



UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE GUAYAQUIL

FACULTAD DE INGENIERÍAS

CARRERA:

INGENIERÍA ELECTRÓNICA

Tesis previa a la obtención del Título de:

**INGENIERO ELECTRÓNICO MENCIÓN EN SISTEMAS
COMPUTACIONALES**

TÍTULO

**DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE
PLANIFICACIÓN DE RUTAS Y NAVEGACIÓN PARA UN
VEHÍCULO AUTÓNOMO NO TRIPULADO USANDO
SOFTWARE LIBRE.**

AUTORES:

**Maximiliano Zambrano Palma
William Omar Ortega Villón**

DIRECTOR:

Ing. Byron Lima

Guayaquil, Marzo del 2015

DECLARATORIA DE RESPONSABILIDAD

Los conceptos desarrollados, análisis realizados y las conclusiones del presente trabajo, son de exclusiva responsabilidad de los autores

Guayaquil, Marzo del 2015

Maximiliano Eddy Zambrano Palma
CI # 1714750260

William Omar Ortega Villón
CI # 0922712526

DEDICATORIA

A Dios primeramente por haberme dado la sabiduría y la fuerza para seguir en este camino a pesar de las dificultades y los obstáculos de cada día poniendo mi fe en mis oraciones la cual a ser perseverante y paciente y estos son los logros por la cual me siento realizado como persona.

A mi madre **Estrella Palma R.**

Que me ha enseñado a lo largo de la vida seguir adelante y alcanzar mis sueños teniendo siempre su apoyo incondicional.

A mi esposa Carolina y mi hija Sarah por ser mi motivación e inspiración para seguir superando las pruebas que se me presente

A mi hermano Erwin y hermana Yita por ser el ejemplo de que si se puede con esfuerzo y ganas a llegar a la meta.

Y por último a todos aquellos que me brindaron su ayuda en su debido momento, a todos los chicos del club de robótica por su apoyo moral. Al Ing. Byron Lima por alentarme a seguir con el proyecto, una mención especial a mi amigo Pablo Ramírez por ayudarme en todas la pruebas.

Maximiliano Eddy Zambrano Palma

DEDICATORIA

Dedico este proyecto y toda mi carrera universitaria a Dios por ser quien ha estado a mi lado en todo momento, dándome las fuerzas necesarias para continuar luchando día tras día y seguir adelante superando las metas trazadas. A mi padre Marcos Ortega y a mi madre María Villón, ya que gracias a ellos soy quien soy hoy en día; ellos fueron los que me dieron ese cariño y protección, son los que han velado por mi salud, mi educación y alimentación.

A mis hermanos porque ellos me dieron su apoyo y me animaron a seguir en los momentos más difíciles.

William Ortega Villón

AGRADECIMIENTO

En la Presente Tesis primero agradezco a Dios, por cuidarme darme salud para poder llegar a terminar la carrera Universitaria.

A mis padres por todos los consejos dados, amor y comprensión.

A mi esposa y mi hija, por su comprensión y su ánimo de seguir mis sueños.

A todos mis amigos, maestros, compañeros de aulas por ser parte del desarrollo profesional.

Son muchas las personas que han formado parte de mi vida, que gustaría agradecerles por su amistad y compañía en momentos difíciles. Sin importar donde estén gracias por todo el apoyo brindado.

Maximiliano Eddy Zambrano Palma

AGRADECIMIENTO

Quiero agradecer a todos mis maestros ya que ellos me enseñaron a valorar los estudios y superarme cada día, también agradezco al Ing. Byron Lima por la paciencia que nos ha tenido durante este proceso, sus consejos, palabras de ánimo que nos supo manifestar en momentos de desánimo.

Y agradezco a Dios por permitirme culminar esta etapa de mi vida con salud. Y con las ganas de seguir adelante; con la convicción de vivir para servir.

William Ortega Villón

ÍNDICE GENERAL

CARÁTULA.....	I
DECLARATORIA DE RESPONSABILIDAD.....	II
DEDICATORIA.....	III
AGRADECIMIENTO.....	V
ÍNDICE GENERAL.....	VII
ÍNDICE DE TABLAS.....	XI
ÍNDICE DE GRÁFICOS.....	XII
RESUMEN.....	XV
ABSTRACT.....	XVI
INTRODUCCIÓN.....	1
CAPÍTULO I. EL PROBLEMA.....	3
1.1. PLANTEAMIENTO DEL PROBLEMA.....	3
1.2. DELIMITACIÓN DEL PROBLEMA.....	3
1.3 OBJETIVOS.....	3
1.3.1 OBJETIVO GENERAL.....	3
1.3.2 OBJETIVOS ESPECÍFICOS.....	4
1.4 JUSTIFICACIÓN.....	4
1.5 HIPÓTESIS.....	4
1.6 VARIABLES E INDICADORES.....	5
1.7. METODOLOGÍA.....	5
1.7.1. MÉTODOS.....	5
1.7.2. TÉCNICAS.....	5
1.7.3. INSTRUMENTOS INVESTIGACIÓN Y RECOLECCIÓN DE DATOS... ..	6
1.8 POBLACIÓN Y MUESTRA	6
1.8.1 POBLACIÓN.....	6
1.8.2 MUESTRA.....	6
1.9 DESCRIPCIÓN DE LA PROPUESTA.. ..	7
1.9.1 BENEFICIARIOS.....	8
1.9.2 IMPACTO.....	8
CAPÍTULO 2.MARCO TEÓRICO.....	9
2.1. ANTECEDENTES.....	9

2.2. FUNDAMENTACIÓN TEÓRICA.....	9
2.2.1 VUELO AUTÓNOMO.....	9
2.2.2 GPS.....	9
2.2.3 CONVERTIDOR FTDI 232R.....	11
2.2.4 ACELERÓMETROS.....	12
2.2.5 GIRÓSCOPIO.....	13
2.2.6 WI-FI.....	14
2.2.7 PAPAARZZI SOFTWARE Y HARDWARE.....	15
2.2.8 DRONE.....	16
2.2.8.1 ESPECIFICACIONES FÍSICAS.....	16
2.2.8.2 SISTEMA INFORMÁTICO INTEGRADO.....	17
2.2.8.3 SISTEMA DE SEGURIDAD.....	18
2.2.8.4 ESTRUCTURA AERODINÁMICA.....	18
2.2.8.5 MOTORES Y ENERGÍA.....	19
2.2.8.6 CÁMARA FRONTAL.....	19
CAPÍTULO 3. IMPLEMENTACIÓN DEL SISTEMA.....	20
3.1 PAPAARZZI CENTER.....	20
3.1.1. CONFIGURACIÓN.....	20
3.1.2. COMPILACIÓN.....	21
3.1.3 EJECUCIÓN.....	21
3.1.4 HERRAMIENTAS.....	21
3.2 CONFIGURACIÓN DE LA ESTRUCTURA DEL AVIÓN.....	23
3.2.1. CREACIÓN DE UN NUEVO AVIÓN.....	24
3.2.1.1. DEFINICIONES DE FIRMWARE Y HARDWARE.....	24
3.2.1.2. DEFINICIÓN DE HARDWARE.....	25
3.3. SUBSISTEMAS.....	25
3.3.1. IMU.....	25
3.3.2. AHRS.....	26
3.3.3. RADIO CONTROL.....	26
3.3.4. TELEMETRÍA (Modem).....	27
3.3.5. GPS.....	27
3.3.6. PARÁMETROS XML.....	28
3.3.7. ARQUITECTURA DEL SISTEMA.....	30

3.4. CONFIGURACIÓN ROTORCRAFT.....	31
3.4.1. DEFINICIONES DE FIRMWARE Y HARDWARE.....	31
3.4.2. PROGRAMACIÓN DE SELECCIÓN DE BOARD.....	31
3.4.3. PROGRAMACIÓN DE ACTUADORES.....	32
3.4.4. PROGRAMACIÓN INS.....	32
3.4.5. ARMADO MOTOR	32
3.4.6. MODOS DEL PILOTO AUTOMÁTICO	33
3.4.7. PROGRAMACIÓN PARÁMETROS XML.....	36
3.4.8. MEZCLA MOTOR.....	37
3.4.8.1 EJEMPLOS MEZCLA	38
3.5. FUSION HARDWARE Y SOFTWARE.....	43
3.5.1 CONVERTIDOR USB Y GPS.....	47
3.6 CGS RUTA DE VUELO	47
3.7 PRODUCTO TERMINADO.....	49
3.8 PRUEBAS DE FUNCIONABILIDAD.....	50
3.8.1 CARACTERÍSTICA TÉCNICAS DE OPERACIÓN.....	51
CAPÍTULO 4. PRÁCTICAS DE LABORATORIO.....	56
4.1 PRÁCTICA # 1 MONITOREO DE SENSORES	56
4.1.2 CÓDIGO DE LA PROGRAMACIÓN FIRMWARE.....	61
4.1.3 DIAGRAMA DE CONECTIVIDAD	62
4.2 PRÁCTICA # 2 CONTROL DE MOTORES.....	65
4.2 .1 DIAGRAMA DE CONECTIVIDAD	65
4.2 .2 CÓDIGO DE PROGRAMACIÓN JOYSTICK.....	67
4.3 PRÁCTICA # 3 PROGRAMACIÓN DE ASCENSO Y DESCENSO.....	70
4.4 PRÁCTICA # 4 PROGRAMACIÓN DE MOVIMIENTO DE ROTACIÓN... ..	73
4.5 PRÁCTICA # 5 PROGRAMACIÓN DE MOVIMIENTO ADELANTE Y HACIA ATRÁS.....	75
4.6 PRÁCTICA # 6 PROGRAMACIÓN DE DERECHA A IZQUIERDA.....	78
4.7 PRÁCTICA # 7 SEGUIMINETO DE TRAYECTORIA.....	81
4.7.1 CÓDIGO DE PROGRAMACIÓN PLAN DE VUELO.....	84
4.8. CÓDIGO COMPLETO DEL AIRFRAME.....	86
CONCLUSIONES.....	93
RECOMENDACIONES.....	94

CRONOGRAMA.....	95
PRESUPUESTO DE LA PLANTA.....	96
REFERENCIAS.....	97
ANEXOS.....	99

ÍNDICE DE TABLAS

TABLA 1. DESCRIPCIÓN DE SEÑALES GPS.....11

TABLA 2. DESCRIPCIÓN DE SEÑALES CONVERTIDOR FTDI 232R..... 12

ÍNDICE DE GRÁFICOS

FIGURA 1: GPS MODELO UBLOX LEA-6H.....	10
FIGURA 2: CONVERTIDORES FTDI232R, USB – UART.....	11
FIGURA 3: PLACA AIR DRONE2.0.....	13
FIGURA 4: PRECESIÓN GIROSCÓPICA.....	13
FIGURA 5: IMAGEN DE COMUNICACIÓN PC-DRONE POR MEDIO DE WI-FI.....	14
FIGURA 6: CONJUNTO DE LA PLATAFORMA Y LA ESTACIÓN TERRENA.....	15
FIGURA 7: PARROT AIR DRONE 2.0.....	16
FIGURA 8: PLACA AIR DRONE.....	17
FIGURA 9: PROTECTORES DE HÉLICES.....	18
FIGURA 10: HÉLICES CON VARIADOR DE VELOCIDAD.....	18
FIGURA 11: CÁMARA INTEGRADA	19
FIGURA 12: MENÚ PLATAFORMA PAPANAZZI.....	23
FIGURA 13: ARCHIVO DE CONFIGURACIÓN BÁSICO DEL FUSELAJE.....	24
FIGURA 14: DETALLE DE FIRMWARE	25
FIGURA 15: LÍNEAS DE DEFINICIONES HARDWARE.....	25
FIGURA 16: DECLARACION DEL IMU.....	26
FIGURA 17: SUBSISTEMA AHRS.....	26
FIGURA 18: DECLARACIÓN DE RADIO CONTROL.....	26
FIGURA 19: TELEMETRÍA DE SERIE TRANSPARENTE.....	27
FIGURA 20: DECLARACIÓN DEL GPS UBLOX.....	28
FIGURA 21. PROCESO DE CONSTRUCCIÓN (BUILD).....	29
FIGURA 22. COMUNICACIÓN ENTRE EL PAPANAZZI UAV.....	30
FIGURA 23. PLUS CRUZ.....	38
FIGURA 24. TIEMPO DE LA CRUZ.....	39
FIGURA 25. HEX (TIEMPO DE LA CRUZ).....	39
FIGURA 26. OCTO (ADEMÁS DE CRUZ).....	40
FIGURA 27. ESQUEMA DE CONTROL AR.DRONE2.0.....	44
FIGURA 28. PANTALLA PRINCIPAL PAPANAZZI CENTER.....	45
FIGURA 29. CONEXIÓN AL WI-FI DE ARDRONE2.....	45

FIGURA 30. CARGA A/C DEL PILOTO AUTOMÁTICO (BOARD).....	46
FIGURA 31. PARA ESCOGER ARDRONE VUELO AUTONOMO.....	46
FIGURA 32. DIAGRAMA DE CONEXIÓN ARDRONE.....	47
FIGURA 33. DIAGRAMA DE BLOQUE FT232R.....	47
FIGURA 34. GCS CON AZULEJOS CARGADOS.....	48
FIGURA 35. EL ARDRONE2.0 CON NUEVO PROTECTOR.....	49
FIGURA 36. VISTA INTERNA DE LA DISTRIBUCIÓN.....	49
FIGURA 37. PILOTO DEL DRONE EN MODO MANUAL.....	51
FIGURA 38. PRUEBA EN LOS PATIOS.....	52
FIGURA 39. PRUEBA EN LOS PATIOS DESPEGUE.....	52
FIGURA 40. PRUEBA EN LOS PATIOS RUTEO.....	53
FIGURA 41. PRUEBA EN LOS PATIOS RUTEO ANTES DEL DECENSO...	53
FIGURA 42. PRUEBA EN LOS PATIOS RUTEO DECENDIENDO.....	54
FIGURA 43. PRUEBA EN LOS PATIOS ATERRIZADO.....	54
FIGURA 44. DIGRAMA DE CONEXIÓN PC –UART- GPS.....	56
FIGURA 45. PANTALLA PRINCIPAL DEL SOFTWARE U-CENTER V8.11.....	57
FIGURA 46. SUBROUTINA DE PARA LA CONFIGURACIÓN DEL GPS.....	57
FIGURA 47. SEGMENTO DEL RAMAL UBX – CFG – NAV5.....	58
FIGURA 48. SEGMENTO DEL RAMAL PTR SE CONFIGURA EL VALOR DE BAUDRATE.....	58
FIGURA 49 SE CONFIGURA BAUDRATE A 38400 COMO RANGO MÁXIMO.....	59
FIGURA 50. SE CAMBIA EL PERIODO DE MEDICIÓN.....	59
FIGURA 51. SE DESHABILITA LA SUB RUTINA SBAS.....	60
FIGURA 52. HABILITACION DE PARAMETROS EN NAV.....	60
FIGURA 53. GUARDA CAMBIOS EN GPS.....	61
FIGURA 54. DIAGRAMA DE CONEXIÓN DE LOS ELEMENTOS.....	62
FIGURA 55. COMPILACIÓN DEL AIRFRAME.....	63
FIGURA 56. COMPILACIÓN SIN ERRORES DEL AIRFRAME.....	63
FIGURA 57. PANTALLA GCS (CONTROL TIERRA).....	64
FIGURA 58. PANTALLA GCS DEL SOFTWARE LIBRE.....	64
FIGURA 59. DIAGRAMA DE CONEXIÓN DE LOS ELEMENTOS CON	

JOYSTICK.....	65
FIGURA 60. SOFTWARE CONFIGURACION CONDICIÓN USO DE JOYSTICK.....	66
FIGURA 61. VERIFICAR EL PROCESO DE ACTIVACIÓN DEL JOYSTICK.....	66
FIGURA 62. UBICACIÓN DE LA CARPETA DONDE EL ARCHIVO DEBE ESTAR GUARDADO	67
FIGURA 63. PRIMERA PRUEBA ENCENDIDO.....	68
FIGURA 64. MOTORES YA EN FUNCIONAMIENTO.....	68
FIGURA 65. DEMOSTRACIÓN DE APAGADO.....	68
FIGURA 66. PRESENTACIÓN DE PANTALLA GCS.....	70
FIGURA 67. EN LA PARTE INFERIOR SE ENCUENTRA EL MANDO DE ACELERACIÓN.....	71
FIGURA 68. EN LA FIGURA DE LA IZQUIERDA EN SEGMENTO PFD....	71
FIGURA 69. BOTÓN B3 LA FUNCIÓN DE YAW.....	73
FIGURA 70 GIROS.....	74
FIGURA 71. MOVIMIENTO HACIA EL FRENTE.....	75
FIGURA 72. MOVIMIENTO DESDE HOME.....	76
FIGURA 73. NUEVA ORDEN DE MANDO.....	76
FIGURA 74. APLICANDO UN MOVIMIENTO HACIA ATRÁS.....	76
FIGURA 75. CAMBIO DE RUTA HACIA ATRÁS.....	78
FIGURA 76. APLICANDO UN MOVIMIENTO HACIA LA DERECHA.....	79
FIGURA 77. RUTA DE VUELO DERECHA.....	79
FIGURA 78. APLICANDO UN MOVIMIENTO HACIA EL LADO IZQUIERDO.....	79
FIGURA 79 CAMBIO EN EL PLANO HACIA LA IZQUIERDA.....	81
FIGURA 80 CONTROL DE MANDO JOYSTICK.....	82
FIGURA 81 CAMBIO HOME A STDBY.....	82
FIGURA 82. MOVIMIENTO DESDE STDBY A P1.....	83
FIGURA 83. BLOQUE DE RUTA GO.....	83
FIGURA 84 RUTA LINE.....	84

RESUMEN

AÑO	ALUMNOS	DIRECTOR DE TESIS	TEMA TESIS
2015	Zambrano Palma, Maximiliano Eddy Ortega Villón William Omar	ING. BYRON LIMA CEDILLO	DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE PLANIFICACIÓN DE RUTAS Y NAVEGACIÓN PARA UN VEHÍCULO AUTÓNOMO NO TRIPULADO USANDO SOFTWARE LIBRE

El uso de vehículos no tripulados para labores de inteligencia se remonta al menos a mediados del siglo XX, aunque ha sido en esta década cuando se han hecho mundialmente conocidos por sus aplicaciones civiles, militar y de entretenimiento.

Por este motivo surge la necesidad de interactuar con estos dispositivos, pero interactuar. ¿De qué forma?. De una manera más objetiva. Actualmente estos drones son utilizados por aplicaciones que tienen un costo y que solo realizan tareas específicas; son controladas manualmente, por este motivo surge la necesidad de encontrar herramientas tecnológicas que faciliten la planificación de rutas específicas sin la necesidad de la intervención de un guía. Por esta necesidad se ha elaborado la siguiente tesis “DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE PLANIFICACIÓN DE RUTAS Y NAVEGACIÓN PARA UN VEHÍCULO AUTÓNOMO NO TRIPULADO USANDO SOFTWARE LIBRE”. La implementación fue realizada utilizando el sistema operativo linux (ubuntu 12.4) para manejar el software paparazzi con el cual se puede configurar las herramientas necesarias para el manejo de los componentes que se utilizaron en la implementación. El aplicativo se comunica con el drone mediante una conexión Wi-Fi con alcance de 40 metros aproximadamente. Este carga el plan de vuelo previamente compilado y configurado para que este lo realice.

PALABRAS CLAVES:

Drone, Paparazzi, Vuelo autónomo, GPS, Acelerómetros, Giroscopio, Wi-fi.

ABSTRACT

YEAR	STUDENTS	THESIS DIRECTOR	THESIS THOPIC
2015	Zambrano Palma, Maximiliano Eddy Ortega Villón William Omar	ING. BYRON LIMA CEDILLO	DESIGN AND IMPLEMENTATION OF A SYSTEM OF PLANNING ROUTES AND NAVIGATION FOR UNMANNED AUTONOMOUS VEHICLE USING FREE SOFTWARE

The use of drones for intelligence goes back at least to the mid-twentieth century, although it was in this decade that has become known worldwide for their civil, military and entertainment applications.

Therefore there is a need to interact with these devices, but interact in what way? In a more objective manner. Currently these drones are used by applications that have a cost and only perform specific tasks; are controlled manually, for this reason there is a need to find technological tools to facilitate the planning of specific routes without requiring the intervention of a guide. For this need has developed the following thesis "DESIGN AND IMPLEMENTATION OF A NAVIGATION SYSTEM PLANNING ROUTES AND FOR UNMANNED AUTONOMOUS VEHICLE USING FREE SOFTWARE". The implementation was done using the Linux operating system (ubuntu 12.4) to handle the paparazzi software with which you can set necessary for the management of components that were used in the implementation tools. The application communicates with the drone via a wireless connection range of about 40 meters. This will load the flight plan previously compiled and configured so that it does it.

KEYWORDS:

Drone, Paparazzi, autonomous flight, GPS, accelerometers, gyroscope, Wi-Fi.

INTRODUCCIÓN

Este proyecto de tesis va enfocado a la dotación de elementos y conceptos de programación, sobre el manejo de forma manual y autónoma de un vehículo no tripulado (UAV).

. Aplicación del software libre paparazzi, el cual permite interactuar con los diferentes sensores y elementos de cada una de las prácticas.

. Ardrone 2.0 con conexión Wi-Fi, el cual permite la comunicación entre software y los elementos electrónicos.

. Sensores de aceleración giro y altura, módulos electrónicos como GPS, convertidor FTDI 232R y joystick 3AXIS.

. Una laptop con sistema operativo Ubuntu 12.4.

El proyecto se desarrolla de manera didáctica en las diferentes prácticas, introducción y preparación del uso de sus sistemas y módulos que permiten la compilación y la carga de un sistema de vuelo autónomo.

El programa está diseñado para el uso exclusivo de sistema de vuelos no tripulados, incentiva la investigación en temas como la robótica y programación de microprocesadores.

En el Capítulo 1, se detallan los hechos que motivaron a ejecutar este tema de tesis, dentro de este capítulo están. El planteamiento del problema, la justificación del tema de tesis, los objetivos, la técnicas empleadas para su desarrollo, sus beneficiarios y el impacto que genera el tema.

En el Capítulo 2, se detalla la parte teórica de los componentes del software y de la aeronave (Drone) y los principios utilizados para desarrollar el tema de tesis.

En el Capítulo 3, se detalla todo el proceso del uso del software la programación necesaria para el manejo de todos los módulos que se utilizaron, así como la obtención del controlador de estabilidad, giro y aceleración de la aeronave (Drone), además de las pruebas de funcionamiento de la aeronave.

En el Capítulo 4, Constan las prácticas que fueron propuestas en el anteproyecto. Demostrando gráficamente cada uno de los procedimientos que se efectuaron para llegar al resultado obtenido.

CAPÍTULO I.

PLANTEAMIENTO DEL PROBLEMA

1.1. PLANTEAMIENTO DEL PROBLEMA:

La necesidad de incursionar en nuevas herramientas tecnológicas presenta la oportunidad de explorar el manejo de sistemas de estabilización de voladores no tripulados y puesto que la sede Guayaquil de nuestra Universidad, actualmente no cuenta con este tipo de proyectos, se ha visualizado de alguna manera empezar con el estudio de estos sistemas y aportar con un proyecto de estas características.

1.2. DELIMITACIÓN DEL PROBLEMA

Espacial.- Este de proyecto de tesis se lleva a cabo en la Universidad Politécnica Salesiana sede Guayaquil.

Temporal.- Para este trabajo de fin de carrera se tomó más del tiempo acordado a partir de su aprobación de un año.

Académica.- Para ejecutar este proyecto se puso en práctica los métodos de investigación, los conocimientos adquiridos en electrónica microcontroladores, control automático, programación; durante el periodo de estudiantes en la Universidad Politécnica Salesiana.

1.3 OBJETIVOS

1.3.1 Objetivo General

Diseñar e implementar un sistema de control de vuelo para un cuadricoptero utilizando la plataforma Paparazzi basado en Linux.

1.3.2 Objetivos Específicos

- Promover la investigación de sistemas de navegación no tripulados UAV
- Analizar e interpretar el funcionamiento de hardware y software libre.
- Verificar el comportamiento de este con los distintos dispositivos a usar en el trabajo de titulación.
- Observar la dinámica de la estructura de la planta, definiendo los parámetros constructivos principales y actuadores.

1.4 JUSTIFICACIÓN

A nivel mundial el uso de estos sistemas se lo realiza de manera comercial, los sistemas de Control Autónomos, tienen un bajo índice de uso y creación, debido a esto se incursiona en este tipo de proyectos que en su mayoría manejan una arquitectura cerrada.

En la actualidad la Universidad cuenta con un club de robótica, el mismo que no se ha introducido en la investigación ni elaboración de proyectos con sistema UAV (Rotorcraft), por tal motivo se fomenta la investigación en los estudiantes en proyectos con estas características.

