MAG_ADD, MPU9150_MAG_CNTL, MPU9150_Read);
MPU9150_Read[0] &= (0<<6);
i2cWriteByte( MPU9150_MAG_ADD, MPU9150_MAG_ASTC, MPU9150_Read[0]);
Delayms(1);

};

// ------------------------MPU9150 READ RAW DATA-------------------------------


unsigned char MPU9150_Read_RAW(int *data){
unsigned char error= 0, buffer[14];
error = i2cReadArray(MPU9150_I2C, MPU9150_ACCEL_XOUT_H, buffer,14);
// acc X
*data++ = ((buffer[0]<<8)|buffer[1]);
// acc Y
*data++ = ((buffer[2]<<8)|buffer[3]);
// acc Z
*data++ = ((buffer[4]<<8)|buffer[5]);
// Temp
*data++ = ((buffer[6]<<8)|buffer[7]);
// Gyr X
*data++ = ((buffer[8]<<8)|buffer[9]);
// Gyr Y
*data++ = ((buffer[10]<<8)|buffer[11]);
// Gyr Z
*data++ = ((buffer[12]<<8)|buffer[13]);
return error;
};

unsigned char MPU9150_Read_RAW_MAG(volatile int *data){
unsigned char error= 0, buffer[6];
error = i2cReadArray(MPU9150_MAG_ADD,MPU9150_MAG_XOUT_H, buffer,6);
*data++ = (buffer[0]<<8)|buffer[1];
*data++ = (buffer[4]<<8)|buffer[5];
*data++ = (buffer[2]<<8)|buffer[3];
return error;
};
```

MPU9150.H

```
#if !defined(__MPU9150_H)
#define __MPU9150_H



// A Simple Module for control MPU9150 by
// ------------- MPU9150 REGISTERS ------------------------
// MPU9150 startup time 50m[s]
#define MPU9150_startup            50
// AD0 pin to GND
#define MPU9150_I2C                0x68
#define MPU9150_MAG_ADD                 0X48
// AD0 pin to VDD
```

```
//#define MPU9150_I2C              0x69


#define MPU9150_TEMP_OUT_H         0x41
#define MPU9150_TEMP_OUT_L         0x42
#define MPU9150_PWR_MGM_1               0x6B
#define MPU9150_PWR_MGM_2               0x6C
#define MPU9150_SMPRT_DIV               0x19
#define MPU9150_CONFIG                  0x1A
#define MPU9150_GYRO_CONFIG             0x1B
#define MPU9150_ACCEL_CONFIG            0x1C
#define MPU9150_ACCEL_XOUT_H            0x3B
#define MPU9150_ACCEL_XOUT_L            0x3C
#define MPU9150_ACCEL_YOUT_H            0x3D
#define MPU9150_ACCEL_YOUT_L            0x3E
#define MPU9150_ACCEL_ZOUT_H            0x3F
#define MPU9150_ACCEL_ZOUT_L            0x40
#define MPU9150_GYRO_XOUT_H             0x43
#define MPU9150_GYRO_XOUT_L             0x44
#define MPU9150_GYRO_YOUT_H             0x45
#define MPU9150_GYRO_YOUT_L             0x46
#define MPU9150_GYRO_ZOUT_H             0x47
#define MPU9150_GYRO_ZOUT_L             0x48
#define MPU9150_USER_CTRL               0x6A
#define MPU9150_WIA                     0x00
#define MPU9150_MAG_XOUT_H              0x03
#define MPU9150_MAG_XOUT_L              0x04
#define MPU9150_MAG_YOUT_H              0x05
#define MPU9150_MAG_YOUT_L              0x06
#define MPU9150_MAG_ZOUT_H              0x07
#define MPU9150_MAG_ZOUT_L              0x08
#define MPU9150_MAG_CNTL                0x0A
#define MPU9150_MAG_ASTC                0x0C



// Function Prototype
void MPU9150_Init(void);
unsigned char MPU9150_Read_RAW(int *data);
#endif
```

UART.C

```
#include "UART2.h"

void UART_Init(void){
IEC1bits.U2RXIE = 0;
U2MODEbits.UARTEN = 0;
U2BRG = 7;
U2MODE &= 0b00000000;
U2MODEbits.BRGH = 1;
U2STAbits.URXISEL = 1;
U2MODEbits.UARTEN = 1;
U2STAbits.UTXEN = 1;
RPINR19bits.U2RXR = 0b0010;
RPOR1bits.RP3R = 0b00101;
IEC1bits.U2RXIE = 0;
};

void UART_Put_Char(int c){
      while(U2STAbits.UTXBF);
U2TXREG = c;
IFS1bits.U2TXIF = 0;
};

char UART_Get_Char(void){
      while(!U2STAbits.URXDA);
IFS1bits.U2RXIF = 0;
return U2RXREG;
};

void UART_Put_String(char *s){
      while(*s)
            UART_Put_Char(*s++);
lineU2();
};
```

```
void UART_Int_Tx(int *Value, char counts){
int i=0;
UART_Put_Char(0x00);
UART_Put_Char(0x00);
      for(i=0; i<counts;++i){
            char Low_Part = (char) (*Value & 0xFF);
            char High_Part = (char) (*Value>>8);
            UART_Put_Char(High_Part);
            UART_Put_Char(Low_Part);
//          *Value++;
      };
};
```

UART.H

```
#if !defined(__UART_H)
#define __UART_H

#include <p33FJ128MC802.h>
#define FOSC()                                  (32000000ul)
#define FCY()                                   (FOSC()/2)
#define     BAUDRATE                            115200
#define MYBAUDRATE                          (((FCY())-(4*BAUDRATE))/(4*BAUDRATE))
#define SETBAUDRATE                             ((FCY())/(4*(MYBAUDRATE+1)))
#define BAUDRATE_ERROR              ((SETBAUDRATE > MYBAUDRATE) ? SETBAUDRATE-BAUDRATE :
BAUDRATE-SETBAUDRATE)
#define BAUDRATE_ERROR_PERCENT      ((BAUDRATE_ERROR*100+BAUDRATE/2)/BAUDRATE)

// -------- A simple Micro for UART errror indication -----------
      #if (BAUDRATE_ERROR_PERCENT > 3)
      #error UART2 frequency error is worse than 3% NOT APPLICABLE!!
                  #elif (BAUDRATE_ERROR_PERCENT > 2)
                  #warning UART2 frequency error is worse than 2% NOT RECOMMENDED
      #endif

#define lineU2()       UART_Put_Char('\r');UART_Put_Char('\n')

#endif
// ---------------- Function Prototype ---------------------
void UART_Init();
void UART_Put_Char(int c);
void UART_Put_String(char *s);
char UART_Get_Char(void);
void UART_Int_Tx(int *Value, char counts);
```

MAIN.C

```
#include <p33FJ128MC802.h>
#include "compiler.h"
#include "DELAY/DELAY.h"
#include "UART/UART2.h"
#include "I2C/I2C.h"
#include "MPU9150.h"
#define GYR_CON 0.000133758f
#define KP_DIVIDER 17179.86918f
#include <math.h>
#define Ki 0.9f
#define DeltaT 0.01f
#include "CUATERNIONES.h"


void GD_Filter(void);
void KP_Gain(void);
void MPU9150_Init(void);
unsigned char MPU9150_Read_RAW_MAG(volatile int *data);
unsigned char MPU9150_Read_RAW(int *data);
int MPU[7], Mag[3];
float MPU9150ACC_Scale=16384;
float MPU9150GYR_Scale=131;
float Kp = 1;
extern float MPU9150_X_Scale, MPU9150_Y_Scale, MPU9150_Z_Scale;
float q0=1, q1=0, q2=0, q3=0;
```

```
_FPOR(HPOL_ON & PWMPIN_ON & ALTI2C_ON);


// --------------- System ---------------------
int main(void){

CLKDIVbits.DOZE =0;
CLKDIVbits.FRCDIV =0;
CLKDIVbits.PLLPRE =0;
OSCCONbits.COSC=0b010;
OSCCONbits.LPOSCEN=0;