La propuesta es dotar de un prototipo para futuras aplicaciones, para que el alumnado se incentive a realizar este tipo de proyectos o a su vez mejorar el sistema propuesto. Impulsar el uso de software libre, la libertad de usar un programa para cualquier propósito, la tecnología de software libre nos permite compartir conocimientos, promover la investigación, investigación que es devuelta a la sociedad.

1.5 HIPÓTESIS.

Mediante el diseño y la implementación de un sistema de vuelo autónomo, se apunta al uso software libre (paparazzi) y al conocimiento del sistema operativo Linux (Ubuntu 12.4), que pueden ser usados por los estudiantes de la materia de programación y robótica de la carrera de electrónica.

1.6 VARIABLES E INDICADORES:

Variable independiente: Elaboración de las practicas utilizando los módulos electrónicos, GPS convertidor Uart y Joystick.

Variable dependiente: Implementación de un sistema de planificación de rutas y navegación para un vehículo autónomo no tripulado.

- **Indicador 1.** Estabilidad de la aeronave.
- **Variable 1.** Estado de carga de batería.
- **Indicador 2.** Tiempo de vuelo autónomo.
- **Variable 2.** Altura máxima de la aeronave en vuelo.
- **Indicador 3.** Posición en el mapa de ruteo.
- **Variable 3.** Ubicación con el GPS.

1.7. METODOLOGÍA

1.7.1. MÉTODOS

Para desarrollar este trabajo de tesis se ha empleado método científico debido a la investigación de partes y equipos más óptimos para la realización del tema.

1.7.2. TÉCNICAS

Medición.- Se hicieron mediciones de posición, velocidad, altura y cualquier dato que permita tener una mejor información del comportamiento del sistema.

Investigación.- Se obtuvo información de proyectos con similares características, se compartió información con personas de diferentes nacionalidades y toda fuente de información posible encontrada en la web.

Experimentación.- Con el Joystick se realizaron pruebas de funcionamiento para evaluar características del drone, adicional se ejecutan varias simulaciones de vuelos y su alcance de comunicación con la PC, de no ser satisfactorias estas pruebas se realizan reajustes en la configuración y en la programación hasta alcanzar el objetivo esperado.

1.7.3. INSTRUMENTOS DE INVESTIGACIÓN Y RECOLECCIÓN DE DATOS.

Las fuentes de investigación para el trabajo desarrollado se tomaron de varias páginas del Internet pero principalmente de una wiki especializada en estos temas. Además de libros foros e intercambio de información con personas que se encuentran inmersos en este tema ya sea de aeromodelismo o de vuelos autónomos.

1.8 POBLACIÓN Y MUESTRA.

1.8.1 Población.

En este tema de tesis la población escogida ha sido los estudiantes de la carrera de Ingeniería Electrónica de la Universidad Politécnica Salesiana sede Guayaquil.

Las características obedecen a las materias ofrecidas por parte de la Universidad en la cuales se encuentra programación, microprocesadores, interfaces y robótica.

1.8.2 Muestra.

Se tomo como referencia a los estudiantes que conforman el club de robótica de la Universidad Politécnica Salesiana sede Guayaquil y los estudiantes que cruzan la materias de programación, robótica y sistemas microprocesadores.

1.9. DESCRIPCIÓN DE LA PROPUESTA.

La aeronave consta de una cruz de fibra de carbono con cuatro variadores de velocidad (ESC), con sus respectivos motores sin escobilla y propelas para cada motor, además del uso de un convertidor UART FTDI RS232, Un GPS Ublox LEA6H y un Joystick de 3 axis, la planta puede alcanzar alturas de 40mts, y una autonomía de vuelo de 15 minutos antes de que la interfaz grafica del sistema de control terrestre (GCS), nos indique batería baja. Entonces hay que hacer la recarga de la misma para continuar con los vuelos que se desee. Paparazzi utiliza control común Proporcional Integral Derivativo (PID) para la estabilidad y la navegación. PID es probablemente el diseño de control de realimentación más comúnmente utilizado, ya que es simple de implementar e intuitivo para operar. Los Controladores PID usan tres términos que operan en el error medido para producir una salida de control. Si $u(t)$ es la señal de control enviada al sistema, $y(t)$ es la salida medida y $r(t)$ es la salida deseada, y el error de seguimiento $e(t) = r(t) - y(t)$, donde un controlador PID tiene la forma general

$$u(t) = K_P e(t) + K_I \int e(t) dt + K_D \dot{e}(t) \quad (1)$$

Formula 1: Control de retroalimentación para el sistema (Paparazziuav, 2012).

La dinámica de bucles cerrados deseados se obtiene mediante el ajuste de los tres parámetros K_P , K_I y K_D , a menudo de forma iterativa por **ajuste** y sin conocimiento específico de un modelo de planta. La estabilidad a menudo se puede asegurar usando sólo el término proporcional en sistemas bien amortiguados. El término integral compensa los errores constantes a largo plazo, y el término derivado se usa para proporcionar amortiguación para reducir la oscilación.

Paparazzi utiliza controladores PID en todos los bucles, pero muchos de los términos I y D no se aplique plenamente ya que a menudo son innecesarios o difícil de ajustar.

La plataforma Paparazzi tiene 2 opciones de uso:

- **Modo Normal:** Para implementar los códigos de programación previo a un vuelo real, es necesario que sea simulado en PAPAARZZI, probado y validado para garantizar el funcionamiento del vehículo y tener un vuelo seguro.
- **Modo Didáctico:** Los estudiante durante las prácticas de programación o sistemas de microprocesadores, bajo su responsabilidad, podrán cambiar o crea nuevas rutas de vuelo, incrementar nuevos waypoints para poder observar y analizar el funcionamiento del vehículo.

1.9.1 BENEFICIARIOS

El trabajo de Tesis esta destinado para que sea utilizado por los estudiantes de la Universidad en especial los de la carrera de Ingeniería Electrónica en su mayoría estudiantes de la materia Programación, Sistemas Microprocesadores, y Robótica.

1.9.2 IMPACTO

En nuestra trayectoria como estudiantes, se recibieron materias como Programación, Control Automático, Interfaz, e inteligencia artificial, con este proyecto el estudiante tiene la oportunidad de experimentar e implementar nuevos segmentos de programación para interactuar con la interfaz de la Plataforma y ampliar así sus conocimientos en las materia de programación.

Otro impacto es que este vehículo no utiliza combustibles fósiles, sino que es eléctrico, ni emite ruido, por lo tanto respeta el medio ambiente.

CAPÍTULO II.

MARCO TEÓRICO

2.1. ANTECEDENTES.

Las siglas UAV corresponden en inglés Unmanned Aerial Vehicle es decir vehículo aéreo no tripulado. El origen de desarrollo de estos vehículos pertenecen a fines militares, ya que dan la posibilidad de realizar operaciones de alto riesgo, incluso la de sobrevolar una zona de desastre o de conflicto para la vigilancia o recogida de información. Existen una gran variedad de vehículos UAVs de grandes dimensiones para vuelos altos y de grandes distancias. Hasta pequeñas dimensiones llegando incluso hasta poco centímetros. (Fernandez, 2013).

2.2. FUNDAMENTACIÓN TEÓRICA.

2.2.1 VUELO AUTÓNOMO.

Los planes de vuelo se definen previamente al despegue y los drones suelen seguir estos planes con muy poca flexibilidad. En otros casos, los drones son maniobrados desde la base por una persona. En definitiva no son sistemas autónomos de vuelo que puedan tomar decisiones sobre el terreno en función de las nuevas condiciones que se produzcan, ni existe la posibilidad de enviar datos de una nueva ruta mientras está en vuelo. (Víctor Manuel Pérez Lozano, 2010).

2.2.2 GPS

El Sistema de Posicionamiento Global (GPS) es un dispositivo que permite a una persona determinar en todo el mundo la posición de un objeto, una persona o un vehículo con una precisión de hasta unos pocos metros. Es una utilidad de propiedad estadounidense, este sistema consiste en tres segmentos: el segmento espacial, el segmento de control, y el segmento de usuario.

La Fuerza Aérea de Estados Unidos desarrolla, mantiene y opera los segmentos espacial y de control.

Sistema de satélites. Está formado por 24 unidades con trayectorias sincronizadas para cubrir toda la superficie del globo terráqueo. Más concretamente, repartidos en 6 planos orbitales de 4 satélites cada uno. La energía eléctrica que requieren para su funcionamiento la adquieren a partir de dos paneles compuestos de celdas solares adosados a sus costados. (Gobierno, 2014).



Figura 1: Gps Ublox lea-6h. Modelo compatible con u-blox soluciones GPS para Android; Fabricado en ISO / TS 16949, realizado en plantas de producción certificadas. (ublox, 2014).

Este módulo GPS integra un receptor GPS LEA-6H con una antena de parche 25x25x4mm cerámica, interfaz serial UART y USB. El alto rendimiento de la u-blox motor de posicionamiento LEA-6H permite la navegación, incluso en entornos de señal débil.

Recomendado para multicopteros, donde la precisión del GPS es muy importante. Cuenta con circuitos activos para la antena con parches de cerámica y una batería de respaldo para arranques en caliente.

Este módulo cuenta con un puerto serie compatible USB V2.0 para la alimentación y la transferencia de datos. (hobbyking, 2012)

Tabla1.

Descripción de señales GPS.

Nombre	Tipo	Descripción
VCC	Input	Pin de entrada 5v
TX	Input	Trasmisión de dato de entrada.
RX	Output	Trasmisión de dato de salida.
GND	Gnd	Pin a Ground.

Nota: Muestra la descripción de las señales GPS, su tipo y su uso en la tesis.

Estaciones terrestres. Envían información de control a los satélites para controlar las órbitas y realizar el mantenimiento de toda la constelación.

Terminales receptores: Indican la posición en la que están; conocidas también como Unidades GPS, son las que podemos adquirir en las tiendas especializadas. (EEUU, 2014)

2.2.3 CONVERTIDOR FTDI 232R.

Los TTL USB convertidor, es un modulo serial por puerto USB que sirve para hacer interface con un computador o dispositivos con entrada UART.

La adquisición y envío de datos se lo realiza por medio de circuitos con microcontroladores, hacia el computador. Permite alimentar circuitos de 5V, proporciona acceso a UART para la transmisión (TX) y la recepción la proporciona (RX). (ftdichip, 2015).

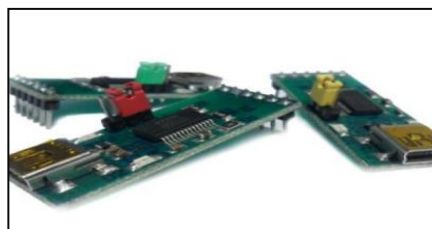


Figura 2: Convertidores FTDI232R, USB - UART I&T 02. Es un módulo de comunicación serial por puerto USB para hacer interface con un computador o PC, permite alimentar circuitos de 5V (Idetec, 2015).

Características del Convertidor ftdi 232R :

- Leds indicadores de TX y RX
- Transferencia de datos 300 Baud hasta 3MBaud
- Soporta 7 a 8 bits datos, 1 o 2 bits stop.
- Datos serial con amplitud seleccionables
- De 5V/3.3V
- Alimentación por interfaz USB desde la PC.

Tabla 2. Descripción de señales Convertidor FTDI 232R.

Nombre	Tipo	Descripción
VCC	Input	Pin de entrada 5v
TX	Input	Trasmisión de dato de entrada.
RX	Output	Trasmisión de dato de salida.
GND	Gnd	Pin a Ground.
RST	Output	Solicitud de envió.
CTS	Input	Borrado para enviar.

Nota: Muestra la descripción de señales con el tipo de entrada y para qué es utilizada según su simbología tomado de (ideastechnology, 2013).

2.2.4 ACELERÓMETROS

Los acelerómetros o sensores de aceleración, están pensados para realizar una medida de aceleración o vibración, proporcionando una señal eléctrica según la variación física, en este caso la variación física es la aceleración o la vibración.

Puede haber diferentes modelos combinando las diferentes tecnologías existentes, principalmente acelerómetros piezoresistivos, acelerómetros piezoeléctricos y acelerómetros capacitivos. (medidaes, 2012)



Figura 3: Placa Air drone 2.0, donde los acelerómetros están incluidos dentro de la placa Air Drone 2.0.

Los rangos de medida son diversos, desde 1 g, hasta los miles de g. Respecto al rango de frecuencia disponible, hay acelerómetros que parten de 0 Hz, para medida de bajas frecuencias, acelerómetros que llegan hasta los miles de Hz para altas frecuencias de vibración, otros modelos de muy alta sensibilidad con bajo rango de frecuencia, etc. (medida.es, 2012).

2.2.5 GIRÓSCOPIO.

Un giróscopo es un aparato en el cual una masa que gira velozmente alrededor de su eje de simetría, permite mantener de forma constante su orientación respecto a un sistema de ejes de referencia. Cualquier cuerpo sometido a un movimiento de rotación acusa propiedades giroscópicas, por ejemplo una peonza. Las propiedades giroscópicas fundamentales son: rigidez en el espacio y precesión.

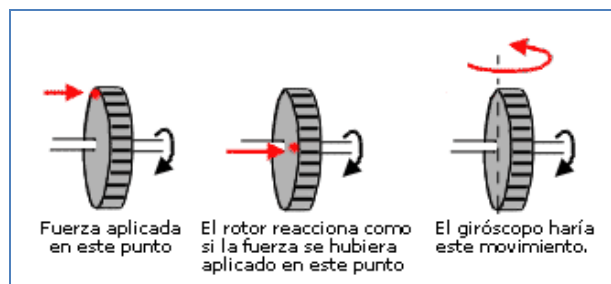


Figura 4: Precesión giroscópica. Se aplica una fuerza F , a un disco girando con velocidad angular ω , (Muñoz, M. A. ,7 de Enero de 2012).

La rigidez en el espacio se puede explicar por la primera Ley del Movimiento de Newton, que dice: "Un cuerpo en reposo tiende a estar en reposo, y un cuerpo en movimiento tiende a permanecer en movimiento en línea recta, salvo que se le aplique una fuerza externa". Siempre y cuando tenga suficiente velocidad, la fuerza de inercia que genera la peonza la hace girar erguida incluso si inclinamos la superficie sobre la cual gira, ofreciendo una gran resistencia a los intentos de volcarla o forzar su inclinación.

La segunda propiedad -precesión- es la respuesta del objeto cuando se le aplica una fuerza deflectiva en algún borde. Volviendo a la peonza, es la reacción de esta cuando en su rápido giro la tocamos en uno de sus bordes. El resultado de esta reacción es como si el punto de aplicación de la fuerza estuviera desplazado 90° en el sentido de giro del objeto. La precesión es inversamente proporcional a la velocidad de giro (a mayor velocidad menor precesión) y directamente proporcional a la cantidad de fuerza de deflexión aplicada. (Munoz, 2013)

2.2.6 WI-FI.

Wi-Fi es la sigla para Wireless Fidelity (Wi-Fi), que literalmente significa Fidelidad inalámbrica. Es un conjunto de redes que no requieren de cables y que funcionan en base a ciertos protocolos previamente establecidos. Si bien fue creado para acceder a redes locales inalámbricas, hoy es muy frecuente que sea utilizado para establecer conexiones a Internet.



Figura 5: Imagen de comunicación pc-drone mediante Wi-fi. La comunicación entre estos dos equipos es con un dispositivo extra conectado al drone como se muestra

Wi-Fi es una marca de la compañía (Wi-Fi Alliance) que está a cargo de certificar que los equipos cumplan con la normativa vigente, que en el caso de esta tecnología es la IEEE 802.11.

Esta nueva tecnología surgió por la necesidad de establecer un mecanismo de conexión inalámbrica que fuera compatible entre los distintos aparatos. En busca de esa compatibilidad fue que en 1999 las empresas 3com, Airones, Intersil, Lucent Technologies, Nokia y Symbol Technologies se reunieron para crear la Wireless Ethernet Compability Aliance (WECA), actualmente llamada Wi-Fi Alliance. (wordpress, 2011)

2.2.7 Paparazzi Software y Hardware.

Paparazzi es un software de código abierto, que contiene todo lo necesario para que un sistema aéreo no tripulado vuele con fiabilidad. El código de navegación y la estabilidad en el aire son altamente eficaces. El control de tierra es una aplicación muy avanzada que permite simulaciones previas a un plan de vuelo.

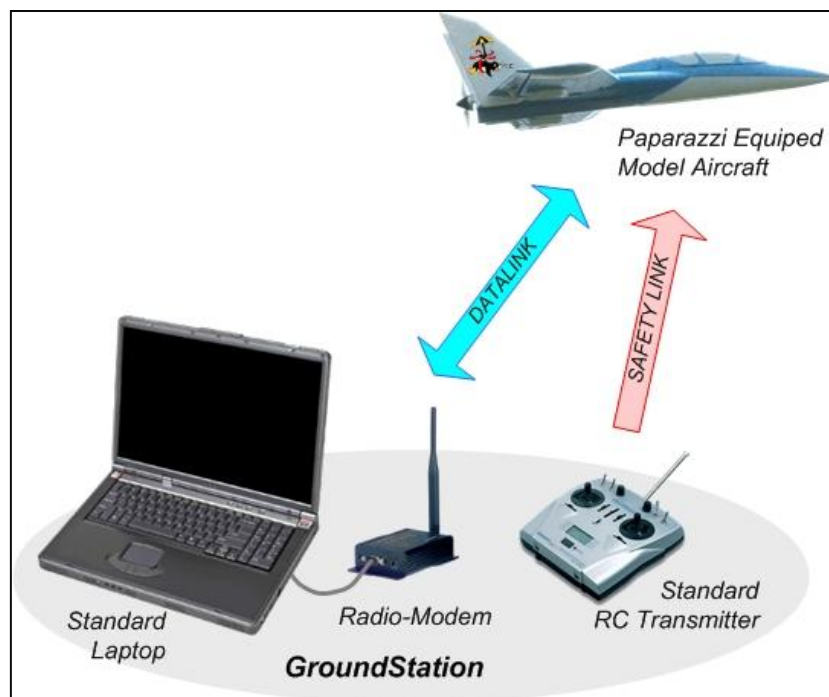


Figura 6: Visión de conjunto de la plataforma y la estación Terrena (paparazziuav, 2012).

El mismo que se resalta como un proyecto de hardware y de software libre y de código abierto que abarca un sistema de piloto automático excepcionalmente potente y versátil para aeronaves, por ser abierto permite agregar más funciones y mejorar el sistema, La planificación y el control se lo realiza por medio de software de monitoreo a través de un enlace de datos bidireccional donde juega un papel importante la telemetría y el control . (Paparazziuav, 2014).

2.2.8 DRONE.

Es un vehículo aéreo no tripulado , controlado con radio control, utilizado en diferentes campos; funciona propulsado por cuatro motores eléctricos Es similar en su estructura básica y aerodinámica a otros modelos radio controlados, pero se diferencia de todos ellos, ya que cuenta con un microprocesador y una serie de sensores; entre los cuales se incluyen dos cámaras que le permiten captar lo que ocurre a su alrededor, más un conector Wi-Fi integrado que le permite vincularse a dispositivos móviles personales o en este caso a una PC. (Parrot, 2014)



Figura 7: Parrot Air drone 2.0, este equipo es utilizado en el tema de las tesis por contar con las características utilizadas para nuestro proyecto.

2.2.8.1 ESPECIFICACIONES FÍSICAS.

Tamaño con casco: 52,5 x 51,5 cm

Tamaño Sin casco: 45 x 29 cm

Peso 380 g con la cubierta del casco para exteriores.420 g con la cubierta del casco para interiores.

Velocidad de marcha (cruce): 5 m/s, 18 km/h.

Techo máximo: Limitado por el alcance de la conexión Wi-Fi (entre 50 y 120 metros dependiendo de las condiciones climáticas.)

Alcance máximo: Limitado por el alcance de la conexión Wi-Fi.

Autonomía de vuelo promedio: 12 minutos (parrot.com, 2014).

2.2.8.2. SISTEMA INFORMÁTICO INTEGRADO.

El sistema integrado posee las siguientes características que se detallan a continuación.

Microprocesador ARM9 RISC de 32 bits @ 468 MHz

Memoria DDR SDRAM de 128 MB @ 200 MHz

Sistema operativo con núcleo Linux

Módem Wi-Fi b / g

Conector USB de alta velocidad

Sistemas de guía inercial

Acelerómetro de 3 ejes construido con tecnología MEMS.

Girómetro de 2 ejes

Girómetro de precisión de 1 eje para el control de guiñada. (parrot.com, 2014)



Figura 8: Placa Air drone, esta imagen muestra lo compacto de la placa con sus componentes recubiertos con un protector de espuma.

2.2.8.3. SISTEMA DE SEGURIDAD.

Dentro de las siguientes características de seguridad nos encontramos con las siguientes más importantes.

Bloqueo automático de las hélices en el caso de contacto.

Batería UL2054.

Interfaz de control con botón de emergencia para detener los motores.

Casco para vuelo en interiores con cubierta para las hélices.



Figura 9: Protectores de hélices, cubierta de espuma que evita el golpe directo con algún obstáculo dentro del plan de vuelo.

2.2.8.4. ESTRUCTURA AERODINÁMICA.

Posee cuatro hélices de alta eficiencia; estructura tubular de fibra de carbono.



Figura 10: Hélices con variador de velocidad y motor; el material de las hélices es plástico común solo la base es de fibra de carbono.

2.2.8.2.5 MOTORES Y ENERGÍA.

El drone consta de 4 motores sin escobillas, funcionando a 3.500 rpm con una potencia de 15 W. Batería de ion de litio de 3 celdas, capaz de entregar 1000 mA/hora con un voltaje nominal de 11,1 V, y que le permite una autonomía de vuelo de entre 12 y 15 minutos.

Capacidad de descarga: 10C.

Tiempo de carga: 45 minutos.

2.2.8.6 CÁMARA FRONTAL.

Este modelo de drone consta de una cámara con sensor CMOS de tipo gran angular de lente diagonal. 93 grados de amplitud.

Resolución 640x480 píxeles (VGA).

Respuesta en Frecuencia: 15 cuadros/s.

La codificación y transmisión en vivo de imágenes lo realiza hacia el dispositivo IOS. Cabe mencionar que en esta tesis no se va a utilizar la cámara, que no es el objetivo de esta propuesta. Simplemente se lo está detallando para conocimiento general.

Codificación y transmisión en vivo de imágenes en hacia el dispositivo IOS. (parrot.com, 2014)



Figura 11: Cámara integrada dentro del drone con resolución 640x480 píxeles (VGA).

CAPÍTULO III.

IMPLEMENTACIÓN DEL SISTEMA.

En este capítulo se detalla el uso del software, la programación necesaria para el manejo de todos los módulos elementos y dispositivos utilizados, así como la obtención del control de estabilidad, giro y aceleración de la aeronave.

3.1 PAPERAZZI CENTER.

El Centro de Paperazzi es una interfaz gráfica de usuario que contiene un grupo de tres módulos principales:

Posee un conjunto de cuadros de selección para configurar un avión con su plan de vuelo y construir los programas que se vayan a simular. Esta carga se lo puede realizar en la opción denominada dispositivo en el aire.

Otra opción es el panel de control que sirve para iniciar los agentes del sistema y manejar colección de programas como sesiones configurables

Un aplicativo parecido a una página web donde el GCS puede ser embebido.

Una consola de registro donde se expone las opciones del panel de configuración y control que básicamente es la descrita como panel de control.

3.1.1. CONFIGURACIÓN.

La parte izquierda de la página de configuración es un editor donde se modifica el archivo conf.xml. Si se quisiera configurar un nuevo avión se lo puede añadir en la opción menú de A/C. En esta misma opción se puede borrar otra configuración anterior.

El editor para los archivos de configuración se toma de la variable de entorno es decir un archivo ya creado que viene por defecto, gedit para Linux (es decir, utiliza el programa predeterminado para archivos .xml).

3.1.2. COMPILACIÓN.

Compilación y construcción se lo realiza desde el panel de construcción. Los nuevos objetivos o puntos referencia pueden ser añadidos a la lista desplegable con el botón Nuevo destino. La compilación y mandos intermitentes se muestran como agentes que se ejecutan libremente, de manera que puedan ser interrumpidas si es necesario.

3.1.3 EJECUCIÓN.

En el panel de ejecución, se encuentra un cuadro combinado que proporciona un conjunto de sesiones predefinidas por el usuario (colecciones de programas).

La sesión de simulación se ejecuta en un aplicativo llamado GCS que es donde se realiza la prueba de lo ya configurado. Tenga en cuenta que el objetivo sim o nps debe haber sido construido antes de la simulación.

Los programas que se ejecutan pueden detenerse y reiniciarse en las opciones (Detener / Rehacer). Las opciones también se pueden editar (en el cuadro de entrada). (Respawn). La opción revivir en automático se habilita estableciendo la casilla de verificación (lado izquierdo del botón Stop / Redo). El conjunto de los procesos actuales se puede guardar como una sesión de usuario (en tiempo real, en el archivo de configuración / control_panel.xml conf) para reiniciarlo más tarde si es necesario, entrando (en el menú de sesión).

3.1.4 HERRAMIENTAS (TOOLS).

El menú de herramientas contiene los programas más importantes para ejecutar todo lo relacionado a un plan de vuelo, donde se encuentran las siguientes opciones.

App Server. Un agente de servidor que conecta a un dispositivo Android mediante la aplicación PPRZonDroid.

Attitude Visualizer. Una herramienta muy específica para la depuración de un avión Quadshot.

Data Link. Describe la conexión a los módems. Donde se visualiza dos opciones

- Velocidad de transmisión: la que muestra por default -s 57600
- Dispositivo de vuelo donde aparece el siguiente mensaje: -d / dev / ttyUSB0 (reemplace ttyUSB0 con el puerto USB del módem real está enchufado).

Environment Simulator. Simulador de medio ambiente. Se utiliza para ordenar un tiempo de aceleración, establecer una dirección de intensidad del viento y simular fracaso de conexión del GPS durante la simulación de un plan de vuelo.

Flight Plan Editor. Para editar el plan de vuelo en extensión xml, ejemplo (plan.xml).

GCS. La estación central terrestre que a su vez contiene un conjunto de opciones de configuración.

GPSD position display. Enlace para conectar un módem GPS a su GCS. Puede ser útil si en la estación central terrestre está en movimiento.

Hardware in the Loop. Conexión de AP (pilotos automáticos) para la simulación en HITL.

Ivy2NMEA. Es un proxy para convertir la información IVY a NMEA, los mensajes de un bus en serie en un puerto de hardware.

Joystick. Herramienta para usar un joystick para enviar comandos a la aeronave utilizando Input2Ivy.

Link combiner. Para establecer un enlace de datos redundantes (múltiples enlaces de datos para la redundancia / seguridad).

Log File Player. Para reproducir los archivos login.

Log Plotter. Un Plotter Log.

Messages. Muestra todos los mensajes del enlace de datos (mensaje observador de bus).

Messages (Python). Muestra todos los mensajes del (mensajes observadores de enlace de datos).

NatNet. Es una aplicación proxy para transformar la posición, velocidad y otros datos pertinentes recibidos en más de una red desde un system Optitrack en el bus ivy.

Real-time Plotter. Se puede visualizar los datos recibidos desde Paparazzi_Center, el números de mensajes en tiempo real (sólo hay que arrastrar y soltar desde los mensajes).

Server. El servidor es un agente que registra, distribuye y procesa previamente los mensajes para el GCS y otros agentes es la descripción general del sistema.

Settings. Una "forma" importante donde en tiempo real se puede, dar parámetros a la aeronave. Esta es la versión ventana independiente de lo que está incrustado en el GCS.

Settings (Python). Una alternativa a los ajustes regulares "form"

Simulator. Simulador se puede construir con los sim / jbsim / nps como destino en el panel de control.

Weather Station. Es una interfaz a una de las estaciones meteorológicas.

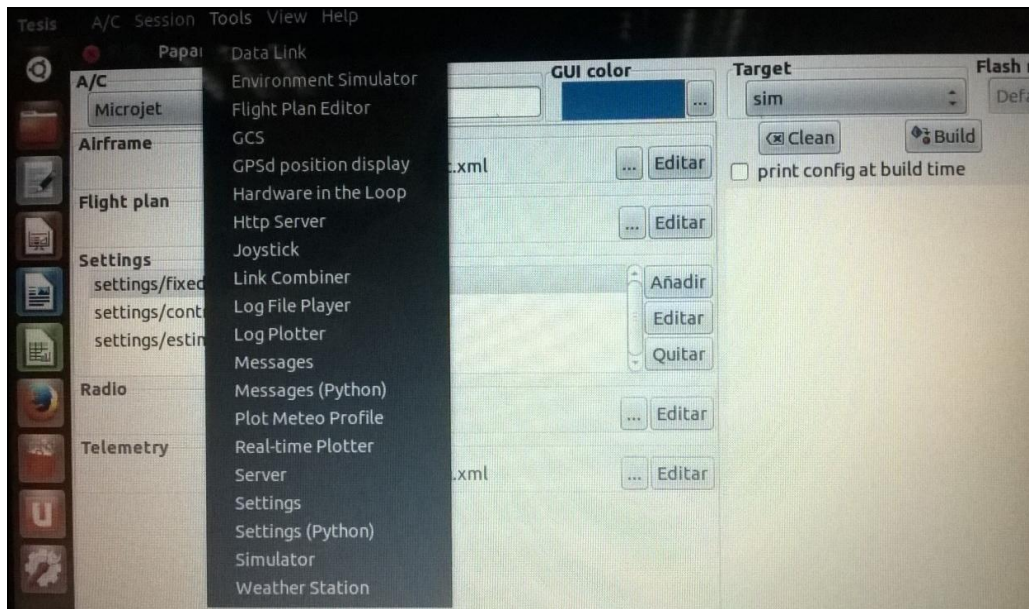
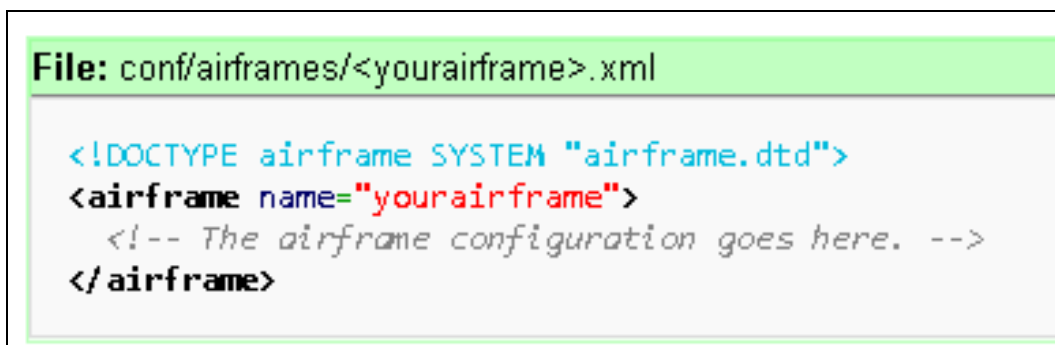


Figura 12: Menú plataforma paparazzi una vez ejecutada la aplicación en un terminal, en la parte superior seleccionamos herramientas (Tools).

3.2 CONFIGURACIÓN DE LA ESTRUCTURA DEL AVIÓN

El fuselaje es el archivo más importante, contiene todas las configuraciones de hardware y de software. En él se describen todos los módulos que tiene el firmware, sensores, algoritmos, que se utilizan. Todos los resultados y las opciones de comportamiento se definen con elementos XML estándar.

El archivo de configuración con extensión .XML del fuselaje o estructura se encuentra en `conf/airframes/<yourairframe>.xml` y siempre empieza con un `<!DOCTYPE airframe SYSTEM "airframe.dtd">`, debería tener este aspecto.

A screenshot of a code editor showing the beginning of an XML file. The title bar at the top reads "File: conf/airframes/<yourairframe>.xml". The code content is as follows:

```
<!DOCTYPE airframe SYSTEM "airframe.dtd">
<airframe name="yourairframe">
  <!-- The airframe configuration goes here. -->
</airframe>
```

Figura 13: Archivo de configuración básico que debe tener todo fuselaje.

(paparazziuav, 2012)

3.2.1. CREACIÓN DE UN NUEVO AVIÓN

El archivo de fuselaje es donde se configura la mayoría de los aspectos de su avión, una aeronave totalmente especificada necesita varios archivos de configuración XML:

- Fuselaje.
- Plan de Vuelo.
- Ajustes.
- Radio (si se utiliza un sistema de R/C basado PPM).
- Telemetría.

Cada avión se le asigna un nombre, único ID y los archivos de configuración asociados están en `conf / conf.xml`. Para crear un nuevo Avión, se da clic en el menú

de A/C en el Centro de Paparazzi y se escoge archivo nuevo de fuselaje. O copiando uno ya existente en conf /conf.xml. (paparazziuav.org, 2014)

3.2.1.1. DEFINICIONES DE FIRMWARE Y HARDWARE.

En primer lugar se debe especificar el firmware que se desea utilizar, es decir, si tiene un aeroplano (fixedwing), o un helicóptero (rotorcraft). Una de estas dos opciones debe tener el archivo.

```
File: conf/airframes/myplane.xml
<firmware name="fixedwing or rotorcraft">
  ...
</firmware>
```

Figura 14: Detalle del firmware, esta son las líneas de código donde están especificadas, el tipo de volador a utilizar sea avión o helicóptero. (paparazziuav, 2014)

3.2.1.2. DEFINICIÓN DE HARDWARE.

Para especificar el hardware del piloto automático que se está utilizando y su configuración de bajo nivel se tiene que añadir el objetivo (target-tag), cada objetivo (target), tiene dos atributos que son nombre y un atributo junta (board). El atributo nombre debe ser "ap" (piloto automático) o "SIM / JSBSim / nps" (simulación).

```
File: conf/airframes/myplane.xml
<firmware name="fixedwing or rotorcraft">
  <target name="sim" board="pc"/> <!-- For simulation. -->
  <target name="ap" board="lisa_m_1.0"/> <!-- Select your board here -->
  ...
</firmware>
```

Figura 15: Líneas de definiciones hardware. Estos son los comandos que deben ir para especificar el Hardware (paparazziuav, 2014).

3.3. SUBSISTEMAS.

Cada piloto automático dispone de ciertos subsistemas que deben ser correctamente configurados. Los más importantes se describen a continuación.

3.3.1. IMU.

Una unidad de medición inercial o IMU (del inglés inertial measurement unit), es un dispositivo electrónico que mide e informa acerca de la velocidad.

```
File: conf/airframes/myplane.xml
<firmware name="fixedwing or rotorcraft">
  ...
  <subsystem name="imu" type="aspirin_v1.5"/>
</firmware>
```

Figura 16: Declaración del IMU. Líneas para añadir el subsistema IMU con el tipo que se va utilizar (paparazziuav, 2012).

3.3.2. AHRS

El subsistema AHRS, sistemas de referencia de actitud y rumbo especifica qué estimación de la orientación del filtro se está utilizando, por ejemplo, para el filtro complementario este sería el código.

```
File: conf/airframes/myplane.xml
<firmware name="fixedwing or rotorcraft">
  ...
  <subsystem name="ahrs" type="int_cmpl_euler"/>
</firmware>
```

Figura 17: Subsistema AHRS depende de un subsistema de IMU, excepto por el ahrs_infrared que depende del módulo infrarrojo. Si el magnetómetro se debe utilizar la sección de campo magnético local debe rellenarse (paparazziuav, 2012).

3.3.3. RADIO CONTROL.

En cuanto al radio control los tipos soportados son: ppm, spektrum y de enlace de datos

```
File: conf/airframes/myplane.xml

<firmware name="fixedwing or rotorcraft">
  ...
  <subsystem name="radio_control" type="ppm"/>
</firmware>
```

Figura 18: Declaración de Radio Control se debe especificar el adecuado al subsistema de control de radio en su, sección de firmware (paparazziuav, 2012).

3.3.4. TELEMETRÍA (Modem).

El protocolo del módem y la velocidad de transmisión se establecen tanto en el archivo del fuselaje y de la estación terrestre. La velocidad de transmisión estándar que viene detallada es la de 9600 siendo adecuada y recomendada 57.600 para la mayoría de los usuarios, para permitir la telemetría de alta velocidad durante el análisis de datos de vuelo. La velocidad de datos real se determina por el número de mensajes que se envían y el período de cada mensaje se lo define en el archivo de telemetría .

El subsistema de telemetría admite los siguientes protocolos de módem:

Serie transparente estándar (pprz) - esto es compatible con todos los módems y se puede utilizar para conectar el piloto automático directamente a un PC para la prueba sin un módem.

Protocolo Maxstream API (XBee) - compatible con todos los módems Maxstream incluyendo la 9XTend y Zigbee. Este protocolo permite abordar hardware, lo que permite gestionar múltiples aeronaves desde un único módem.

```
File: conf/airframes/myplane.xml

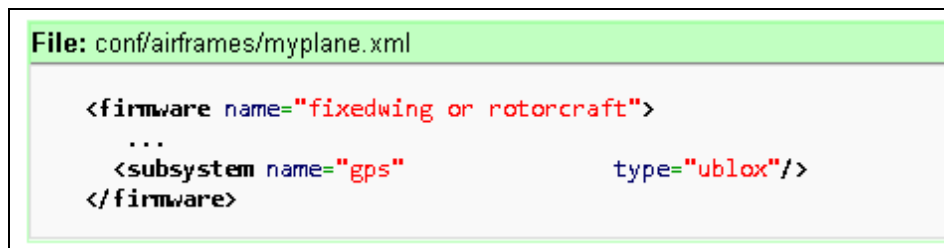
<firmware name="fixedwing or rotorcraft">
  ...
  <subsystem name="telemetry" type="transparent"/>
</firmware>
```

Figura 19: Telemetría de serie transparente. Es compatible con todos los módems y se puede utilizar para conectar el piloto automático directamente a un PC (paparazziuav, 2012).

3.3.5. GPS.

En la configuración del puerto serie este debe coincidir con la del GPS y se configura en esta parte, junto con los archivos necesarios para interpretar el dispositivo u-blox UBX protocolo binario, donde tenemos que tomar en cuentas dos opciones.

- Sólo hay que especificar el adecuado subsistema GPS en su sección de firmware.
- Se debe elegir entre dos tipos de Gps que son el uBlox y ublox_utm para las series más antiguas con 4 módulos que siguen ofreciendo un mensaje UTM.

A screenshot of a text editor window showing an XML configuration file. The title bar reads "File: conf/airframes/myplane.xml". The XML content is as follows:

```
<firmware name="fixedwing or rotorcraft">
...
  <subsystem name="gps" type="ublox"/>
</firmware>
```

Figura 20: Declaración del subsistema GPS tipo ublox, ya que no es posible su comunicación directamente (paparazziuav, 2012).

Los módulos GPS u-blox son configurados con 9.600 baudios de fábrica, se recomienda 38.400 baudios, junto con los otros cambios necesarios. Al Gps se puede acceder directamente a través del túnel del UART y realizar su configuración con el software u-center.

3.3.6. PARÁMETROS XML.

También llamada configuración XML Describe los ajustes disponibles y válidos para varios parámetros. Estos son los comandos en resumen para controlar la aeronave.

```
<commands>
  <axis name="THROTTLE" failsafe_value="0"/>
  <axis name="ROLL" failsafe_value="0"/>
  <axis name="PITCH" failsafe_value="0"/>
</commands>
```

Para helicóptero, por lo general es:

```
<commands>
  <axis name="PITCH"  failsafe_value="0"/>
  <axis name="ROLL"   failsafe_value="0"/>
  <axis name="YAW"    failsafe_value="0"/>
  <axis name="THRUST" failsafe_value="0"/>
</commands>
```

Cada comando también se asocia con un valor a prueba de fallos, si no hay controlador activo, durante la inicialización de la tarjeta del piloto automático. El toma rango de estos valores [-9600: 9600]. Para "MARIPOSA", un rango de [0, 9600] y en la definición del servo correspondiente al neutro y min suelen ser los mismos para los servos PWM.

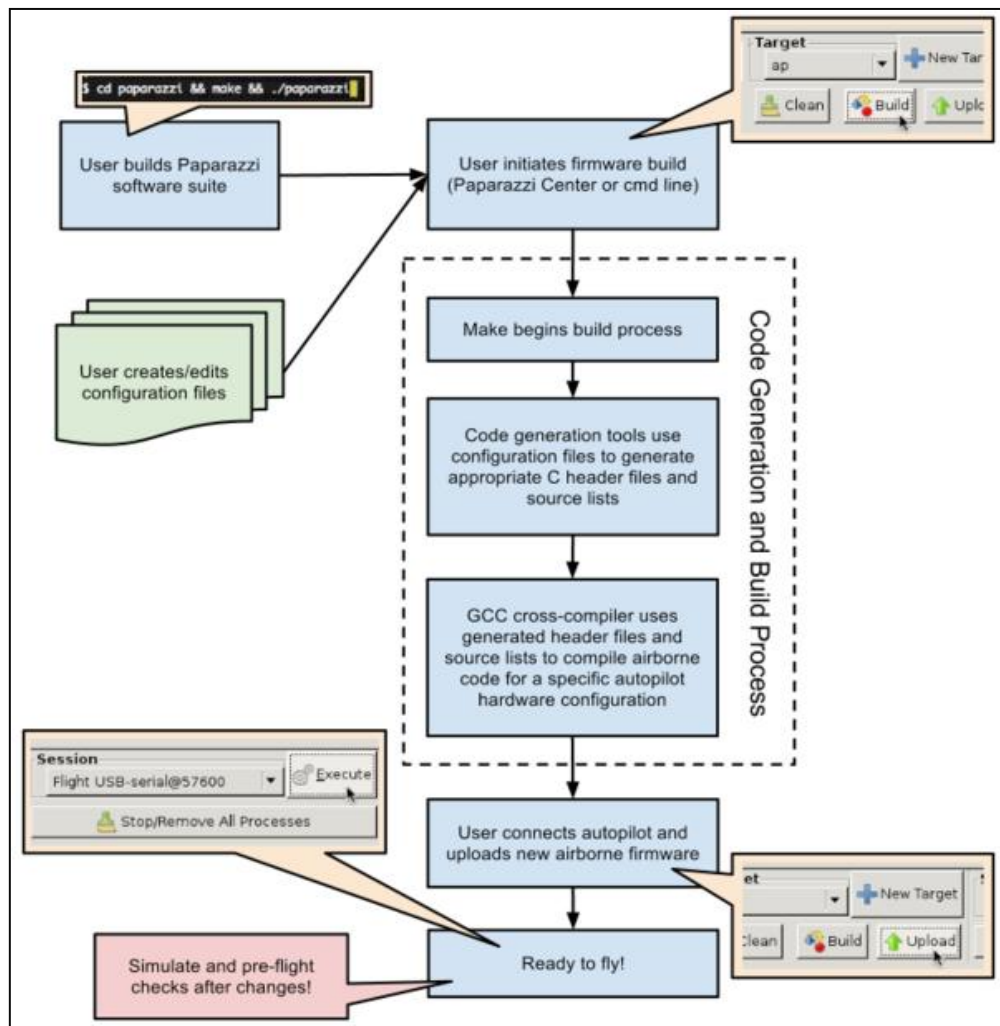


Figura 21. Proceso de Construcción (Build) del Software Paparazzi Airborne. Tomado de (paparazziuav, 2012).

3.3.7. ARQUITECTURA DEL SISTEMA.

La siguiente figura muestra los principales agentes (procesos o programas), del sistema: una (o varias) de las aeronaves y su arquitectura de tierra distribuida.

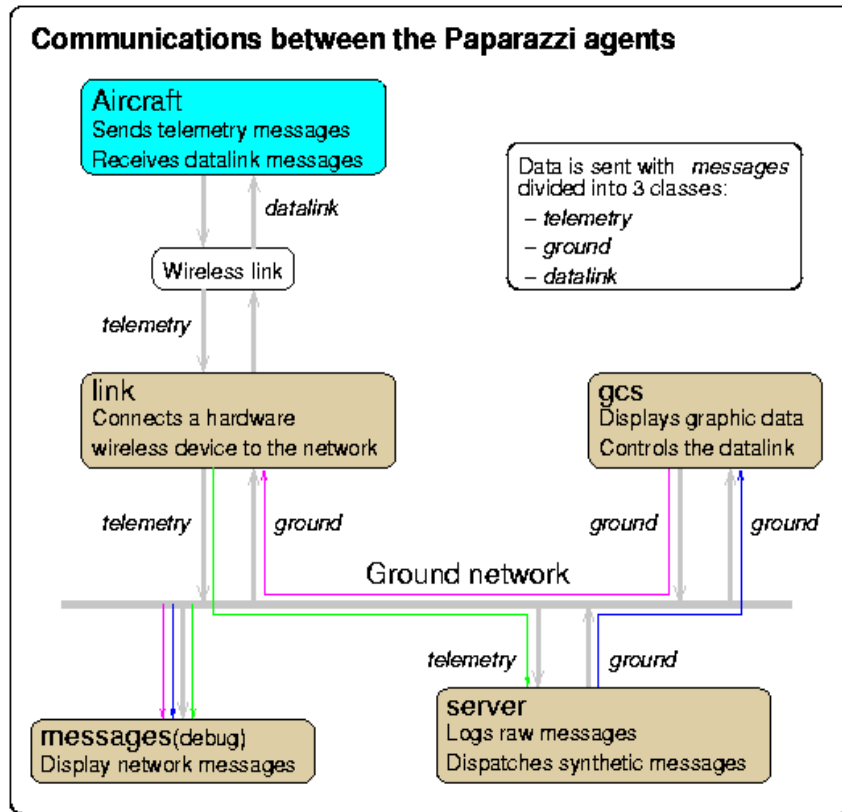


Figura 22. Comunicación entre el paparazzi UAV {en azul} está navegando de forma autónoma y es monitoreado y controlado desde el suelo {en marrón}. (paparazziuav, 2012).

La estación de control en tierra (GCS), o agente de GCS, proporciona una interfaz gráfica de usuario con los datos de telemetría recibidas por el agente de enlace que gestiona el módem de radio en tierra. El agente de enlace distribuye datos de telemetría a través de la red (un solo equipo, una red local o de Internet) donde se puede utilizar de forma local o remota.

Todos estos procesos se ejecutan simultáneamente y cada módulo se lanza de forma independiente donde encontramos los siguientes datos.

Mensajes: Una pantalla numérica en tiempo real de todos los datos de telemetría.

Un número de otros agentes útiles, incluyendo al mismo agente:

Un editor de plan de vuelo en base al GCS, para modificar puntos de interés.

Un UAV simulador para probar los planes de vuelo y las modificaciones de código.

Un plotter en tiempo real para la visualización de datos de telemetría gráfica.

Un plotter de registro para la visualización gráfica de telemetría después de un vuelo.

3.4. CONFIGURACIÓN ROTORCRAFT.

En esta sección se describe las opciones de configuración específicas para el firmware de helicópteros en el archivo del fuselaje.

3.4.1. DEFINICIONES DE FIRMWARE Y HARDWARE.

Este es un ejemplo de una definición bastante estándar firmware Quadcopter:

File: conf/airframes/myrotorcrafft.xml

```
<firmware name="rotorcrafft">
  <target name="nps" board="pc">
    <subsystem name="fdm" type="jsbsim"/>
  </target>
  <target name="ap" board="lisa_m_1.0"/>
  <subsystem name="radio_control" type="ppm"/>
  <subsystem name="telemetry" type="transparent"/>
  <subsystem name="actuators" type="mkk"/>
  <subsystem name="imu" type="aspirin_v1.5"/>
  <subsystem name="gps" type="ublox"/>
  <subsystem name="ahrs" type="int_cmpl_quat"/>
  <subsystem name="stabilization" type="int_quat"/>
</firmware>
```

3.4.2. PROGRAMACIÓN DE SELECCIÓN DE BOARD.

File: conf/airframes/myrotorcrafft.xml

```
<firmware name="rotorcrafft">
```

```

<target name = "sim" tablero = "pc" /> <! - Para la simulación ->
<target name = "ap" tablero = "lisa_m_2.0" /> <! - Selecciona tu tablero aquí ->
...
</firmware>

```

3.4.3. PROGRAMACIÓN DE ACTUADORES.

File: conf/airframes/myrotorcraft.xml

```

<firmware name="rotorcraft">
<target name="ap" board="booz_1.0"/>
...
<subsystem name="actuators" type="mkk"/><! – tipo de actuadores ->
</firmware>

```

3.4.4. PROGRAMACIÓN INS.

El INS (Sistema Integrado de Navegación) contiene un filtro y las estimaciones para el GPS. Estos son usados para una mejor estimación de posición y velocidad. En este caso el subsistema INS es opcional y esta descrito de la siguiente forma.

```

<firmware name="rotorcraft">
<target name="ap" board="booz_1.0"/>
...
<subsystem name="ins" type="hff">
  <define name="GPS_LAG=0.2"/>
</subsystem>
</firmware>

```

3.4.5. ARMADO MOTOR.

Por defecto, los motores están armados con cero aceleradores. Los motores no se inician si AHRS no está alineada.

Otras secuencias de armado se pueden configurar de la siguiente manera.

USE_KILL_SWITCH_FOR_MOTOR_ARMING definida (1):

- Kill Switch apagará para armar a los motores.

- Si kill switch está apagado durante el arranque, es necesario volver a matar primero, luego Unkill para empezar.
- Acelerador tiene que estar abajo, otros palos centrados para arrancar motores.
- Necesita estar en modo manual para iniciar los motores.