TRISBbits.TRISB2=1;
TRISBbits.TRISB3=0;
TRISBbits.TRISB7=1;
RPOR1bits.RP3R = 5;
RPINR19bits.U2RXR=2;
extern int __C30_UART; __C30_UART = 2;
AD1PCFGL = 0xffff;
Delayms(50);
initI2C();
MPU9150_Init();
MPU9150_Init();
UART_Init();
KP_Gain();
int Rotate_Axis[3];
      while(1)
      {
MPU9150_Read_RAW_MAG(Mag);
MPU9150_Read_RAW(MPU);
Delayms(5);
GD_Filter();
                  Rotate_Axis[0] = (int) 10000*(atan2(2*(q1*q2)-2*(q0*q3), 2*(q0*q0)+2*(q1*q1)-
1));
                  Rotate_Axis[1] = (int) -10000*(asin(2*(q1*q3)+2*(q0*q2)));
                  Rotate_Axis[2] = (int) 10000*(atan2(2*(q2*q3)-2*(q0*q1), 2*(q0*q0)+2*(q3*q3)-
1));


printf("YAW \t");
      //printf("%f", (double) Rotate_Axis[0]*0.05729578);
        printf("%f", (double) MPU[0]/16384);
printf("\t PITCH \t");
      //printf("%f", (double) Rotate_Axis[1]*0.05729578);
        printf("%f", (double) MPU[1]/16384);
printf("\t ROLL \t");
      //printf("%f", (double) Rotate_Axis[2]*0.05729578);
        printf("%f", (double) MPU[2]/16384);
printf("\n");
      };
};



void GD_Filter(void){
float gx = (float)((MPU[4]/MPU9150GYR_Scale)*GYR_CON);
float gy = (float)((MPU[5]/MPU9150GYR_Scale)*GYR_CON);
float gz = (float)((MPU[6]/MPU9150GYR_Scale)*GYR_CON);
float mx = (float)((Mag[1])/MPU9150_Y_Scale);
float my = (float)(Mag[0]/MPU9150_X_Scale);
float mz = (float)(Mag[2]/MPU9150_Z_Scale);

float ax = (float)(MPU[0]/MPU9150ACC_Scale);
float ay = (float)(MPU[1]/MPU9150ACC_Scale);
float az = (float)(MPU[2]/MPU9150ACC_Scale);
float norm;
float hx, hy, hz, bx, bz;
float vx, vy, vz, wx, wy, wz;
float ex, ey, ez;
//ADD q1,q2,q3;
float q0q0 = q0*q0;
float q0q1 = q0*q1;
float q0q2 = q0*q2;
float q0q3 = q0*q3;
```

```
float q1q1 = q1*q1;
float q1q2 = q1*q2;
float q1q3 = q1*q3;
float q2q2 = q2*q2;
float q2q3 = q2*q3;
float q3q3 = q3*q3;
float exInt = 0, eyInt = 0, ezInt = 0;
norm = sqrt(ax*ax + ay*ay + az*az);
ax = ax / norm;
ay = ay / norm;
az = az / norm;
norm = sqrt(mx*mx + my*my + mz*mz);
mx = mx / norm;
my = my / norm;
mz = mz / norm;
hx = 2*mx*(0.5 - q2q2 - q3q3) + 2*my*(q1q2 - q0q3) + 2*mz*(q1q3 + q0q2);
hy = 2*mx*(q1q2 + q0q3) + 2*my*(0.5 - q1q1 - q3q3) + 2*mz*(q2q3 - q0q1);
hz = 2*mx*(q1q3 - q0q2) + 2*my*(q2q3 + q0q1) + 2*mz*(0.5 - q1q1 - q2q2);
bx = sqrt((hx*hx) + (hy*hy));
bz = hz;
vx = 2*(q1q3 - q0q2);
vy = 2*(q0q1 + q2q3);
vz = q0q0 - q1q1 - q2q2 + q3q3;
wx = 2*bx*(0.5 - q2q2 - q3q3) + 2*bz*(q1q3 - q0q2);
wy = 2*bx*(q1q2 - q0q3) + 2*bz*(q0q1 + q2q3);
wz = 2*bx*(q0q2 + q1q3) + 2*bz*(0.5 - q1q1 - q2q2);
ex = (ay*vz - az*vy) + (my*wz - mz*wy);
ey = (az*vx - ax*vz) + (mz*wx - mx*wz);
ez = (ax*vy - ay*vx) + (mx*wy - my*wx);
exInt = exInt + ex*Ki;
eyInt = eyInt + ey*Ki;
ezInt = ezInt + ez*Ki;
gx = gx + Kp*ex + exInt;
gy = gy + Kp*ey + eyInt;
gz = gz + Kp*ez + ezInt;
q0 = q0 + (-q1*gx - q2*gy - q3*gz)*DeltaT;
q1 = q1 + (q0*gx + q2*gz - q3*gy)*DeltaT;
q2 = q2 + (q0*gy - q1*gz + q3*gx)*DeltaT;
q3 = q3 + (q0*gz + q1*gy - q2*gx)*DeltaT;
norm = sqrt(q0*q0 + q1*q1 + q2*q2 + q3*q3);
q0 = q0 / norm;
q1 = q1 / norm;
q2 = q2 / norm;
q3 = q3 / norm;
};

void KP_Gain(void){
int i=0;
float GrX=0, GrY=0, GrZ=0;
      for (i=0; i<100; i++){
            MPU9150_Read_RAW(MPU);
            GrX += (float)MPU[4];
            GrY += (float)MPU[5];
            GrZ += (float)MPU[6];
      };
GrX = GrX/100; GrY = GrY/100; GrZ = GrZ/100;
Kp = 0.8660254*(sqrt(GrX*GrX+GrY*GrY+GrZ*GrZ)/(KP_DIVIDER));
};
```

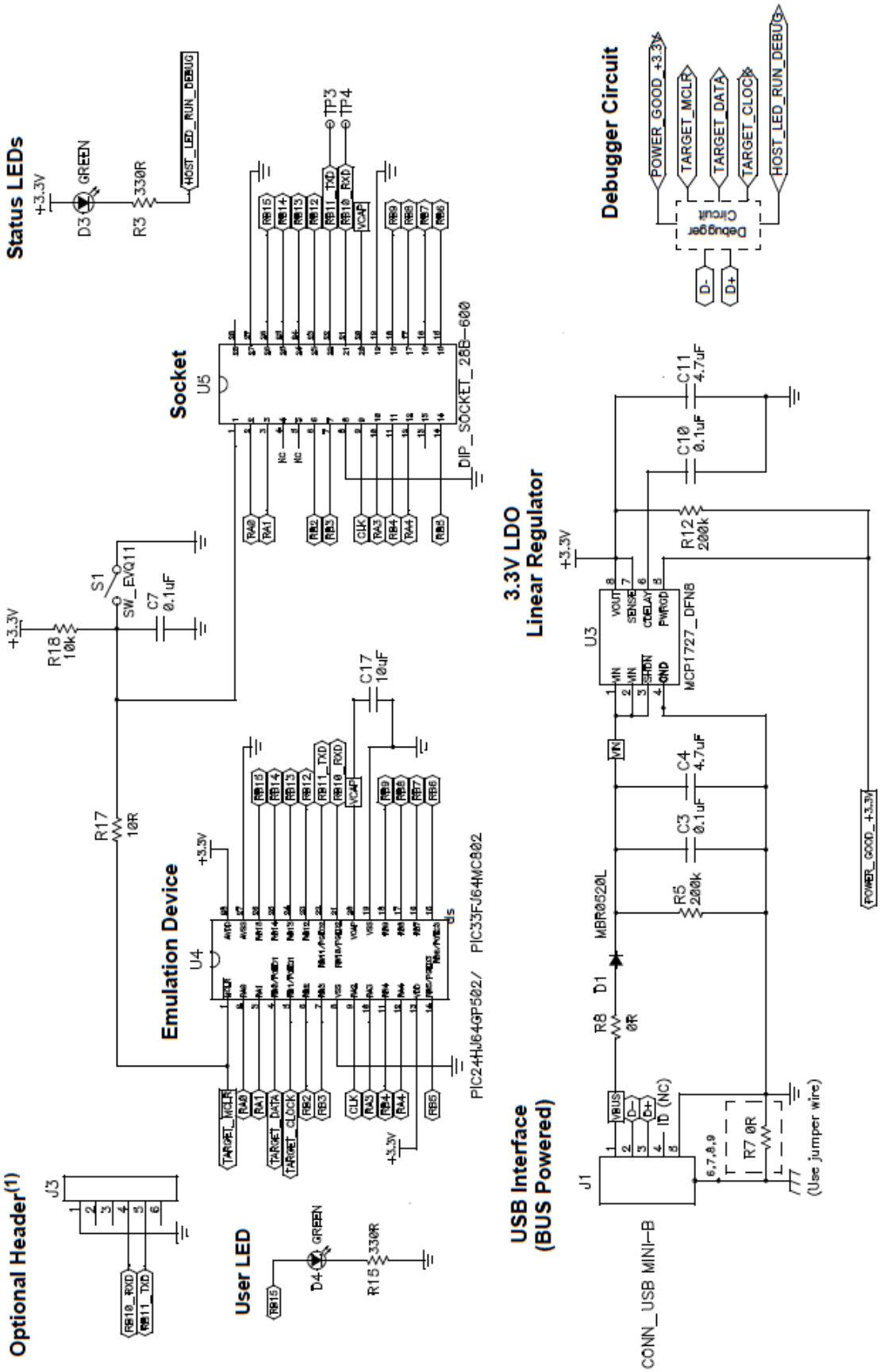
Código 2 Bluetooth y PWM:

PWM1.C