USE_THROTTLE_FOR_MOTOR_ARMING definida (1):

- iniciará automáticamente motores al aplicar acelerador.
- si acelerador no estaba abajo en el inicio, es necesario poner el acelerador de nuevo primero.
- otros palos tienen que estar centrado para arrancar motores.
- necesita estar en modo manual para iniciar los motores.

3.4.6. MODOS DEL PILOTO AUTOMÁTICO.

Para los rotorcrafts tenemos diferentes modos que se asignan a su conmutador de 3 posiciones (Manual, Auto 1, Auto 2). El modo horizontal y vertical se hace una selección de forma diferente que la de los modos posibles indican (Son las siglas que aparecen en la GCS).

AP_MODE_FAILSAFE (SAFE).

Este es un modo a prueba de fallo que se inicia si: la señal RC se pierde, o GPS y RC se pierde en el modo NAV.

El comportamiento estándar es que el piloto automático se nivela, si el helicóptero esta fuera (puntos de ajuste de inclinación y balanceo ángulos cero) y descender a 0,5 m/s hacia abajo. Pero este comportamiento también se cambia a otra cosa por supuesto.

AP_MODE_KILL (matar).

Los motores están simplemente desconectados.

AP_MODE_RATE_DIRECT (TASA).

Este es básicamente el modo "más" manual que se puede obtener. No controla la actitud (ángulos de balanceo y cabeceo), pero si la tasa de rotación. También puede poner el acelerador directamente con su RC.

AP_MODE_ATTITUDE_DIRECT (ATT).

Se controla la actitud (balance, cabeceo y guiñada ángulos), pero la aceleración es directamente proporcional a su posición del stick.

AP_MODE_RATE_RC_CLIMB (R_RCC).

Se controla la velocidad de rotación y la velocidad vertical de acuerdo a su posición de mando del acelerador.

Si usted tiene su stick de gas en la posición media, la altitud se mantiene, hasta que baja a una velocidad proporcional a su posición del stick (lo mismo para arriba).

AP_MODE_ATTITUDE_RC_CLIMB (A_RCC).

Se controla la actitud (ángulos de balanceo, cabeceo y guiñada) y la velocidad vertical de acuerdo a su posición de mando del acelerador.

Si usted tiene su stick de gas en la posición media de la altitud se mantiene, si mueve el stick hacia abajo que desciende a una velocidad proporcional a su posición del stick (lo mismo para arriba).

AP_MODE_ATTITUDE_CLIMB (ATT_C).

Controla la actitud (ángulos de balanceo, cabeceo y guiñada) y la velocidad vertical. La velocidad vertical se ajusta a través fms (por ejemplo, la palanca de mando).

AP_MODE_RATE_Z_HOLD (R_ZH).

Controla la velocidad de rotación y mantiene la altitud que estaba en cuando entrar en este modo.

Su posición del stick del acelerador todavía limita la autoridad del acelerador, máximo a menos de lesionados con NO_RC_THRUST_LIMIT.

AP_MODE_ATTITUDE_Z_HOLD (A_ZH).

Controla la actitud (ángulos de balanceo, cabeceo y guiñada) y mantiene la altitud que estabas en cuando entrar en este modo.

Su posición del stick del acelerador todavía limita la autoridad del acelerador máximo a menos de lesionados con NO_RC_THRUST_LIMIT.

AP_MODE_HOVER_DIRECT (flotar).

El helicóptero pasa a la posición horizontal que estaba en cuando entrar al modo de (control de posición). Se debe poner el acelerador directamente con su RC. La orden de guiñada en el RC permite el cambio de rumbo. Si USE_SPEED_REF = 1, entonces las órdenes de cabeceo y balanceo de la velocidad de control RC están de acuerdo a la velocidad máxima fijada con REF_MAX_SPEED en la sección GUIDANCE_H de archivo fuselaje.

AP_MODE_HOVER_CLIMB (HOV_C).

El helicóptero se pasa a la posición que estaba en el momento que entran al modo de (control de posición). Los comandos del RC funcionan igual que en el modo de vuelo estacionario. La velocidad vertical se ajusta a través fms (por ejemplo, la palanca de mando).

AP_MODE_HOVER_Z_HOLD (H_ZH).

El helicóptero se pasa a la posición 3D, en el momento que entran al modo de (control de posición y altitud). Los comandos del RC funcionan igual que en el modo de vuelo estacionario.

La posición del stick del acelerador todavía limita la autoridad del acelerador máximo a menos de lesionados con NO_RC_THRUST_LIMIT.

AP_MODE_NAV (NAV).

Modo de navegación completa. El helicóptero sigue su plan de vuelo. Si se tiene una señal RC válida, la posición del mando del gas todavía limita la autoridad del acelerador máximo a menos de lesionados con NO_RC_THRUST_LIMIT.

AP_MODE_RC_DIRECT (RC_D).

Piloto de seguridad comandos directos para helicóptero.

AP_MODE_CARE_FREE_DIRECT (CF).

Igual que AP_MODE_ATTITUDE_DIRECT, pero los comandos de cabeceo se basan en el ángulo de guiñada al entrar en este modo. (Paparazziuav.org, 2014)

3.4.7. PROGRAMACIÓN PARÁMETROS XML

En este segmento se visualiza de modo general los parámetros principales que se deben ir incluidos dentro del mismo.

File: conf/airframes/myrotorcraft.xml

```
<section name="MODE" prefix="MODE_">
  <define name="MANUAL" value="AP_MODE_ATTITUDE_DIRECT" />
  <define name="AUTO1" value="AP_MODE_ATTITUDE_Z_HOLD" />
  <define name="AUTO2" value="AP_MODE_NAV" />
</section>
```

File: conf/airframes/myrotorcraft.xml

```
<commands>
  <axis name="PITCH" failsafe_value="0"/>
  <axis name="ROLL" failsafe_value="0"/>
  <axis name="YAW" failsafe_value="0"/>
```

```
<axis name="THRUST" failsafe_value="0"/>
</commands>
```

3.4.8. MEZCLA MOTOR.

En esta sección se describe la "mezcla" que se utiliza para la configuración multirotor.

Este subsistema tiene balance, cabeceo, guiñada, los comandos de empuje como entradas y los "cruces" que reciben las órdenes finales para sus motores individuales de acuerdo a la disposición de ellos.

Después de mezclar las órdenes motoras separadas puede "saturar", lo que significa que no se puede simplemente dar empuje negativo o mayor que el máximo. Si una saturación se genera (comando de motor deseada fuera del rango posible MIN_MOTOR / MAX_MOTOR), una compensación de saturación se aplica a todos los motores con el fin de dar actitud, manda una prioridad mayor que el empuje. MOTOR_MIXING_MAX_SATURATION_OFFSET (por defecto es 10% de mando máximo (Paparazziuav.org ,2014)

File: conf/airframes/myrotorcraft.xml

```
<firmware name="rotorcraft">
...
<subsystem name="motor_mixing"/>
...
</firmware>
```

File: conf/airframes/myrotorcraft.xml

```
<firmware name="rotorcraft">
<section name="MIXING" prefix="MOTOR_MIXING_">
  <define name="TRIM_ROLL" value="0"/>
  <define name="TRIM_PITCH" value="0"/>
  <define name="TRIM_YAW" value="0"/>
```

```

<definename="MAX_SATURATION_OFFSET value="MAX_PPRZ/10"/>
<define name="NB_MOTOR" value="4"/>
<define name="SCALE" value="256"/>
<define name="ROLL_COEF" value="{ 0 , 0, 256, -256 }"/>
<define name="PITCH_COEF" value="{ 256, -256, 0, 0 }"/>
<define name="YAW_COEF" value="{ -256, -256, 256, 256 }"/>
<define name="THRUST_COEF" value="{ 256, 256, 256, 256 }"/>
</section>
</firmware>

```

3.4.8.1 EJEMPLOS MEZCLA.

Sugerencia: Si sus rotores están girando opuesta a la dirección que se muestra en la imagen, revertir los signos en la línea YAW_COEF.

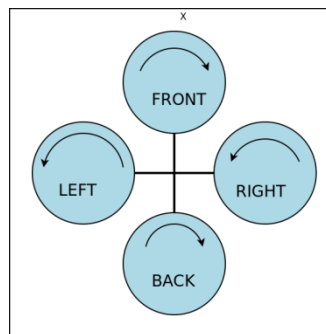


Figura 23. Plus Cruz. Suponiendo que el orden de los motores, que se describe en la sección "servos" es parte delantera, trasera, izquierda, derecha (paparazziuav, 2014).

```

<Define name = valor "ROLL_COEF" = "{0, 0, 256, -256}" />
<Define name = valor "PITCH_COEF" = "{256, -256, 0, 0}" />
<Define name = valor "YAW_COEF" = "{-256, -256, 256, 256}" />
<Define name = valor "THRUST_COEF" = "{256, 256, 256, 256}" />

```

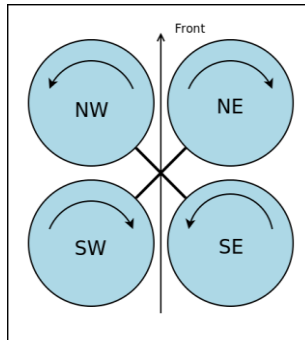


Figura 24. Tiempo de la Cruz Suponiendo que el orden de los motores, que se describe en la sección "servos" es NE, SE, SW, NW. (paparazziuav, 2014).

Suponiendo que el orden de los motores, que se describe en la sección "servos" es NE, SE, SW, NW.

```
<Define name = valor "ROLL_COEF" = "{-256, -256, 256, 256}" />
<Define name = valor "PITCH_COEF" = "{256, -256, -256, 256}" />
<Define name = valor "YAW_COEF" = "{-256, 256, -256, 256}" />
<Define name = valor "THRUST_COEF" = "{256, 256, 256, 256}" />
```

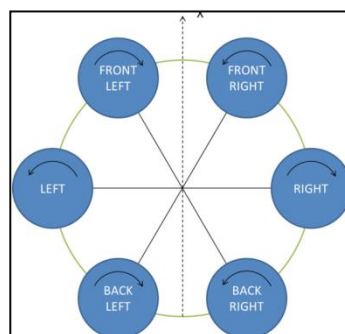


Figura 25. Hex (Tiempo de la Cruz) Suponiendo que el orden de los motores, que se describe en la sección "servos" es FRONT_LEFT, FRONT_RIGHT, DERECHA, BACK_RIGHT, BACK_LEFT, Izquierda (paparazziuav, 2014).

Suponiendo que el orden de los motores, que se describe en la sección "servos" es FRONT_LEFT, FRONT_RIGHT, DERECHA, BACK_RIGHT, BACK_LEFT, Izquierda.

```
<Define name = valor "ROLL_COEF" = "{128, -128, -256, -128, 128, 256}" />
<Define name = valor "PITCH_COEF" = "{256, 256, 0, -256, -256, 0}" />
<Define name = valor "YAW_COEF" = "{-256, 256, -256, 256, -256, 256}" />
<Define name = valor "THRUST_COEF" = "{256, 256, 256, 256, 256, 256}" />
```

```

<Define name = valor "ROLL_COEF" = "{106,-106,-256,-256,-106,106, 256, 256}"
/>
<Define name = valor "PITCH_COEF" = "{256,256, 106,-106,-256,-256,-106,
106}" />
<Define name = valor "YAW_COEF" = "{-256,256, -256,256, -256, 256, -256,
256}" />
<Define name = valor "THRUST_COEF" = "{256, 256,256, 256,256, 256,256,
256}" />

```

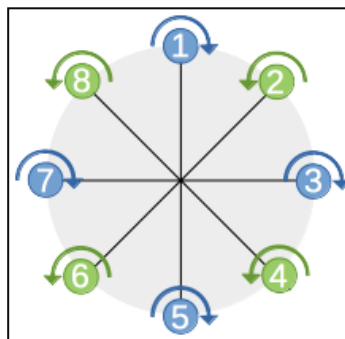


Figura 26. Octo (además de cruz) (paparazziuav, 2014).

```

<Define name = valor "ROLL_COEF" = "{0, -181, -256, -181, 0, 181, 256, 181}" />
<Define name = valor "PITCH_COEF" = "{256, 181, 0,-181, -256,-181, 0, 181}" />
<Define name = valor "YAW_COEF" = "{-256, 256, -256,256, -256,256, -
256,256}" />
<Define name = valor "THRUST_COEF" = "{256,256, 256,256, 256,256, 256,
256}" />

```

Dirección.

Hay dos conjuntos de parámetros de orientación: vertical (altura) y horizontal (posición).

File: conf/airframes/myrotorcrafft.xml

```

<section name="GUIDANCE_V" prefix="GUIDANCE_V_">
<define name="HOVER_KP" value="150"/>
<define name="HOVER_KD" value="80"/>

```



```
<define name="HOVER_KI" value="20"/>

<!-- optional parameters -->
<define name="NOMINAL_HOVER_THROTTLE" value="0.5"/>
<define name="ADAPT_THROTTLE_ENABLED" value="TRUE"/>
<define name="REF_MIN_ZD" value="-3.0" unit="m/s"/>
<define name="REF_MAX_ZD" value="3.0" unit="m/s"/>
<define name="MAX_RC_CLIMB_SPEED" value="-3.0" unit="m/s"/>
<define name="MAX_RC_DESCENT_SPEED" value="3.0" unit="m/s"/>
</section>
```

HOVER_K x.

Parámetros PID para bucle de control de vuelo estacionario verticales.

NOMINAL_HOVER_THROTTLE.

Porcentaje de aceleración esperada necesaria para estacionario (por defecto es 0,4 = 40%)

ADAPT_THROTTLE_ENABLED.

Permitir nominal estimación del acelerador libración adaptativo (por defecto es TRUE, se establece en FALSE para desactivar).

REF_MIN_ZD.

Límite inferior de referencia de velocidad vertical (desde z-down es positivo -> velocidad máxima hacia arriba) (-3.0m default / s).

REF_MAX_ZD.

Límite superior de referencia de velocidad vertical (desde z-down es positivo -> velocidad máxima hacia abajo) (3.0m default / s).

MAX_RC_CLIMB_SPEED.

Subir la velocidad en la entrada max RC en modo RC_CLIMB (por defecto es REF_MIN_ZD).

MAX_RC_DESCENT_SPEED.

Velocidad de descenso en la entrada max RC en modo RC_CLIMB (por defecto es REF_MAX_ZD).

Orientación horizontal.

File: conf/airframes/myrotorcrafft.xml

```
<section name="GUIDANCE_H" prefix="GUIDANCE_H_">
  <define name="PGAIN" value="50"/>
  <define name="DGAIN" value="100"/>
  <define name="IGAIN" value="20"/>
  <!-- optional parameters -->
  <define name="AGAIN" value="0"/>
  <define name="VGAIN" value="0"/>
  <define name="MAX_BANK" value="20" unit="deg"/>
  <define name="USE_SPEED_REF" value="TRUE"/>
  <define name="REF_MAX_SPEED" value="5.0" unit="m/s"/>
  <define name="REF_MAX_ACCEL" value="5.66" unit="m/s2"/>
  <define name="REF_OMEGA" value="67" unit="deg"/>
  <define name="REF_ZETA" value="0.85"/>
  <define name="REF_TAU" value="0.5"/>
  <define name="APPROX_FORCE_BY_THRUST" value="FALSE"/>
</section>
```

[PID] GAIN.

Ganancias de PID para el control horizontal

OTRA VEZ.

Ganancia de feedforward de aceleración (por defecto 0).

VGAIN.

Ganancia de feedforward de velocidad (por defecto 0).

MAX_BANK.

Máximo ángulo de balanceo / inclinación que se establece a partir de la orientación horizontal (20deg defecto, máximo 40 grados).

USE_SPEED_REF.

Desde v5.1, dar consignas a través de RC en GUIDANCE_H_MODE_HOVER (predeterminado: TRUE).

REF_MAX_SPEED.

Velocidad horizontal máxima de referencia en m / s (5,0 m por defecto / s).

REF_MAX_ACCEL.

Aceleración máxima horizontal de referencia en m / s² (TANF por defecto (RadOfDeg (30.)) * 9,81 = 5,66).

REF_OMEGA.

Frecuencia natural modelo de segundo orden.

REF_ZETA.

Segundo orden del modelo de amortiguación.

REF_TAU.

Primer orden constante de tiempo.

APPROX_FORCE_BY_THRUST.

Tratar de mejores comandos de fuerza aproximadas tomando impulso en cuenta (por defecto FALSE, se establece en TRUE para usarlo).

3.5. FUSION HARDWARE Y SOFTWARE.

La integración se da con la aeronave (Ar.drone2.0) y el software, se utilizo un receptor GPS preciso, un control de mando (joystick), y un ordenador portátil con Ubuntu 12.4.

Estructura de bloque de nuestra Aeronave.

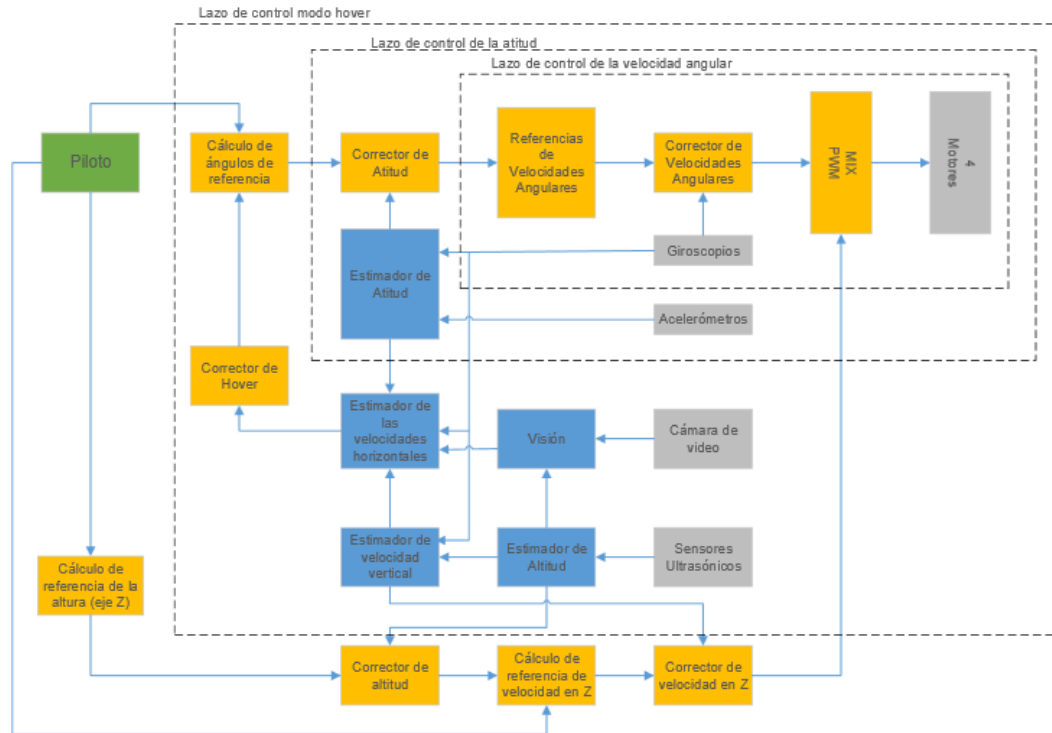


Figura 27. Esquema de estructura del ar.Drone2.0 (bibdigital, 2013).

Actualmente, para ser capaz de volar el AR Drone 2 necesita ARDrone2 v2.4.8 firmware o superior en el Drone AR 2.

Para darle al ARDrone2 la autonomía esencial para el vuelo autónomo, el Centro de Control Paparazzi debe iniciarse abriendo un terminal y las líneas que se debió escribir en la terminal son las siguientes.

1.- maximiliano@maximiliano-Satellite-L455:~\$ cd paparazzi

2.- maximiliano@maximiliano-Satellite-L455:~/paparazzi\$./paparazzi

Al ejecutar Paparazzi Center se pone en marcha y se muestra una pantalla como la detallada:

En la parte superior se tiene la barra de menú A/C, Session, Tools, Wiew Y Help.

En el parte izquierda de la pantalla cuenta con todo el conjunto necesarios para el fuselaje el Airframe, Flight plan, Setting, Radio y Telemetry.

En el parte derecha de la pantalla se tiene el segmento de compilación y ejecución en modo de simulación, y de carga si en modo AP.

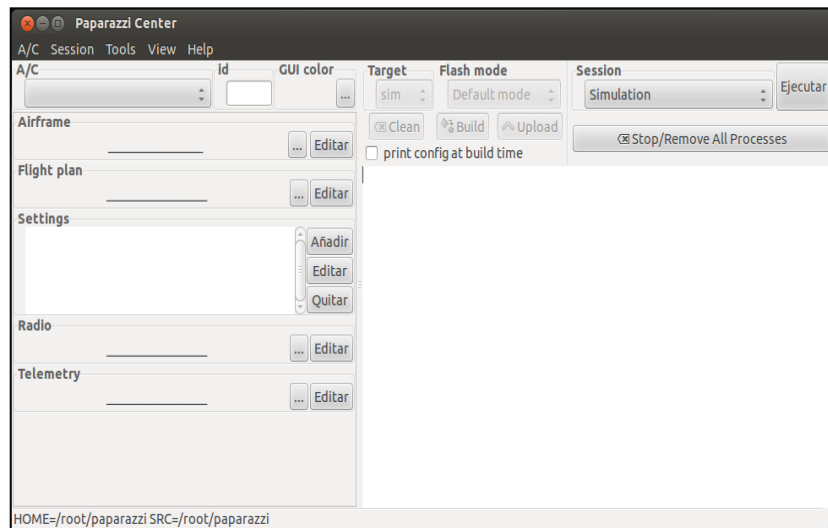


Figura 28. Pantalla Principal Paparazzi Center.

Ahora que tiene Paparazzi Centro corriendo agarre el ARDrone2 configurarlo para el vuelo autónomo:

Inserte el módulo GPS LEA6H UBLOX en su ARDrone2 y ponerlo en marcha mediante la conexión de la batería.

Establecer una conexión WiFi entre el ARDrone2 y su ordenador portátil

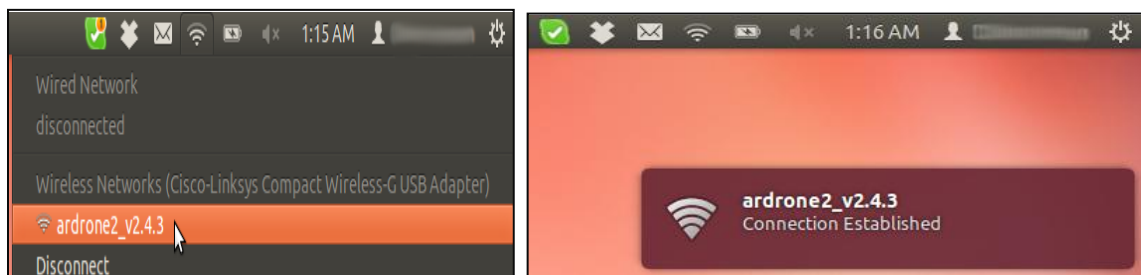


Figura 29. Conexión al WI-FI de ARDrone2. Muestra la señal wi-fi que reconoce el software para conectarse con el drone.

En este momento el ARDrone2 debe estar conectado a un dispositivo GPS atreves del convertidor USB 232R, se ha establecido una conexión Wi-Fi con el ordenador. Ahora puede continuar cargando el piloto automático a la ARDrone2.0.

En Paparazzi Center seleccionar ardrone2_raw en el cuadro desplegable "A / C"

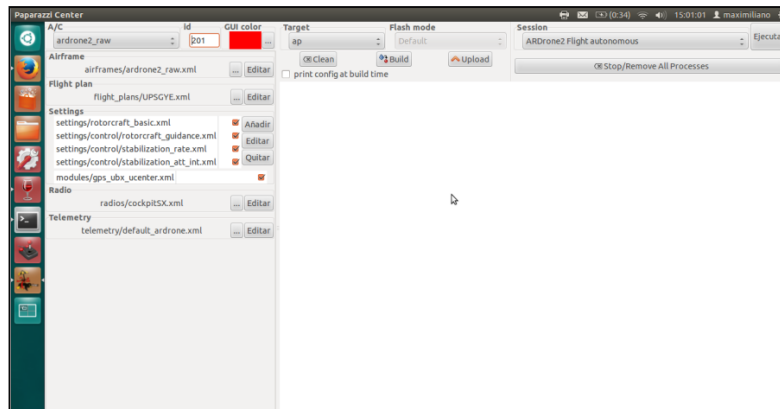


Figura 30. Carga A/C del piloto automático (board).

Estos son los pasos que hay que seguir para poner en funcionamiento del plan de vuelo:

Seleccionar "ap" para la carga del Firmware (piloto automático), se establece como objetivo para el board.

Se presiona (clean), esto lo hace para limpiar cualquier proceso solo por seguridad.

A continuación (Build), esto compila nuestro firmware y verifica si no hay errores en la estructura de construcción del piloto automático.

Pulse el botón (Upload), para cargar el firmware a la aeronave ARDrone2.

Ahora en la parte superior derecha selecciona la sección de ARDrone2 Flight autónomo.

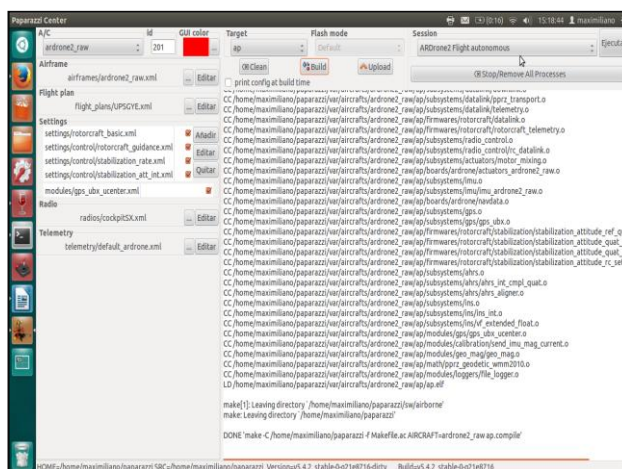


Figura 31. Opciones para escoger ARDrone2 Flight autonomous.

3.5.1 CONVERTIDOR USB Y GPS.

Antes de pasar a los vuelos se explica la conexión entre el GPS y el ARDrone2.0 para este tipo de aeronave no viene con un GPS, es un módulo externo que se debió adaptar a la planta, el ARDrone2.0 viene con una entrada USB. Para el GPS.

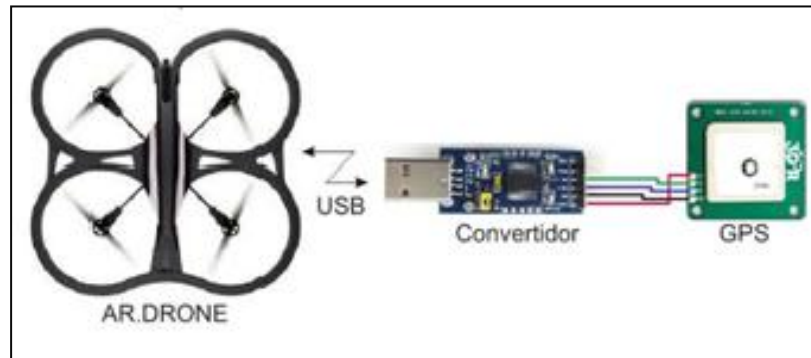


Figura 32. Diagrama de conexión ARDrone2.0, Convertidor UART y GPS.

Para realizar esta comunicación se utilizó un convertidor UART FTDI232R, este hace un puente de comunicación entre el GPS y el ARDrone2.0

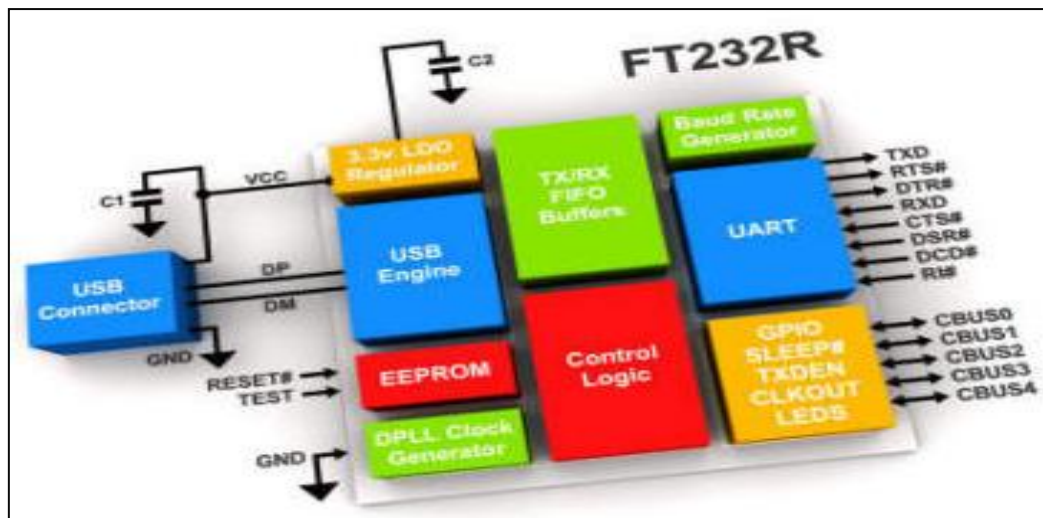


Figura 33. Diagrama de bloque FT232R. (ftdichip, 2015)

3.6 CGS RUTA DE VUELO.

Después de explicar la conexión del GPS, vamos a volar, tomando la aeronave y portátil, hay que tomar en cuenta una visión clara del cielo, es esencial para que el GPS funcione correctamente. Una buena señal del GPS es necesaria para el vuelo

autónomo. Sin embargo GPS es sólo una precisión de varios metros, así que asegúrese de que tiene suficiente espacio libre alrededor para dejarlo volar sin chocar con algo.

Después de esperar durante un minuto la señal GPS ya debe de estar sincronizada, con la ubicación del dron; ARDrone2.0 debe aparecer ahora en el mapa GCS.

Al volar por primera vez en cualquier lugar se necesita descargar el mapa de Google para su computadora portátil. Conecte el Wi-Fi al router inalámbrico, cuando se tiene conexión al internet se da click en el mapa negro GCS. Una vez descargadas las imágenes se guardan para que no se necesite una conexión a Internet la próxima vez.

También puede ejecutar el GCS de antemano Obtenga las imágenes, y luego realizar el vuelo sin una conexión a Internet.

Luego de cargar el mapa reconéctese con la aeronave. Si no hay conexión Wi-Fi desde su ordenador portátil a la ARDrone2, si se tiene una pantalla en blanco. Eso es perfectamente normal. Sólo hay que asegurarse de que haga una conexión Wi-Fi al dron primero.

Para Despegar Pulse el botón [Takeoff] y tener un buen ojo en donde su Dron vuela.

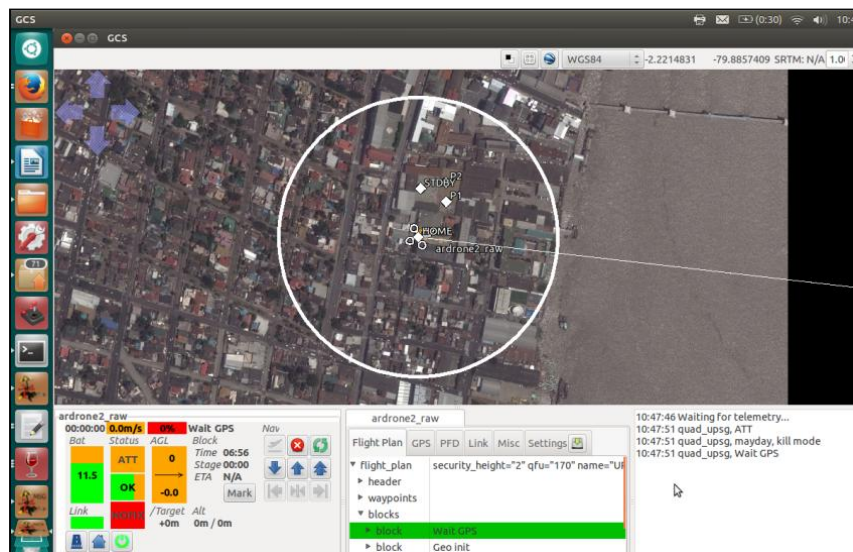


Figura 34. GCS con azulejos cargados. Cuando no tenemos el mapa y la pantalla este negra nos ubicamos en maps buscamos maps fill o combinamos Ctrl+g para la cargar las imágenes.

Home.

Es el punto de inicio para el despegue del ARDrone2.0.

Trayectoria de vuelo.

Son los bloques de trayectoria de vuelo como route, go, línea, círculo.

3.7 PRODUCTO TERMINADO

Para darle el acabo final se tomó otro protector nuevo para las propelas, ya que el anterior se lo uso para pruebas y este término en muy mal estado.



Figura 35. El Ardrone2.0 con nuevo protector. Para proteger las hélices.

En la parte interna del protector se hizo el montaje de los circuitos del GPS y el convertidor Ftdi232r.



Figura 36. Vista interna de la distribución de los módulos electrónicos y montaje de la antena en la parte superior.

3.8. PRUEBAS DE FUNCIONABILIDAD.

- Se enciende el ARdrone2.0 conectado a la batería se observa el encendido de los leds de cada uno de los variadores de velocidad en color rojo.
- Luego de un tiempo los motores se sincronizan M1 suena, 2 segundos mas suena M2, así M3 y por último M4 ahora los leds cambia de color de rojo a verde esto indica que está listo para trabajar.
- Con el computador que tenga S.O. Linux Ubuntu 12.4, ejecutamos un terminal y escribe # cd paparazzi y presiona la tecla enter el siguiente paso. /paparazzi nuevamente enter este ejecutara el paparazzi center.
- En la parte de AC escoge el Airframe que se desee trabajar, compilamos para no tener ningún error o advertencia. Si no hay error tiene que conectarse al WI-FI del drone, se carga el código del Airframe con Upload.
- Con la carga en éxito se selecciona Ejecutar, se abre la pantalla de GCS, en la pantalla central el área de vuelo en la parte inferior los datos de operación, en la parte central lecturas de sensores.
- En la GCS tiene que observar en estatus si el Joystick esta OK, si todo está operativo se deben encender los motores, por seguridad el acelerador debe estar en cero caso contrario los motores no encienden. Con esto ya está en un modo manual.
- Espere que el GPS se alinee con los satélite tiene que quedar 3D, cuando este operativo ya se puede pasar a un modo auto2.
- Con la aceleración al máximo presione takeoff y este comienza con el vuelo a los puntos ya establecido en su plan de vuelo.

Se hizo pruebas de funcionamiento con el controlador de mando en modo manual, y en modo Auto2 con el Paparazzi.



Figura 37. Piloto del dron en modo manual para el encendido, pruebas de vuelo de estabilidad a 5mts de altura en los patios de la UPS sede Guayaquil.

Durante las prueba de vuelo se obtuvo muchos inconveniente a continuación se detallan y su soluciones.

- Uno de los variadores se quemó y se lo tuvo que cambiar por otro, en prácticas de vuelo la planta se estrelló a una altura de 15mt provocando el daño en uno de los engranes de las propelas.
- Durante la visita técnica los motores no se sincronizaba, la solución fue desarmar el dron para revisar los cables y se encontró uno en corto; Se lo aisló y volvió a estar operativo.
- Otro de los problemas fue la perdida de estabilidad, esto se lo resolvió tomado las lectura de los sensores y la data, se la agrego al Airframe se compilo y se cargó, se hicieron pruebas de vuelo quedando todo operativo.
- En una de las pruebas el GPS no dio lectura esto paso por que en los puntos de conexión una de las pistas se levantó y otros pines se aflojaron, se soldó y se puenteo uno de los terminales.

3.8.1 CARATECTRÍSTICAS TÉCNICAS DE OPERACIÓN

El wiki detalla textualmente la siguiente advertencia. Previa a cualquier prueba, tenga en cuenta que el uso del software libre paparazzi no es compatible ni avalado por la empresa Parrot S.A. Utilizando este software podrá anular la garantía o destruir la aeronave. Cuando haya terminado el vuelo autónomo con el paparazzi y quieres volar con las aplicaciones estándar de ARDrone2, sólo tiene que desenchufar y volver a conectar la batería.

http://wiki.paparazziuav.org/wiki/AR_Drone_2/getting_started

<http://ardrone2.parrot.com/support/>

Antes del despegue se verifique primero las condiciones climática, favorables de vuelo, con un aproximado de 3km/h del viento. Mayor a esto no es recomendado volar.

Una vez comprobado que todos los sistemas están operativos se comienza con el arranque de motores se ejecuta Takeoff y comienza con el bloque STDBY.



Figura 38. Prueba en los patios de la UPS sede Guayaquil antes del despegue.

Tiempo de llegada: 3s

Velocidad: 6.5m/s

Aceleración: 58%

Tiempo de espera: 20s

Total de vuelo: 20s

Altura: 6mt.



Figura 39. Prueba en los patios de la UPS sede Guayaquil durante el despegue.

Después de unos 20s se pasara al siguiente bloque stay_P1.

Tiempo de llegada: 3s

Velocidad: 7.1m/s

Aceleración: 72%

Tiempo de espera: 15s

Total de vuelo: 33s

El siguiente bloque será de go_P2 pasado los 15 s, esto es ida y vuelta sin espera en P2.

Tiempo de llegada: 6s

Velocidad: 6.5m/s

Aceleración: 53%

Tiempo de espera: 15s

Total de vuelo: 48s



Figura 40. Prueba en los patios de la UPS sede Guayaquil durante el ruteo de puntos. Bloque de línea en entre P1 y HOME con un tiempo de espera en Home de 10s.

Idea

Tiempo de llegada: 4s

Velocidad: 7.0m/s

Aceleración: 62%

Tiempo de espera: 10s

Total de vuelo: 56s

Vuelta

Tiempo de llegada: 3s

Velocidad: 6.5m/s

Aceleración: 53%

Tiempo de espera: 10s

Total de vuelo: 1.m 10s

Bloque de círculo antes de llegar a P1 la aeronave toma la ruta a círculo alrededor del punto P2 con un radio de 2mt.

Tiempo de llegada: 4s

Velocidad: 7.5m/s

Aceleración: 75%

Tiempo de espera: 15s

Total de vuelo: 1m 12ss



Figura 41. Prueba en los patios de la UPS sede Guayaquil durante el ruteo de antes del descenso.

Bloque de ruteo, este bloque es el mas extenso tomo una ruta de forma cuadrática desde P1 a P2, de P2 A STABY, de STABY a HOME y por ultimo de home a P1.

Tiempo de llegada: 4s

Velocidad: 5.5m/s

Aceleración: 72%

Tiempo de espera: 40s

Total de vuelo: 1m 52s

Bloque de aterrizaje, este es el último bloque, después de realizar el bloque de ruteo la aeronave desde el último punto P1, busca a HOME para descender en este punto.

Tiempo de llegada: 5s

Velocidad: 3.0m/s

Aceleración: 82%

Tiempo de espera: 58s

Total de vuelo: 2m 10s



Figura 42. Prueba en los patios de la UPS sede Guayaquil drone descendiendo.



Figura 43. Prueba en los patios de la UPS sede Guayaquil drone aterrizado.

Por seguridad se validó que el uso del joystick es necesario para tomar el control de la nave en caso de ser necesario.

- Hay una alerta de aviso para la batería en caso de que este con una carga baja.
- Si uno de los puntos esta fuera del radio de acción la nave regresara a HOME.
- Las alertas se dan de forma visual y por audio en inglés.

CAPÍTULO IV.

PRÁCTICAS DE LABORATORIO

4.1 PRÁCTICA # 1.

Tema:

- Monitoreo de sensores presentes en el sistema autopiloto.

Objetivo General:

- Visualizar el estado de cada uno de los sensores disponible en el sistema de vuelo.

Objetivos Específicos:

- Realizar configuración adecuada para GPS.
- Diseñar el código para manejo de la tarjeta controladora.

Materiales:

- Pc (portátil) con sistema operativo Linux Ubuntu
- GPS u-Blox LEA6H
- AR.Drone2.0
- convertidor UART Ftdi232
- software u-center.

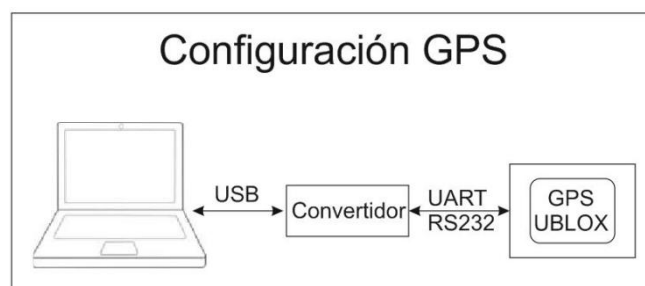


Figura 44. Diagrama de conexión entre la Pc. El convertidor FTDI RS232 y GPS Ublox LEAH6.

Desarrollo:

Para esta práctica lo primero que hay que configurar es el GPS. Usando el software u-center_v8.11. El cual se puede conseguir en <http://www.u-blox.com/en/evaluation-tools-a-software/u-center/u-center.html> .

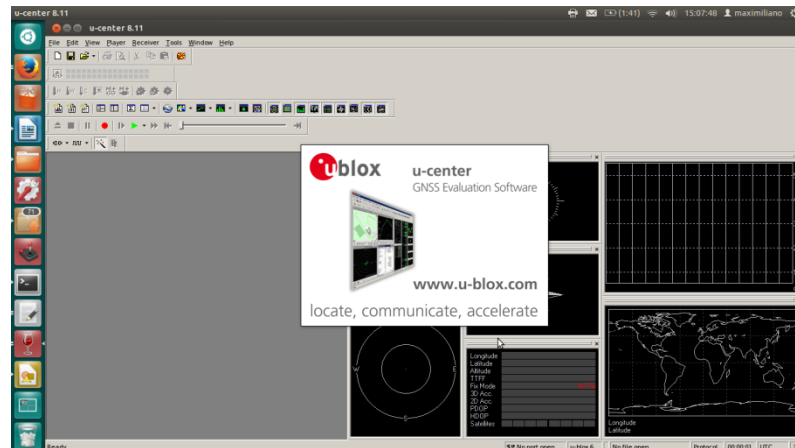


Figura 45. Pantalla principal del software U-center V8.11.

Se cambia los siguientes parámetros en el software:

Se presiona F9 luego haga clic derecho sobre el texto **NMEA** en la parte inferior del árbol y seleccione deshabilitar mensajes para niño (**Disable Child Messages**)

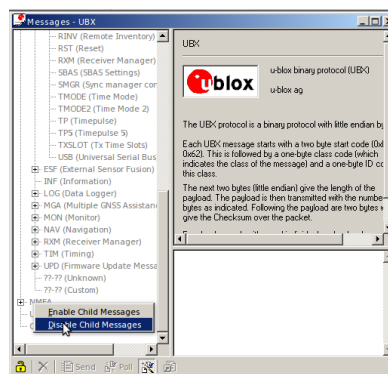


Figura 46. Subrutina de para la configuración del GPS.

Siguiente paso es buscar en el árbol UBX se abre los ramales y se buscara CFG (config) en el siguiente ramal se selecciona NAV5 (Navegación 5) - configurarlo para utilizar **pedestrian** Para una posición fija flotando heli, ajuste peatones.

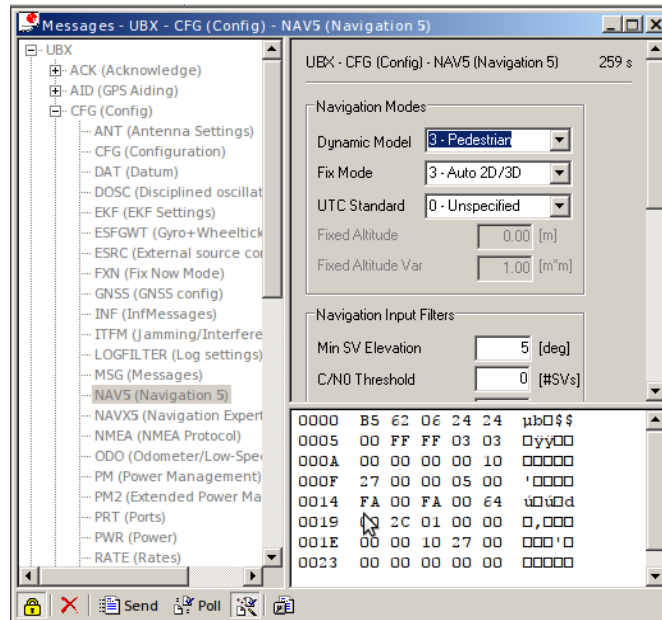


Figura 47. Segmento del ramal UBX – CFG – NAV5.

Se continua en el ramal UBX se busca CFG y se ubica en PRT –se establece **USART1 a 4800 bps** (debe coincidir con el valor en el archivo de la estructura del ardrone2.0)

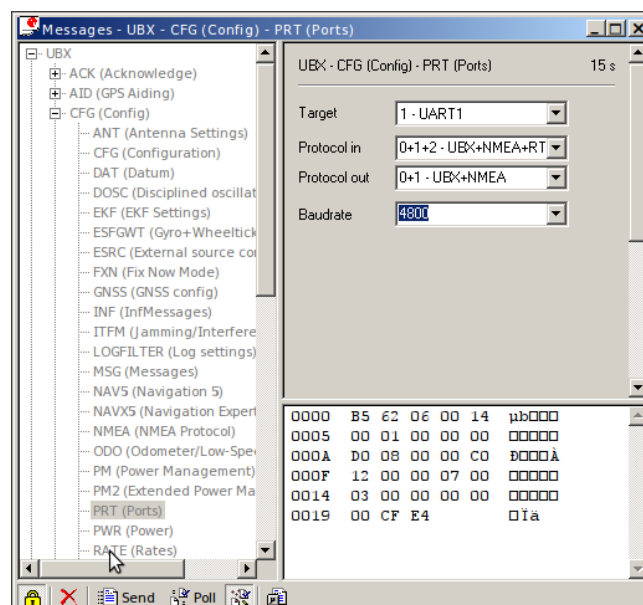


Figura 48. Segmento del ramal PTR se configura el valor de baudrate.

Se cambia la velocidad de transmisión de T-Center a 38400 bps si la conexión se pierde en este rango.

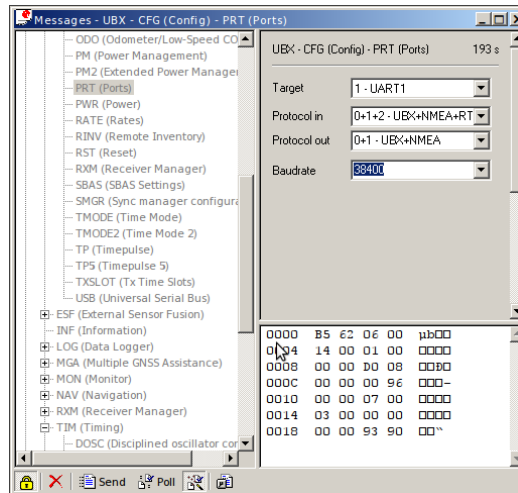


Figura 49. Se configura baudrate a 38400 como rango máximo.

Se mantiene en UBX y en el ramal CFG, se busca la opción de RATE (tarifas) se cambia el **periodo de medición a 250ms**. Esto da una actualización de la posición 4 Hz desde el 4 x 250 ms es un segundo.

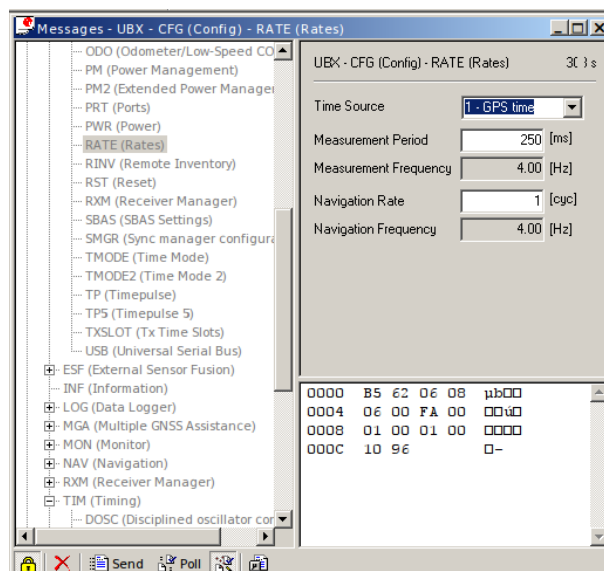


Figura 50. Se cambia el periodo de medición.

Al cambiar el periodo de medición se debe de seguir en el ramal UBX buscar CFG, seleccionar deshabilitar SBAS: **Disable** (SBAS parece causar errores cálculos en altitud severas ocasionales)

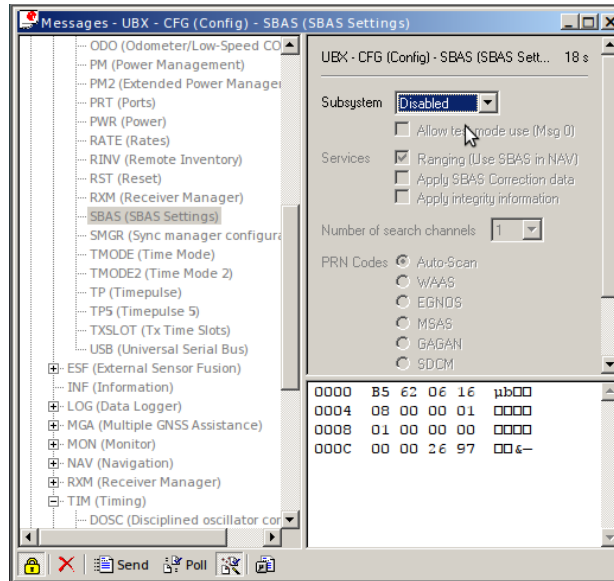


Figura 51. Se deshabilita la sub rutina SBAS.

En UBX y se busca la opción de NAV (no la que está ubicado el segmento de UBX opción CFG y luego NAV): se hace doble clic en **POSLH**, **SOL**, **STATUS**, **SVINFO**, **VELNED** estos cambian de color gris a negro.

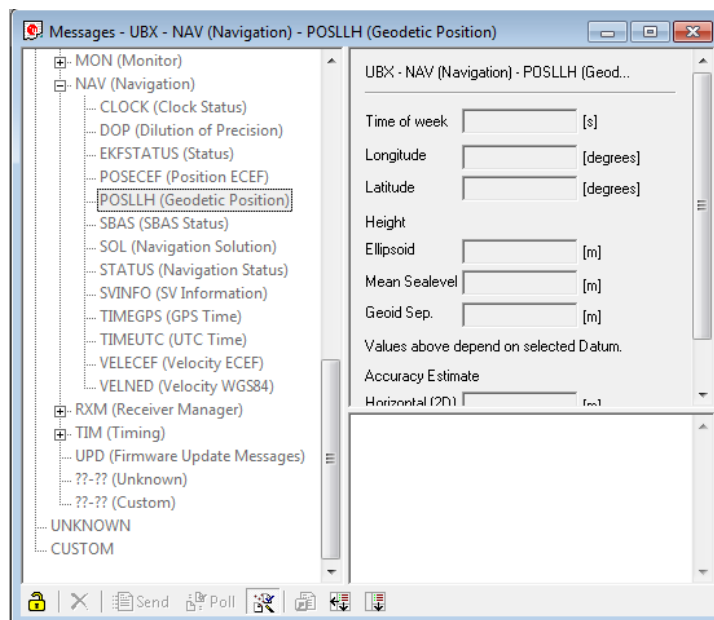


Figura 52. En esta sección se habilitan cinco opciones.

Para guardar todo los cambios hay que volver a UBX se busca el ramal CFG y selecciona la opción CFG: **salvar de configuración actual**, se hace clic en **"send"**

en la esquina inferior izquierda para guardar permanentemente esta configuración para el receptor.

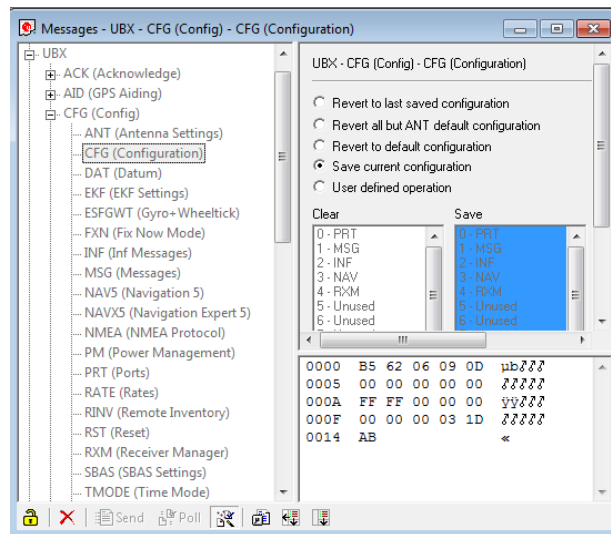


Figura 53. Se guardan todos los cambios que se han realizado al GPS.

Después de configurar el GPS, Se comienza con la primera parte de la programación. Donde esta parte corresponde al segmento del Firmware principal en el cual se detalla los tipos de sensores a utilizar por el ARDrone2.0.

4.1.2. Código de la programación del firmware.

```
<!DOCTYPE airframe SYSTEM "airframe.dtd">
<!-- MultirotorQuadrocoptero sistema autopiloto Universidad Politécnica Salesiana
Sede Guayaquil AR.drone GPS Ublox y WiFi-->
<airframename="ardrone2_raw"><!--nombre de nuestro quadcopter -->
<firmware name="rotorcrafter"><!--tipo de aeronave -->
<target name="ap" board="ardrone2_raw"><!--tipo de placa autopiloto -->
<define name="AUTOPILOT_DISABLE_AHRS_KILL"/>
<subsystemname="telemetry" type="udp"/><!-- sensores de comunicación-->
<subsystemname="radio_control" type="datalink"/><!--Para control de RC y
joystick -->
</target>
<target name="nps" board="pc"><!--tipo de placa para simulación -->
```

```

<subsystemname="fdm" type="jsbsim"/><!--tipo de nave para simulación -->
<subsystemname="radio_control" type="ppm"/><!--tipo señal del RC -->
</target>
<define name="USE_SONAR" value="TRUE"/><!--tipo sensor -->

<!-- Sección de Subsistemas -->
<subsystemname="motor_mixing"/><!--combinación de motores -->
<subsystemname="actuators" type="ardrone2"/><!-- control de los motores-->
<subsystemname="imu" type="ardrone2"/><!--para sensores de imu unidad de
medición inercial-->
<subsystemname="GPS" type="ublox"/><!--para sensores de GPS -->
<subsystemname="stabilization" type="int_quat"/><!--para sensores de
estabilización -->
<subsystemname="ahrs" type="int_cmpl_quat"/><!--para sensores actitud rumbo
cabeceo balanceo -->
<subsystemname="ins" type="extended"/><!--para sistema de nav integrado -->
</firmware>

```

Cerramos el firmware de aquí en adelante solo se trabaja con la configuración de combinación de motores, sistemas de orientación vertical, orientación horizontal, mensajes de alertas.

4.1.3. DIAGRAMA DE CONECTIVIDAD

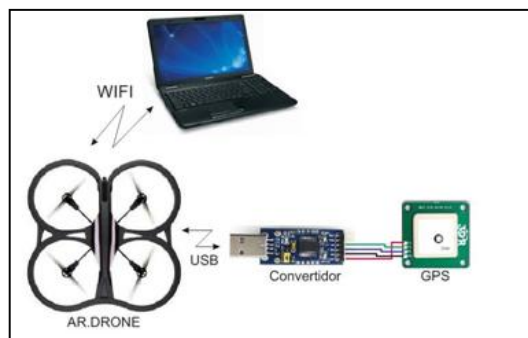


Figura 54. Diagrama de conexión de los módulos a utilizar. Para un vuelo autónomo.

Compilación de programa si no hay errores se obtiene al final el siguiente mensaje para la carga el código al AR.drone,

DONE 'make -C/home/maximiliano/paparazzi -f Makefile.ac AIRCRAFT=ardrone2_raw ap.compile'.

Verificar la conexión con el WiFi ardrone2.

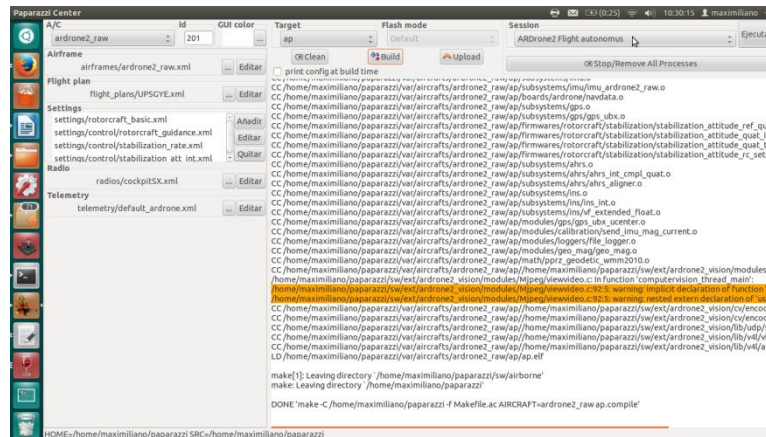


Figura 55. Compilación del Airframe.

Se carga el código seleccionando con el botón Upload. Si la carga es exitosa se obtiene el siguiente mensaje:

DONE 'make -C /home/maximiliano/paparazzi -f Makefile.ac AIRCRAFT=ardrone2_raw ap.upload',

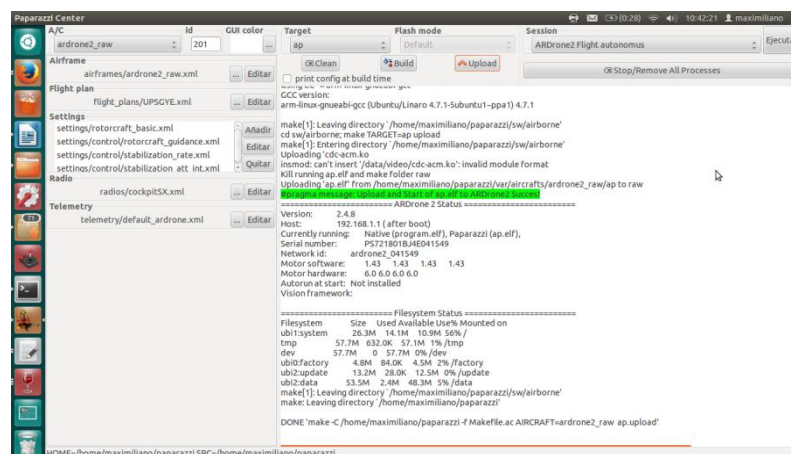


Figura 56. Compilación sin errores del airframe.

Ejecute el código seleccionando el botón Ejecutar esquina superior derecha. Y se da la apertura de la ventana del GCS (control tierra). En la parte inferior izquierda se demuestra que los sensores están operativos.

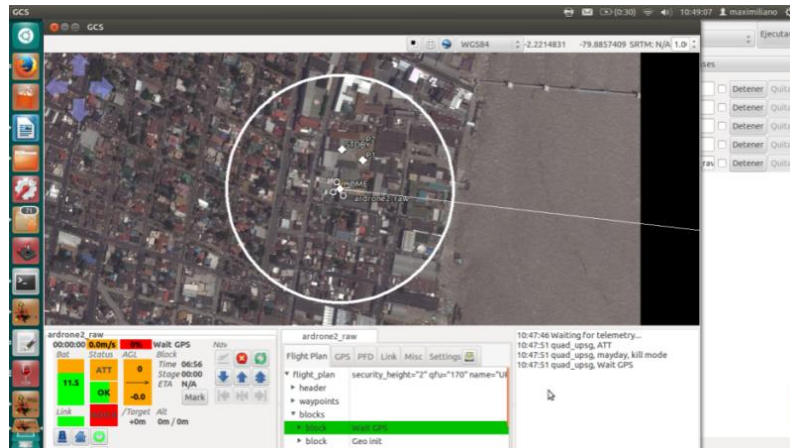


Figura 57. Pantalla GCS (control tierra) sin señal GPS.

Bat: Niveles de batería

Link: conectividad entre pc y AR.drone.

Satus: Att. Modo manual, Nav modo automático

Ok: conectividad RC o Joystick,

NOFIX: GPS no operativo.

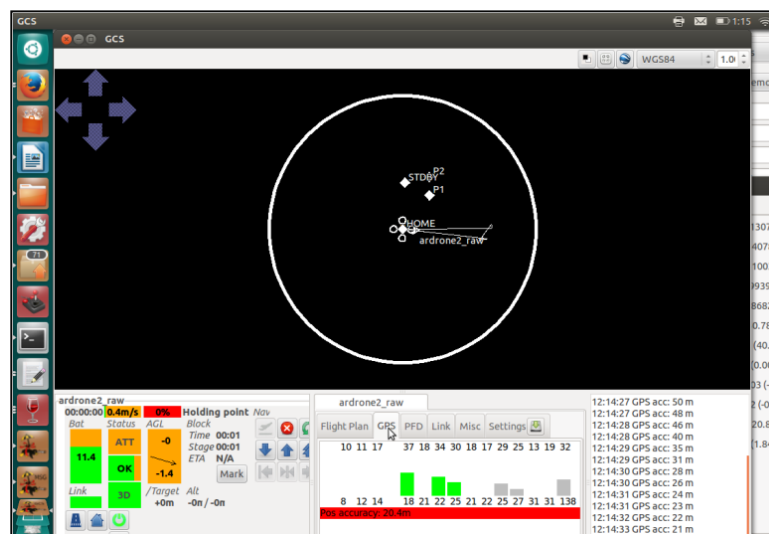


Figura 58. Pantalla GCS (control tierra) con señal GPS.

Bat: Niveles de batería

Link: conectividad entre pc y AR.drone.

Satus: Att. Modo manual, Nav modo automático

Ok: conectividad RC o Joystick,

3D: Gps Operativo.

4.2. PRÁCTICA # 2.

Tema:

- Control de motores.

Objetivo General:

- Visualizar el funcionamiento de los cuatro motores usados en la planta.

Objetivos Específicos:

- Cargar la programación adecuada para control de motores.
- Implementar el código fuente para el airframe que controla la tarjeta y los variadores de velocidad.

Materiales:

- Pc (portátil) con sistema operativo Linux Ubuntu.
- Software paparazzi.
- Joystick Genius f17.
- AR.Drone2.0.

Desarrollo:

Para esta práctica se usaron los controles del joystick. La plataforma paparazzi y la programación del control del mando genenius_f17.

4.2.1. DIAGRAMA DE CONECTIVIDAD

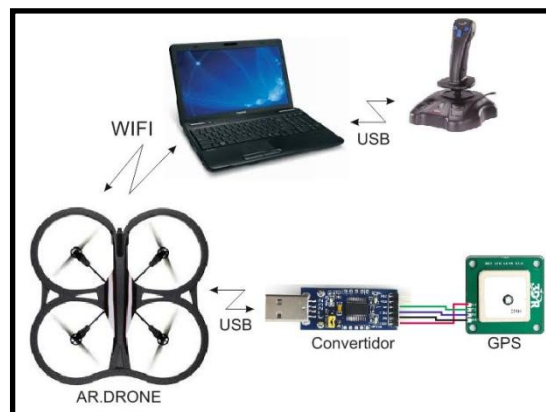


Figura 59. Diagrama de conexión de los elementos con Joystick.

Se compila el programa, ubicar en la pantalla del paparazzi center del software en la parte subir en herramientas (tools), se selecciona la opción de joystick.

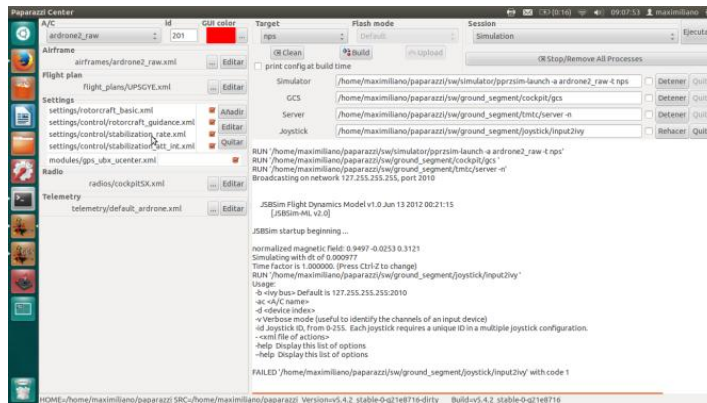


Figura 60. Condición de uso del joystick.

En la línea de joystick se escribe el tipo de joystick a usar con la siguiente línea de código por defecto esta.

/home/maximiliano/paparazzi/sw/ground_segment/joystick/input2ivy, y seguido de esta línea se agrega `-ac ardrone2_raw genius_f17.xml`.

Luego se hace clic en rehacer si todo está bien se obtiene el siguiente mensaje.

`RUN/home/maximiliano/paparazzi/sw/ground_segment/joystick/input2ivy-ac ardrone2_raw genius_f17.xml`

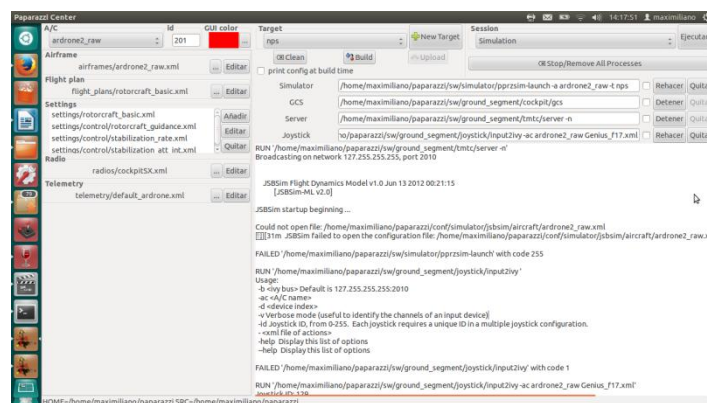


Figura 61. Se verifica el proceso de activación del joystick.

Se debe de tener en cuenta la dirección de ubicación de la carpeta donde está guardado el código del joystick esta se guarda en la siguiente dirección de carpeta.

Carpeta personal, se ubica en la carpeta paparazzi seleccione la carpeta **conf** y por último la carpeta con el nombre **joystick** donde se guardan los archivos de los joystick.



Figura 62. Ubicación de la carpeta donde el archivo debe estar guardado para su uso.

Para usar el joystick hay que crea el siguiente código fuente.

4.2.2. CÓDIGO DE PROGRAMACIÓN JOYSTICK:

```

<joystick>
<input>
<axis index="0" name="roll"/><!-- línea de configuración el axis roll-->
<axis index="1" name="pitch"/><!-- línea de configuración el axis pitch-->
<axis index="2" name="throttle"/><!-- línea de configuración el axis throttle-->
<buttonindex="0" name="trigger"/><!-- línea para asignar nombre de botón 1-->
<buttonindex="1" name="thumb"/><!-- línea para asignar nombre de botón 2-->
<buttonindex="2" name=" thumb2"/><!-- línea para nombre de botón 3-->
<buttonindex="3" name=" b3"/><!-- línea para nombre de botón 4-->

<!-- manual por default y cuando se presione trigger, AUTO1 en thumb, AUTO2 en
thumb2 -->

<variables>
<var name="mode" default="0"/>
<set var="mode" value="0" on event="trigger"/>
<set var="mode" value="1" on event="thumb"/>
<set var="mode" value="1" on event="thumb2"/>
</variables/>
< messages period="0.025">
< message class="datalink" name="RC_4CH" send_always="true">
< field name="mode" value="mode" />
< field name="throttle" value="127-Scale(throttle,0,128)" />
<field name="roll" value="roll" />
< field name="yaw" value="b3*126"/>
< field name="pitch" value="pitch"/>

```

</message>
</messages>
</joystick>

Para realizar las prueba con el joystick; se tiene que ubicar nuevamente en la pantalla GCS de paparazzi y se presiona el botón de encendido para probar los motores en este caso es b3 por 5 segundos. Se muestra los resultados de la prueba en tres fotos.



Figura 63. Primera prueba encendido de motores.



Figura 64. Motores ya en funcionamiento.

Para el proceso de apagado procedemos a presionar el mismo botón de encendido por un lazo de 5 segundos y estos se apagaran.



Figura 65. Apagado luego de presionar el botón b3 como indica la figura 33.

Esta es el la parte del código fuente del airframe que controla la sección del uso de los motores.

```
<commands><!-- comandos necesarios para este tipo de Aeronave -->
<axis name="PITCH" failsafe_value="0"/>
<axis name="ROLL" failsafe_value="0"/>
<axis name="YAW" failsafe_value="0"/>
<axis name="THRUST" failsafe_value="3000"/>
</commands>

<!-- configuración de motores 0,1,2,3 -->
<servos driver="Default">
<servo name="TOP_LEFT" no="0" min="0" neutral="1" max="500"/>
<servo name="TOP_RIGHT" no="1" min="0" neutral="1" max="500"/>
<servo name="BOTTOM_RIGHT" no="2" min="0" neutral="1" max="500"/>
<servo name="BOTTOM_LEFT" no="3" min="0" neutral="1" max="500"/>
</servos>
<!-- configuración del minimo, neutron y maximo de los servos-->

<!-- combinación de motores-->
<section name="MIXING" prefix="MOTOR_MIXING_">
<define name="TRIM_ROLL" value="0"/>
<define name="TRIM_PITCH" value="0"/>
<define name="TRIM_YAW" value="0"/>
<define name="NB_MOTOR" value="4"/>
<!--definición de cantidad de motores -->
<define name="SCALE" value="255"/>

<!--Diseño en cruz(X), en orden NW (CW), NE (CCW), SE (CW), SW (CCW)->
<define name="ROLL_COEF" value="{ 256, -256, -256, 256 }"/>
<define name="PITCH_COEF" value="{ 256, 256, -256, -256 }"/>
<define name="YAW_COEF" value="{ -256, 256, -256, 256 }"/>
<define name="THRUST_COEF" value="{ 256, 256, 256, 256 }"/>
</section>

<command_laws>
<call fun="motor_mixing_run(autopilot_motors_on,FALSE,values)"/>
<set servo="TOP_LEFT" value="motor_mixing.commands[0]"/>
<set servo="TOP_RIGHT" value="motor_mixing.commands[1]"/>
<set servo="BOTTOM_RIGHT" value="motor_mixing.commands[2]"/>
<set servo="BOTTOM_LEFT" value="motor_mixing.commands[3]"/>
</command_laws>
```

4.3. PRÁCTICA # 3.

Tema:

- Programación de movimiento de ascenso y descenso.

Objetivo General:

- Control de aceleración.

Objetivos Específicos:

- Prueba de movimiento de ascenso y descenso usando la aceleración (throttle) del joystick.

Materiales:

- Pc (portátil) con sistema operativo Linux Ubuntu.
- Software paparazzi.
- Joystick Genius f17.
- AR.Drone2.0.

Desarrollo:

Para esta práctica se utiliza el acelerador del joystick. Y el software libre para la demostración. Y el ARDrone2.0.

Una vez compilado el programa se ubican en la GCS del paparazzi, luego encenderán los motores con B3 en modo manual ATT.

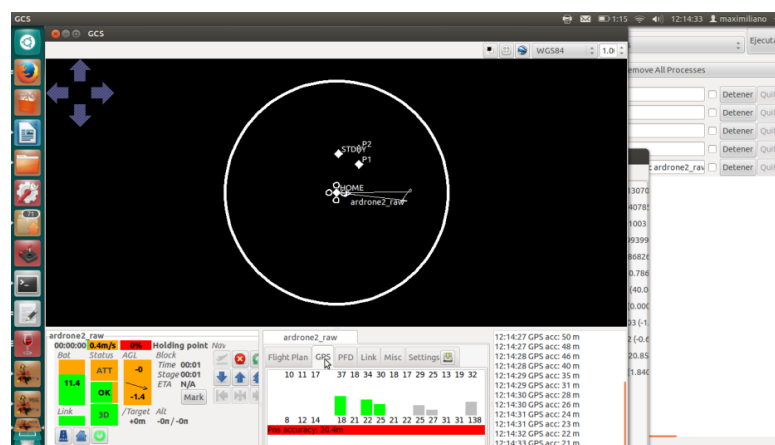


Figura 66. En la GCS, en la parte inferior izquierda muestra el estado actual es este caso modo manual (ATT).

En la parte inferior del joystick se encuentra el acelerador (throttle), con un movimiento hacia arriba se tiene que aumentar de a poco hasta llegar al máximo.



Figura 67. En la parte inferior se encuentra el mando de aceleración, como se muestra en la gráfica.

En la pantalla del GCS se selecciona la pestaña PFD en la cual el operador se observa la aceleración el grado de inclinación, el acenso y descenso.

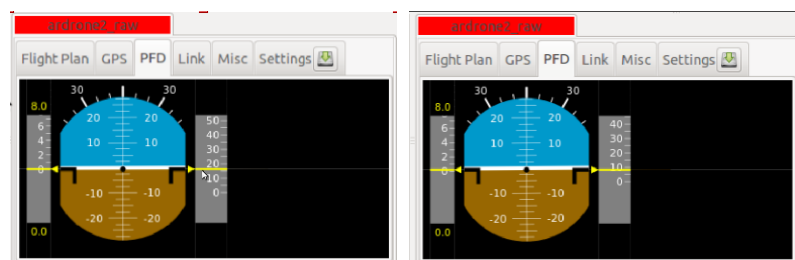


Figura 68. En la figura de la izquierda en segmento PFD observamos el horizontal en 15, en la figura de la derecha se muestra el horizontal en 10.

Este es el segmento de fuente del airframe que controla la parte de la aceleración, estabilidad y orientación.

```
<!--subsistema de orientación .-->
<section name="AHRS" prefix="AHRS_">
<!-- Toulouse -->
<define name="H_X" value="0.513081"/>
<define name="H_Y" value="-0.00242783"/>
<define name="H_Z" value="0.858336"/>
<!-- Delft -->
<!--define name="H_X" value="0.3892503"/>
<define name="H_Y" value="0.0017972"/>
<define name="H_Z" value="0.9211303"/ -->
```

```

</section>
<!--subsistema especifica que la posición y velocidad algoritmo de estimación que
está utilizando.-->
<section name="INS" prefix="INS_">
<define name="SONAR_MAX_RANGE" value="2.2"/>
<define name="SONAR_UPDATE_ON_AGL" value="TRUE"/>
</section>

<!--subsistema control de velocidad-->
<section name="STABILIZATION_RATE" prefix="STABILIZATION_RATE_">
<!--setpoints -->
<define name="SP_MAX_P" value="10000"/>
<define name="SP_MAX_Q" value="10000"/>
<define name="SP_MAX_R" value="10000"/>
<define name="DEADBAND_P" value="20"/>
<define name="DEADBAND_Q" value="20"/>
<define name="DEADBAND_R" value="200"/>
<define name="REF_TAU" value="4"/>
<!-- feedback -->
<define name="GAIN_P" value="400"/>
<define name="GAIN_Q" value="400"/>
<define name="GAIN_R" value="350"/>
<define name="IGAIN_P" value="75"/>
<define name="IGAIN_Q" value="75"/>
<define name="IGAIN_R" value="50"/>
<!-- feedforward -->
<define name="DDGAIN_P" value="300"/>
<define name="DDGAIN_Q" value="300"/>
<define name="DDGAIN_R" value="300"/>
</section>

```


4.4. PRÁCTICA # 4.

Tema:

- Programación de movimiento de rotación.

Objetivo General:

- Movimiento de giro.

Objetivos Específicos:

- Usando el control de mando joystick para la rotación en vuelo.

Materiales:

- Pc (portátil) con sistema operativo Linux Ubuntu.
- Software paparazzi.
- Joystick Genius f17.
- AR.Drone2.0.

Desarrollo:

El joystick tiene la características de axis en este caso solo tiene tres axis (X,Y) y throttle o aceleración, lo ideal sería un joystick de cuatro axis.

Para esta demostración se debió programar uno de los botones del control de mando, en este es el yaw, es el mismo de encendido B3.

```
<field name="yaw" value="b3*126" />
```

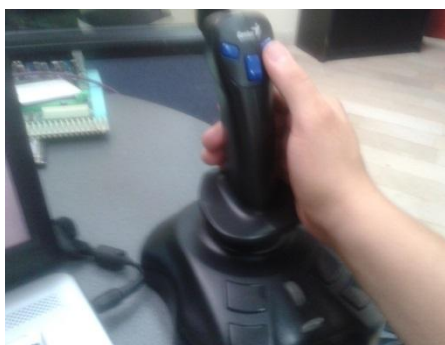


Figura 69. El botón B3 la función de Yaw.

En la pantalla de la GCS se notara el movimiento del giro cuando se dio la orden desde el control de mando.

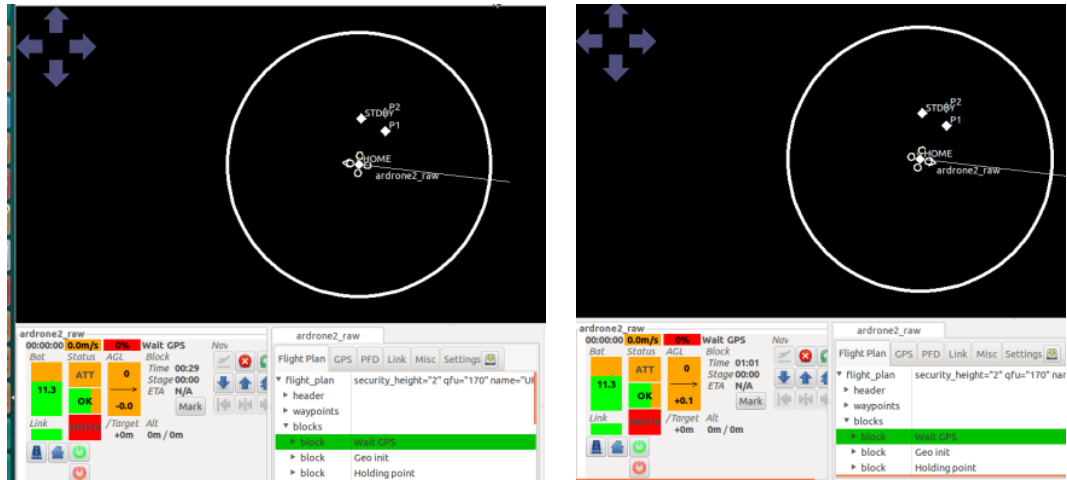


Figura 70. En la figura de la izquierda muestra la aeronave con la flecha apuntando a la izquierda, una vez que se presionó el botón B3 la dirección de la flecha ha cambiado como muestra la gráfica de la derecha.

Segmento del código fuente del airframe donde se controla el Pitch, roll y el yaw.

```
<!--subsistema control de Acctitudroll,pitch,yaw-->
```

```
<sectionname="STABILIZATION_ATTITUDE"
prefix="STABILIZATION_ATTITUDE_">
```

```
<!-- setpoints punto de ajustes -->
```

```
<define name="SP_MAX_PHI" value="45" unit="deg"/>
<define name="SP_MAX_THETA" value="45" unit="deg"/>
<define name="SP_MAX_R" value="600" unit="deg/s"/>
<define name="DEADBAND_A" value="0"/>
<define name="DEADBAND_E" value="0"/>
<define name="DEADBAND_R" value="250"/>
```

```
<!-- referencia-->
```

```
<define name="REF_OMEGA_P" value="450" unit="deg/s"/>
<define name="REF_ZETA_P" value="0.9"/>
<define name="REF_MAX_P" value="600." unit="deg/s"/>
<define name="REF_MAX_PDOT" value="RadOfDeg(8000.)"/>
<define name="REF_OMEGA_Q" value="450" unit="deg/s"/>
<define name="REF_ZETA_Q" value="0.9"/>
<define name="REF_MAX_Q" value="600." unit="deg/s"/>
<define name="REF_MAX_QDOT" value="RadOfDeg(8000.)"/>
<define name="REF_OMEGA_R" value="200" unit="deg/s"/>
<define name="REF_ZETA_R" value="0.9"/>
<define name="REF_MAX_R" value="300." unit="deg/s"/>
<define name="REF_MAX_RDOT" value="RadOfDeg(4000.)"/>
```

4.5. PRÁCTICA # 5.

Tema:

- Programación de movimiento de adelante y hacia atrás.

Objetivo General:

- Pruebas de movimiento adelante y atrás.

Objetivos Específicos:

- Usando el control de mando joystick con un movimiento hacia adelante.
- Usando el control de mando joystick con un movimiento hacia atrás.

Materiales:

- Pc (portátil) con sistema operativo Linux Ubuntu.
- Software paparazzi.
- Joystick Genius f17.
- AR.Drone2.0.

Desarrollo:

En la pantalla de la GCS se toma el punto de referencia HOME con un movimiento hacia adelante de la palanca se nota el movimiento que este hizo en modo manual.

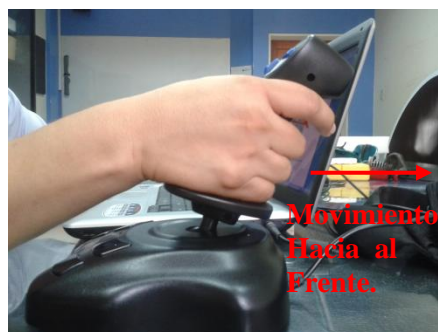


Figura 71. Movimiento hacia adelante la aeronave recibe la orden de movimiento.

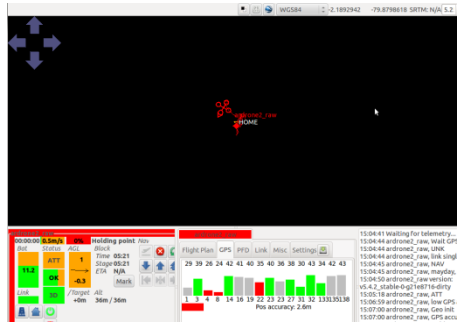


Figura 72. En la pantalla se observa que el movimiento se ha realizado con éxito desde el punto de HOME.

Se realizó el movimiento hacia atrás en modo manual similar al ejercicio anterior pero en inversa.



Figura 73. Movimiento hacia atrás desde el control de mando.



Figura 74. En la pantalla se observara que el cambio realizado en el plano de la ruta de vuelo con relación a la figura 73.

Segmento de código fuente del airframe donde se controla los movimientos para las pruebas de laboratorio.

```
<!--feedback retro alimentación hacia a atrás -->
<define name="PHI_PGAIN" value="592"/>
<define name="PHI_DGAIN" value="303"/>
<define name="PHI_IGAIN" value="0"/>

<define name="THETA_PGAIN" value="606"/>
<define name="THETA_DGAIN" value="303"/>
<define name="THETA_IGAIN" value="0"/>

<define name="PSI_PGAIN" value="529"/>
<define name="PSI_DGAIN" value="353"/>
<define name="PSI_IGAIN" value="0"/>

<!--feed forward retro alimentación hacia a delante -->
<define name="PHI_DDGIN" value="0"/>
<define name="THETA_DDGIN" value="0"/>
<define name="PSI_DDGIN" value="52"/>
</section>
```

4.6. PRÁCTICA # 6.

Tema:

- Programación de movimiento de derecha o izquierda.

Objetivo General:

- Pruebas de movimientos derecha e izquierda.

Objetivos Específicos:

- Usando el control de mando joystick con un movimiento hacia derecha.
- Usando el control de mando joystick con un movimiento hacia izquierda.

Materiales:

- Pc (portátil) con sistema operativo Linux Ubuntu.
- Software paparazzi.
- Joystick Genius f17.
- AR.Drone2.0.

Desarrollo:

En la pantalla de la GCS se toma el punto de referencia HOME con un movimiento hacia la derecha de la palanca notamos el movimiento que este hizo en modo manual.



Figura 75. Movimiento hacia la derecha la aeronave recibe la orden de movimiento.



Figura 76. La ruta de vuelo este se desplazó a la derecha.

Se realizó el movimiento hacia la izquierda en estado manual como se realizo en la práctica anterior pero en inversa.



Figura 77. Movimiento hacia el lado izquierdo desde el control de mando.

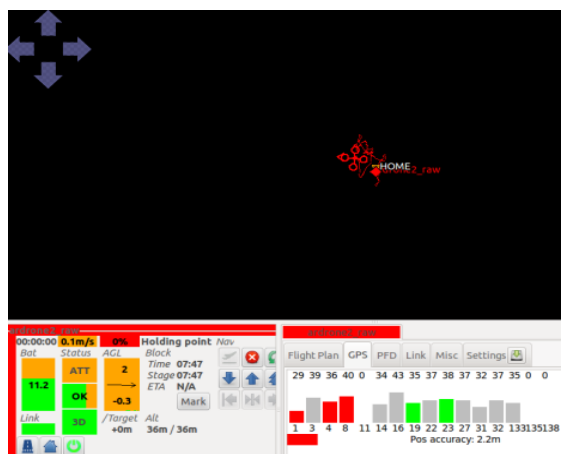


Figura 78. En la pantalla se observara que el cambio realizado en el plano de la ruta de vuelo de derecha a izquierda con relación a la figura 77.

Segmento de código fuente del airframe donde se controla la orientación vertical y horizontal para la pruebas de laboratorio.

`<!-- Parámetro de orientación vertical -->`

```
<section name="GUIDANCE_V" prefix="GUIDANCE_V_">
<define name="HOVER_KP" value="283"/>
<define name="HOVER_KD" value="82"/>
<define name="HOVER_KI" value="13"/>
<define name="NOMINAL_HOVER_THROTTLE" value="0.655"/>
<define name="ADAPT_THROTTLE_ENABLED" value="FALSE"/>
</section>
```

`<!-- Parámetro de orientación horizontal -->`

```
<section name="GUIDANCE_H" prefix="GUIDANCE_H_">
<!-- Good weather buen tiempo -->
<define name="MAX_BANK" value="20" unit="deg"/>
<!-- Bad weather mal tiempo-->
<!-- define name="MAX_BANK" value="32" unit="deg"/ -->
<define name="PGAIN" value="79"/>
<define name="DGAIN" value="100"/>
<define name="IGAIN" value="30"/>
</section>
```


4.7 PRÁCTICA #7.

Tema:

- Seguimiento de trayectoria.

Objetivo General:

- Programación de una trayectoria de vuelo.

Objetivos Específicos:

- Aplicar una ruta de vuelo autónoma.

Materiales:

- Pc (portátil) con sistema operativo Linux Ubuntu.
- Software paparazzi.
- Joystick Genius f17.
- AR.Drone2.0.

Desarrollo:

En esta práctica se usaron los elementos de Joystick para control manual ATT y para el modo de navegación autónomo NAV el computador portátil.

En la pantalla GCS en la parte inferior derecha se dará el cambio de modo de manual ATT a modo de navegación automático NAV.

Presionar el siguiente botón para la cambiar estado de la aeronave.



Figura 79. En el control de mando del Joystick se presiona el botón como muestra la gráfica esta pasa de modo manual ATT y pasa a modo de vuelo autónomo NAV.

En el software se dio el cambio a NAV la planta se movió del punto HOME al punto STDBY y se mantuvo en este punto hasta la siguiente orden.

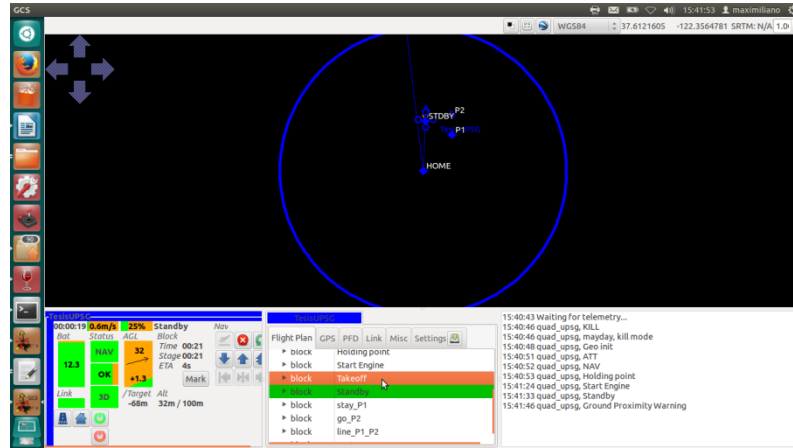


Figura 80. Se demuestró el cambio del punto Home al punto STDBY.

Siguiendo con el plan de vuelo ya diseñado el siguiente punto es desde el punto STDBY al punto P1

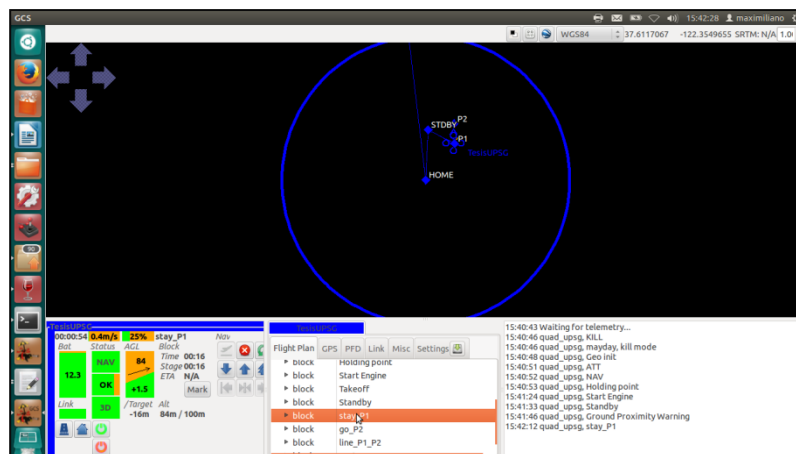


Figura 81. La aeronave sigue con las órdenes dadas en el plan de vuelo en este momento ha pasado al punto P1.

La siguiente ruta es del bloque de GO desde el punto P1 a P2, en este bloque la aeronave se mueve del punto P1 al punto P2 retornando inmediatamente.

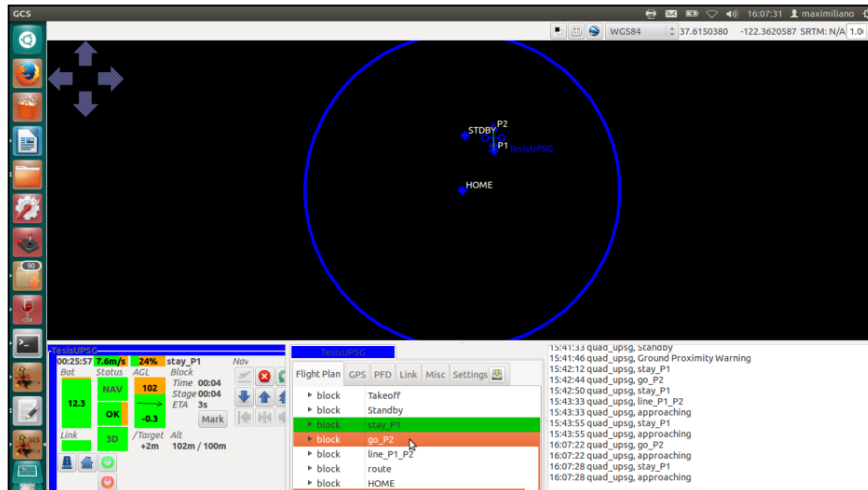


Figura 82. Bloque de ruta de vuelo GO.

El siguiente bloque LINE es muy parecido al tramo anterior es una línea de punto entre el P1 Y P2 a diferencia de GO este nos da una línea de Trayectoria en la cual la aeronave se queda en modo de estambay por unos segundo retornando a al punto P1.

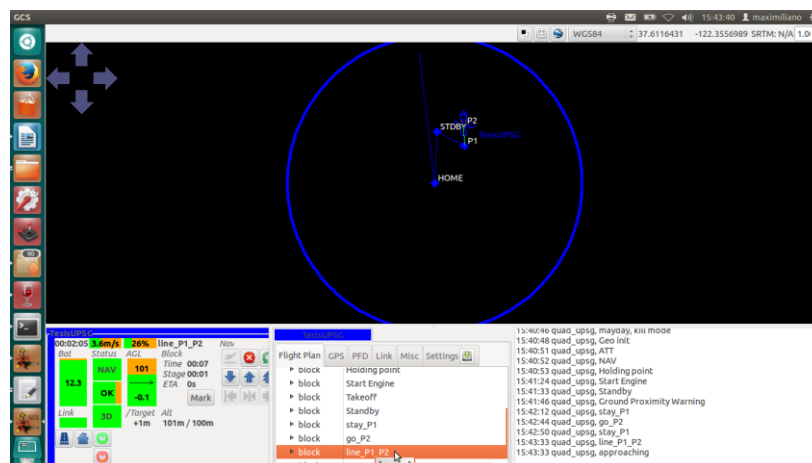


Figura 83. La aeronave sigue con la orden de ruta LINE.

En la última trayectoria es desde el punto P1 al Punto de Home. Al ubicarse en el punto de home el descenso lo hace de una manera controlada y estable y se aterriza la aeronave.

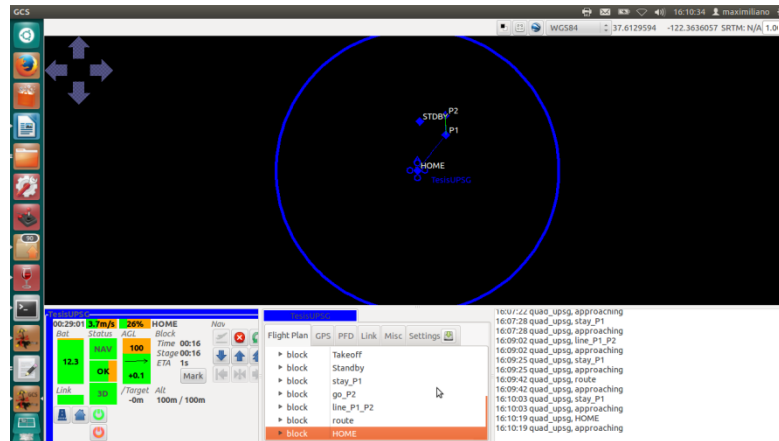


Figura 84. La aeronave regresa al punto HOME.

Para futuras rutas de vuelo se podrán usar las siguientes rutinas.

(Attitude) altitud, sólo mantener una altitud fija; la partida: mantener un determinado curso.

(Heading), la partida, para mantener un determinado curso.

(Path), ruta lista de waypoints vinculado por ir de un a otro punto.

(Circle), el círculo en torno a un punto de recorrido.

(Oval), dos semicírculos con una recta entre dos puntos de navegación.

(Follow), seguir otra aeronave.

4.7.1. CÓDIGO DE PROGRAMACIÓN PLAN DE VUELO:

```
<!DOCTYPEflight_plan SYSTEM "flight_plan.dtd">
<!--Punto inicio del plan de vuelo cordenasas x,y latitud y longitud -->
<flight_plan alt="100" ground_alt="0" lat0="-2.219777" lon0="-79.888239"
max_dist_from_home="250" name="UPSGYE" qfu="170" security_height="2">
<header>
  #include "autopilot.h"
</header>
<!--Ingreso de puntos a usar por defecto siempre se inicia con HOME y STDBY -->
<waypoints>
<waypoint name="HOME" x="0" y="0"/>
<waypoint name="STDBY" x="4.4" y="86.6"/>
<waypoint name="P1" x="50.1" y="63.0"/>
```

```

<waypoint name="P2" x="49.3" y="97.1"/>
</waypoints>
<!--Bloque de lectura de repuesta del GPS -->
<blocks>
<block name="Wait GPS">
<call fun="NavKillThrottle()"/>
<while cond="!GpsFixValid()"/>
</block>
<!--Bloque de lectura del mapa wsg84 -->
<block name="Geo init">
<while cond="LessThan(NavBlockTime(), 10)"/>
<call fun="NavSetGroundReferenceHere()"/>
</block>
<!--Bloque de lectura de tendencia de los puntos -->
<block name="Holding point">
<call fun="NavKillThrottle()"/>
<attitude pitch="0" roll="0" throttle="0" vmode="throttle" until="FALSE"/>
</block>
<!--Bloque inicio de los motores -->
<block name="Start Engine">
<call fun="NavResurrect()"/>
<attitude pitch="0" roll="0" throttle="0" vmode="throttle" until="FALSE"/>
</block>
<!--Bloque de despegue -->
<block name="Takeoff" strip_button="Takeoff" strip_icon="takeoff.png">
<exception cond="stateGetPositionEnu_f()->z > 2.0" deroute="Standby"/>
<call fun="NavSetWaypointHere(WP_HOME)"/>
</block>
<!--Bloque de posicion al punto Stanby -->
<block name="Standby" strip_button="Standby" strip_icon="home.png">
<stay wp="STDBY"/>
</block>

<!--Bloque de ubicacion al punto P1 -->
<block name="stay_P1">
<stay wp="P1"/>
</block>
<!--Bloque de ubicacion al ida al P2 -->
<block name="go_P2">
<go wp="P2"/>
<deroute block="stay_P1"/>
</block>
<!--Bloque de linea enter P1 y P2 -->
<block name="line_P1_P2">

```

```

<go from="P1" hmode="route" wp="P2"/>
<stay wp="P2" until="stage_time>10"/>
<go from="P2" hmode="route" wp="P1"/>
<deroute block="stay_P1"/>
</block>
<!--Bloque de ruta entre P1 y P2 -->
<block name="route">
<go from="P1" hmode="route" wp="P2"/>
<go from="P2" hmode="route" wp="P1"/>
<go from="P1" hmode="route" wp="P2"/>
<deroute block="stay_P1"/>
</block>
<!--Bloque para circulo -->
<!--<block name="circle">
<circle radius="nav_radius" wp="P1"/>
</block>
<block name="land here" strip_button="Land Here" strip_icon="land-right.png">
<call fun="NavSetWaypointHere(WP_HOME)"/>
</block>-->
<!--<block name="land">
<go wp="STDBY"/>
</block>-->
<!--<block name="flare">
<exception cond="NavDetectGround()" deroute="Holding point"/>
<exception cond="!nav_is_in_flight()" deroute="landed"/>
<call fun="NavStartDetectGround()"/>
<stay climb="-0.8" vmode="climb" wp="HOME"/>
</block>
<block name="landed"><!--Bloque de aterrizaje-->
<attitude pitch="0" roll="0" throttle="0" vmode="throttle" until="FALSE"/>
</block>-->

</blocks>
</flight_plan>

```

4.8. CÓDIGO COMPLETO DEL AIRFRAME:

```

<!DOCTYPE airframe SYSTEM "airframe.dtd">

```

```

<!-- Multirotor Quadrocoptero sistema autopiloto
Universidad Politécnica Salesiana Sede Guayaquil
AR.drone gps Ublox y wifi-->

```

```

<airframe name="ardrone2_raw">

  <firmware name="rotorcraft">
  <target name="ap" board="ardrone2_raw"><!--Tipo de placa autopiloto -->
  <define name="AUTOPILOT_DISABLE_AHRS_KILL"/>
  <subsystemname="telemetry" type="udp"/><!--Para sensores de comunicación -->
  <subsystem name="radio_control" type="datalink"/><!--Para control de RC y
  joystick -->
  </target>

  <target name="nps" board="pc">
  <subsystem name="fdm" type="jsbsim"/>
  <subsystem name="radio_control" type="ppm"/>
  </target>

  <define name="USE_SONAR" value="TRUE"/>

    <!-- Sección de Subsistemas -->
  <subsystem name="motor_mixing"/>
  <subsystem name="actuators" type="ardrone2"/><!--Para control de los motores-->
  <subsystemname="imu" type="ardrone2"/><!--Para sensores de imu unidad de
  medición inercial-->
  <subsystemname="gps" type="ublox"/><!--para sensores de Gps -->
  <subsystemname="stabilization" type="int_quat"/><!--para sensores de
  estabilización -->
  <subsystemname="ahrs" type="int_cmpl_quat"/><!--para sensores actitud rumbo
  cabeceo balanceo -->
  <subsystemname="ins" type="extended"/><!--para Sistema de navegación integrado
  -->
  </firmware>

  <modules main_freq="512">
  <load name="gps_ubx_ucenter.xml"/>
  <load name="send_imu_mag_current.xml"/>
  <load name="file_logger.xml"/>
  <load name="geo_mag.xml"/>
  <load name="../../sw/ext/ardrone2_vision/modules/Mjpeg/ViewVideo.xml"/>
  </modules>

  <commands>
  <axis name="PITCH" failsafe_value="0"/>
  <axis name="ROLL" failsafe_value="0"/>
  <axis name="YAW" failsafe_value="0"/>
  <axis name="THRUST" failsafe_value="3000"/>

```

```

</commands>
<!--configuración de motores 0,1,2,3 -->
<servos driver="Default">
<servo name="TOP_LEFT" no="0" min="0" neutral="1" max="500"/>
<servo name="TOP_RIGHT" no="1" min="0" neutral="1" max="500"/>
<servo name="BOTTOM_RIGHT" no="2" min="0" neutral="1" max="500"/>
<servo name="BOTTOM_LEFT" no="3" min="0" neutral="1" max="500"/>
</servos>
<!--combinación de motores-->
<section name="MIXING" prefix="MOTOR_MIXING_">
<define name="TRIM_ROLL" value="0"/>
<define name="TRIM_PITCH" value="0"/>
<define name="TRIM_YAW" value="0"/>
<define name="NB_MOTOR" value="4"/>
<define name="SCALE" value="255"/>

<!-- Diseño en cruz (X), con orden NW (CW), NE (CCW), SE (CW), SW (CCW) --
>
<define name="ROLL_COEF" value="{ 256, -256, -256, 256 }"/>
<define name="PITCH_COEF" value="{ 256, 256, -256, -256 }"/>
<define name="YAW_COEF" value="{ -256, 256, -256, 256 }"/>
<define name="THRUST_COEF" value="{ 256, 256, 256, 256 }"/>
</section>

<command_laws>
<call fun="motor_mixing_run(autopilot_motors_on,FALSE,values)"/>
<set servo="TOP_LEFT" value="motor_mixing.commands[0]"/>
<set servo="TOP_RIGHT" value="motor_mixing.commands[1]"/>
<set servo="BOTTOM_RIGHT" value="motor_mixing.commands[2]"/>
<set servo="BOTTOM_LEFT" value="motor_mixing.commands[3]"/>
</command_laws>
<!--subsistema de orientación x, y.-->
<section name="IMU" prefix="IMU_">

<!--Accelerometer -->
<define name="ACCEL_X_NEUTRAL" value="2048"/>
<define name="ACCEL_Y_NEUTRAL" value="2048"/>
<define name="ACCEL_Z_NEUTRAL" value="2048"/>

<!-- Magneto calibration -->
<define name="MAG_X_NEUTRAL" value="6"/>
<define name="MAG_Y_NEUTRAL" value="8"/>
<define name="MAG_Z_NEUTRAL" value="34"/>
<define name="MAG_X_SENS" value="20.9085278928" integer="16"/>

```



```

<define name="MAG_Y_SENS" value="23.9440720651" integer="16"/>
<define name="MAG_Z_SENS" value="24.1479384029" integer="16"/>
<!-- Magneto current calibration -->
<define name="MAG_X_CURRENT_COEF" value="-0.00459066515564"/>
<define name="MAG_Y_CURRENT_COEF" value="-0.00459056529077"/>
<define name="MAG_Z_CURRENT_COEF" value="-0.000883245335938"/>

```

```

<define name="BODY_TO_IMU_PHI" value="0." unit="deg"/>
<define name="BODY_TO_IMU_THETA" value="0." unit="deg"/>
<define name="BODY_TO_IMU_PSI" value="0." unit="deg"/>
</section>

```

```

<!--Campo magnetico Local -->
<!-- http://wiki.paparazziuav.org/wiki/Subsystem/ahrs#Local\_Magnetic\_Field -->
<!--sistema de orientación .-->

```

```

<section name="AHRS" prefix="AHRS_">

```

```

<!-- Toulouse -->

```

```

<define name="H_X" value="0.513081"/>
<define name="H_Y" value="-0.00242783"/>
<define name="H_Z" value="0.858336"/>

```

```

<!-- Delft -->

```

```

<!--define name="H_X" value="0.3892503"/>
<define name="H_Y" value="0.0017972"/>
<define name="H_Z" value="0.9211303"/ -->

```

```

</section>

```

```

<!--sistema especifica que la posición y velocidad algoritmo de estimación que
está utilizando.-->

```

```

<section name="INS" prefix="INS_">

```

```

<define name="SONAR_MAX_RANGE" value="2.2"/>
<define name="SONAR_UPDATE_ON_AGL" value="TRUE"/>

```

```

</section>

```

```

<!--sistema control de velocidad-->

```

```

<section name="STABILIZATION_RATE" prefix="STABILIZATION_RATE_">

```

```

<!--setpoints -->

```

```

<define name="SP_MAX_P" value="10000"/>
<define name="SP_MAX_Q" value="10000"/>
<define name="SP_MAX_R" value="10000"/>
<define name="DEADBAND_P" value="20"/>
<define name="DEADBAND_Q" value="20"/>
<define name="DEADBAND_R" value="200"/>
<define name="REF_TAU" value="4"/>

```

```

<!-- feedback -->
<define name="GAIN_P" value="400"/>
<define name="GAIN_Q" value="400"/>
<define name="GAIN_R" value="350"/>

<define name="IGAIN_P" value="75"/>
<define name="IGAIN_Q" value="75"/>
<define name="IGAIN_R" value="50"/>

<!-- feedforward -->
<define name="DDGAIN_P" value="300"/>
<define name="DDGAIN_Q" value="300"/>
<define name="DDGAIN_R" value="300"/>
</section>
<!--sistema control de Actitud roll, pitch, yaw-->
<section name="STABILIZATION_ATTITUDE"
prefix="STABILIZATION_ATTITUDE_">
<!--setpointspunto de ajustes -->
<define name="SP_MAX_PHI" value="45" unit="deg"/>
<define name="SP_MAX_THETA" value="45" unit="deg"/>
<define name="SP_MAX_R" value="600" unit="deg/s"/>
<define name="DEADBAND_A" value="0"/>
<define name="DEADBAND_E" value="0"/>
<define name="DEADBAND_R" value="250"/>

<!-- referencia-->
<define name="REF_OMEGA_P" value="450" unit="deg/s"/>
<define name="REF_ZETA_P" value="0.9"/>
<define name="REF_MAX_P" value="600." unit="deg/s"/>
<define name="REF_MAX_PDOT" value="RadOfDeg(8000.)"/>

<define name="REF_OMEGA_Q" value="450" unit="deg/s"/>
<define name="REF_ZETA_Q" value="0.9"/>
<define name="REF_MAX_Q" value="600." unit="deg/s"/>
<define name="REF_MAX_QDOT" value="RadOfDeg(8000.)"/>

<define name="REF_OMEGA_R" value="200" unit="deg/s"/>
<define name="REF_ZETA_R" value="0.9"/>
<define name="REF_MAX_R" value="300." unit="deg/s"/>
<define name="REF_MAX_RDOT" value="RadOfDeg(4000.)"/>

<!-- retro alimentación hacia a atrás -->
<define name="PHI_PGAIN" value="592"/>

```

```

<define name="PHI_DGAIN" value="303"/>
<define name="PHI_IGAIN" value="0"/>

<define name="THETA_PGAIN" value="606"/>
<define name="THETA_DGAIN" value="303"/>
<define name="THETA_IGAIN" value="0"/>

<define name="PSI_PGAIN" value="529"/>
<define name="PSI_DGAIN" value="353"/>
<define name="PSI_IGAIN" value="0"/>

<!-- retro alimentación hacia a delante -->
<define name="PHI_DDGIN" value="0"/>
<define name="THETA_DDGIN" value="0"/>
<define name="PSI_DDGIN" value="52"/>
</section>
<!-- Parámetro de orientación vertical -->
<section name="GUIDANCE_V" prefix="GUIDANCE_V_">
<define name="HOVER_KP" value="283"/>
<define name="HOVER_KD" value="82"/>
<define name="HOVER_KI" value="13"/>
<define name="NOMINAL_HOVER_THROTTLE" value="0.655"/>
<define name="ADAPT_THROTTLE_ENABLED" value="FALSE"/>
</section>
<!-- Parámetro de orientación horizontal -->
<section name="GUIDANCE_H" prefix="GUIDANCE_H_">
<!-- buen tiempo -->
<define name="MAX_BANK" value="20" unit="deg"/>
<!-- mal tiempo-->
<!-- define name="MAX_BANK" value="32" unit="deg"/ -->
<define name="PGAIN" value="79"/>
<define name="DGAIN" value="100"/>
<define name="IGAIN" value="30"/>
</section>
<!--para poder hacer la simulacion es el sgte. Bloque -->
<section name="SIMULATOR" prefix="NPS_">
<define name="ACTUATOR_NAMES" value="{&quot;nw_motor&quot;,
&quot;ne_motor&quot;, &quot;se_motor&quot;, &quot;sw_motor&quot;}"/>
<define name="JSBSIM_MODEL" value="&quot;simple_ardrone2&quot;"/>
<define name="JSBSIM_INIT" value="&quot;reset00&quot;"/>
<define name="SENSORS_PARAMS"
value="&quot;nps_sensors_params_ardrone2.h&quot;"/>
</section>

```

```

<!--Para el control el cambio de estado de manual a auto1 y auto 2 -->
<section name="AUTOPILOT">
<define name="MODE_STARTUP" value="AP_MODE_NAV"/>
<define name="MODE_MANUAL" value="AP_MODE_ATTITUDE_DIRECT"/>
<define name="MODE_AUTO1" value="AP_MODE_ATTITUDE_Z_HOLD"/>
<define name="MODE_AUTO2" value="AP_MODE_NAV"/>
</section>

<!--Segmento de alerta de bacteria baja -->
<section name="BAT">
<define name="MILLIAMP_AT_FULL_THROTTLE" value="8700"/>
<define name="CATASTROPHIC_BAT_LEVEL" value="9.3" unit="V"/>
<define name="CRITIC_BAT_LEVEL" value="9.6" unit="V"/>
<define name="LOW_BAT_LEVEL" value="9.7" unit="V"/>
<define name="MAX_BAT_LEVEL" value="12.4" unit="V"/>
</section>
<section name="GCS">
<define name="ALT_SHIFT_PLUS_PLUS" value="5"/>
<define name="ALT_SHIFT_PLUS" value="1"/>
<define name="ALT_SHIFT_MINUS" value="-1"/>
<define name="SPEECH_NAME" value="quad_upsg"/>
<define name="AC_ICON" value="rotorcrafter"/>
</section>
</airframe>

```

CONCLUSIONES

- Con las pruebas realizadas denotamos muchas experiencias, la estabilidad de la planta fue uno de los principales problemas, el viento era uno de estos factores que influían de manera negativa.
- Otros de los factores que intervenían era la posición, se confirmó que mientras más alto realizaba el vuelo el equipo más estabilidad tenía, en vuelos bajos su equilibrio no era del todo correcto y tendía a inclinarse a ciertos lados, esto debido a que perdía comunicación con los satélites.
- En sitios cerrados la comunicación con los satélites es nula, por ende no se puede planificar una ruta, en estos casos el uso debe ser netamente de forma manual y tomado en consideración el pulso del piloto.
- Como el mecanismo de comunicación entre la planta se la realiza mediante Wi-Fi, esta respondió de una manera ideal llevándola a una distancia de 50mt sin problema alguno.

RECOMENDACIONES

- EL ARdrone2.0 presenta un problema estructural en la parte de los motores, el eje que sostiene las hélices, por eso hay que reforzar esta parte del drone para evitar daños en los engranes y motores.
- Para cambiar cualquier pieza en la estructura de la aeronave en caso de daño alguno, tener las herramientas necesaria y revisar sitio web oficiales.
- Hay que verificar la conexión WI-FI entre el computador y el ARdrone2.0 para realizar una carga efectiva. Para evitar tener problemas durante el vuelo y tener un excelente funcionamiento.
- Los vuelos se deben realizar en un área amplia y sin obstáculos, no perder de vista la nave para evitar colisiones o daños estructurales.
- Para pruebas de vuelo autónomo hay que tener un clima favorable y despejado para una mejor conectividad con el GPS.
- No utilizar el ARdrone2.0 con la carga de batería baja, ya que puede sufrir una caída y provocar daños en la estructura del drone.
- La plataforma paparazzi no ha sido creada para fines comerciales solo para fines de investigación y desarrollo.

CRONOGRAMA

Los meses mencionados se considerarán a partir de la aprobación del tema.

Cronograma de Desarrollo de Tesis						
	1er Mes	2do Mes	3er Mes	4to Mes	5to Mes	6to Mes
Investigación	■					
Pruebas de (Hardware y Software)		■				■
Pruebas de Dispositivos Electrónicos		■	■			
Análisis e Investigación del Sistema autopiloto			■	■		
Implementación, pruebas y corrección de errores				■	■	
Diseño de Prototipo				■		
Desarrollo de la Monografía					■	■
Presentación de Documentación y la Implementación					■	■

PRESUPUESTO DE LA PLANTA.

PRESUPUESTO	
Implementos y gastos varios	Costo
1 AEDrone 2.0	\$650,00
1 GPS_Ublox lea 6	\$ 127,00
1Cable de FTDI 232.	\$ 25,00
1 Convertidor FTDI 232.	\$ 27,00
1 Joystick Genius F_17	\$30,00
1 Laptop TOSHIBA	\$650,00
Gatos Varios	\$ 150,00
Total	\$ 1.659,00

REFERENCIAS

- bibdigital. (2013). bibdigital.epn.edu.ec. Recuperado el 17 de Enero de 2015, de <http://bibdigital.epn.edu.ec/bitstream/15000/7372/1/CD-5521.pdf>.
- EEUU, G. d. (2014). <http://www.gps.gov/>. Recuperado el 16 de Marzo de 2015, de <http://www.gps.gov/systems/gps/>
- Fernandez, J. C. (2013). uc3m.es. Recuperado el 05 de Marzo de 2015
- ftdichip. (2015). <http://www.ftdichip.com>. Recuperado el Marzo de 2015, de <http://www.ftdichip.com/Products/ICs/FT232R.htm>
- ftdichip. (2015). <http://www.ftdichip.com/>. Recuperado el 22 de Marzo de 2015, de <http://www.ftdichip.com/Products/Cables/USBTTLSerial.htm#TTL-232R-5V>
- Gobierno, E. (2014). <http://www.gps.gov/>. Recuperado el 16 de Marzo de 2015, de <http://www.gps.gov/systems/gps/>
- hobbyking. (2012). <http://www.hobbyking.com/>. Recuperado el 16 de Marzo de 2015, de http://www.hobbyking.com/hobbyking/store/__33998__UBLOX_LEA_6H_GPS_Module_w_Built_in_Antenna_2_5m_Accuracy.html?strSearch=gps%20lea
- Idetec. (2015). <http://www.ideastechnology.com>. Recuperado el 11 de Febrero de 2015, de <http://www.ideastechnology.com/?q=node/122>
- medida.es. (2012). <http://www.sensores-de-medida.es/>. Recuperado el 7 de Enero de 2015, de http://www.sensores-de-medida.es/sensing_sl/SENSORES-Y-TRANSDUCTORES_35
- Munoz, M. (2013). <http://www.manualvuelo.com/>. Obtenido de <http://www.manualvuelo.com/INS/INS22.html>
- paparazziuav. (2012). <http://wiki.paparazziuav.org/>. Recuperado el 26 de Enero de 2015, de http://wiki.paparazziuav.org/wiki/Theory_of_Operation
- paparazziuav. (2014). <http://wiki.paparazziuav.org/wik>. Recuperado el Febrero de 05 de 2015, de <http://wiki.paparazziuav.org/wiki/Overview>
- paparazziuav. (2014). http://wiki.paparazziuav.org/wiki/Main_Page. Recuperado el 18 de Diciembre de 2014, de http://wiki.paparazziuav.org/wiki/Getting_Started
- parrot.com. (2014). <http://ardrone2.parrot.com/>. Recuperado el 17 de Febrero de 2015
- ublox. (2014). <http://www.u-blox.com/en/gps-module> Recuperado el 16 de Enero de 2015

Víctor Manuel Pérez Lozano. (2010). <http://sgitt-otri.ua.es>. Recuperado el 18 de Marzo de 2015, de <http://sgitt-otri.ua.es/es/empresa/documentos/ot-1502-drones.pdf>

wordpress. (Mayo de 2011). <https://redwifi.wordpress.com>. Obtenido de <https://redwifi.wordpress.com/definicion-de-wifi/>

Anexos

Anexo A

INSTALACION DE PAPAARAZZI

La instalación de la plataforma paparazzi se lo puede realizar en cualquier ordenador portátil o estación de trabajo que ejecute el sistema operativo Linux Ubuntu.

Básicamente la instalación consta de cuatro pasos.

- Instalar herramientas y tener los requisitos previos necesarios para Paparazzi.
- Descargar el código fuente desde el repositorio de código fuente.
- Compilar el programa Paparazzi de código fuente
- Realizar la configuración final

Instalación en Ubuntu

Ingresando a la ruta en donde se va a instalar los paquetes de software de los paparazzi.

Ejecutamos en un terminal:

```
sudo add-apt ppa-repositorio: paparazzi-UAV / ppa
```

(paparazziuav.org, http://wiki.paparazziuav.org/wiki/Main_Page, 2014)

Para actualizar el inventario de paquetes del sistema e instalar las principales dependencias de software Paparazzi. Ejecutamos estos comandos

```
sudo apt-get update
```

```
sudo apt-get install paparazzi-dev
```

```
sudo add-apt-repository ppa:paparazzi-uav/ppa && sudo add-apt-repository  
ppa:terry.guo/gcc-arm-embedded && sudo apt-get update
```

Donde se empiezan a cargar los siguientes archivos, como se muestra en la imagen.

```
root@omar-HP-455-Notebook-PC: /home/omar
Obj http://security.ubuntu.com precise-security/universe i386 Packages
Obj http://security.ubuntu.com precise-security/multiverse i386 Packages
Obj http://security.ubuntu.com precise-security/main TranslationIndex
Obj http://security.ubuntu.com precise-security/multiverse TranslationIndex
Obj http://security.ubuntu.com precise-security/restricted TranslationIndex
Obj http://security.ubuntu.com precise-security/universe TranslationIndex
Obj http://ppa.launchpad.net precise Release
Obj http://security.ubuntu.com precise-security/main Translation-en
Obj http://security.ubuntu.com precise-security/multiverse Translation-en
Obj http://ppa.launchpad.net precise/main Sources
Obj http://ppa.launchpad.net precise/main i386 Packages
Ign http://ppa.launchpad.net precise/main TranslationIndex
Obj http://security.ubuntu.com precise-security/restricted Translation-en
Obj http://security.ubuntu.com precise-security/universe Translation-en
Obj http://ppa.launchpad.net precise/main Sources
Obj http://ppa.launchpad.net precise/main i386 Packages
Ign http://ppa.launchpad.net precise/main TranslationIndex
Ign http://ppa.launchpad.net precise/main Translation-es_EC
Ign http://ppa.launchpad.net precise/main Translation-es
Ign http://ppa.launchpad.net precise/main Translation-en
Ign http://ppa.launchpad.net precise/main Translation-es_EC
Ign http://ppa.launchpad.net precise/main Translation-es
Ign http://ppa.launchpad.net precise/main Translation-en
Leyendo lista de paquetes... Hecho
root@omar-HP-455-Notebook-PC: /home/omar#
```

Luego de culminada la instalación de esos paquetes proseguimos con otro grupo ejecutando los siguientes comandos

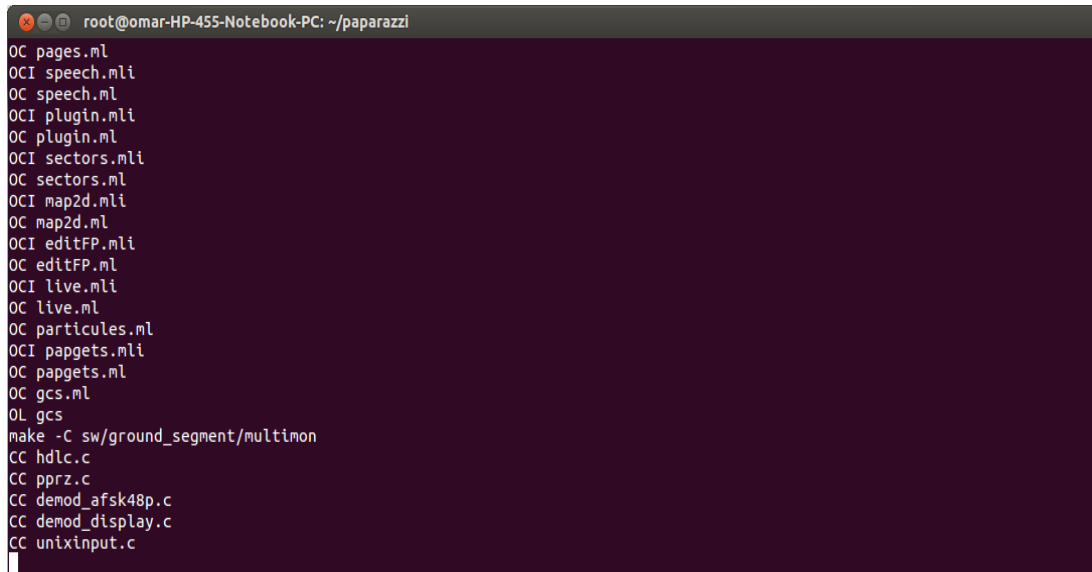
`sudo apt-get install paparazzi-dev gcc-arm-none-eabi && cd ~ && git clone https://github.com/paparazzi/paparazzi.git`

```
root@omar-HP-455-Notebook-PC: ~
Configurando libgnome2-0 (2.32.1-2ubuntu1.1) ...
Configurando libbonoboui2-0 (2.24.5-0ubuntu1.1) ...
Configurando libgnomeui-0 (2.24.5-2ubuntu2) ...
Configurando libstdc++6-4.6-dev (4.6.3-1ubuntu5) ...
Configurando g++-4.6 (4.6.3-1ubuntu5) ...
Configurando g++ (4:4.6.3-1ubuntu5) ...
update-alternatives: utilizando /usr/bin/g++ para proveer /usr/bin/c++ (c++) en modo automático.
Configurando build-essential (11.5ubuntu2.1) ...
Configurando libgnome2-dev (2.32.1-2ubuntu1.1) ...
Configurando libbonoboui2-dev (2.24.5-0ubuntu1.1) ...
Configurando libgnomeui-dev (2.24.5-2ubuntu2) ...
Configurando liblglgtk2-gnome-ocaml (2.14.2+dfsg-2build1) ...
Configurando liblglgtk2-gnome-ocaml-dev (2.14.2+dfsg-2build1) ...
Configurando python-gnome2 (2.28.1+dfsg-1) ...
Configurando paparazzi-dev (3.12) ...
Procesando disparadores para libc-bin ...
ldconfig deferred processing now taking place
Procesando disparadores para python-support ...
Cloning into 'paparazzi'...
remote: Counting objects: 114520, done.
remote: Compressing objects: 100% (23472/23472), done.
remote: Total 114520 (delta 90483), reused 114368 (delta 90331)
Receiving objects: 100% (114520/114520), 54.04 MiB | 120 KiB/s, done.
Resolving deltas: 100% (90483/90483), done.
root@omar-HP-455-Notebook-PC: ~#
```

`cd ~/paparazzi && git checkout master && sudo cp conf/system/udev/rules/50-paparazzi.rules /etc/udev/rules.d/`