```
#include <stdio.h>
#include <stdlib.h>
#include <p33FJ128MC802.h>


//_FPOR(RST_PWMPIN & PWM1H_ACT_HI & PWM1L_ACT_HI);
_FPOR( HPOL_ON &  PWMPIN_OFF & LPOL_ON );
void PWM_INIT(void){

            PMD1bits.PWM1MD=0;
            PMD3bits.PWM2MD=0;
             P1TCON = 0;
             SEVTCMP = 0;
             P1TCONbits.PTEN   =1;
             P1TCONbits.PTSIDL =0;
             P1TCONbits.PTOPS  =0;
             P1TCONbits.PTCKPS =0;
             P1TCONbits.PTMOD  =0b00;
             P1TMR  = 0;
             P2TCON = 0;
             P2TCONbits.PTEN   =1;
             P2TCONbits.PTSIDL =0;
             P2TCONbits.PTOPS  =0;
             P2TCONbits.PTCKPS =0;
             P2TCONbits.PTMOD  =0b00;
             P2TMR  = 0;
             P1TPER = 0X67FD;//Fpwm=601 Hz,Fcy=16MHz
             P2TPER = 0X67FD;//Fpwm=601 Hz,Fcy=16MHz
             PWM1CON1bits.PMOD3 = 1; // PWM in independent mode
             PWM1CON1bits.PMOD2 = 1; // PWM in independent mode
             PWM1CON1bits.PMOD1 = 1; // PWM in independent mode
             PWM2CON1bits.PMOD1 = 1; // PWM in independent mode


  PWM1CON1bits.PEN1H = 1; // PWM High pin is enabled
  PWM1CON1bits.PEN2H = 1;
  PWM1CON1bits.PEN3H = 1; // PWM HIGH pin enabled
  PWM2CON1bits.PEN1H = 1;

  PWM1CON2bits.IUE=1;
  PWM1CON2bits.OSYNC=1;
  PWM1CON2bits.UDIS=0;
  PWM2CON2bits.IUE=1;
  PWM2CON2bits.OSYNC=1;
  PWM2CON2bits.UDIS=0;

     P1DC1 =1200;
     P1DC2=1200;
     P1DC3=40;
     P2DC1=1200;



     };


MAIN.C
/*
 * File:   MAIN.c
 * Author: michaellevkovsky
 *
 * Created on 6 de febrero de 2013, 14:25
 */

#include <stdio.h>
#include <stdlib.h>
#include<p33FJ128MC802.h>

/*

 *
 */
void PWM_INIT(void);
```

```
int main(int argc, char** argv) {
    AD1PCFGL = 0xffff;
PWM_INIT();
    return (EXIT_SUCCESS);
};

PWM1.H
/*
 * File:   PWM1.h
 * Author: michaellevkovsky
 *
 * Created on 6 de febrero de 2013, 14:24
 */

#ifndef PWM1_H
#define     PWM1_H

#ifdef      __cplusplus
extern "C" {
#endif



#ifdef      __cplusplus
}
#endif

#endif      /* PWM1_H */

void PWM_INIT(void);
```

Esquemático MICROSTICK II:



Esquemático del sistema embebido: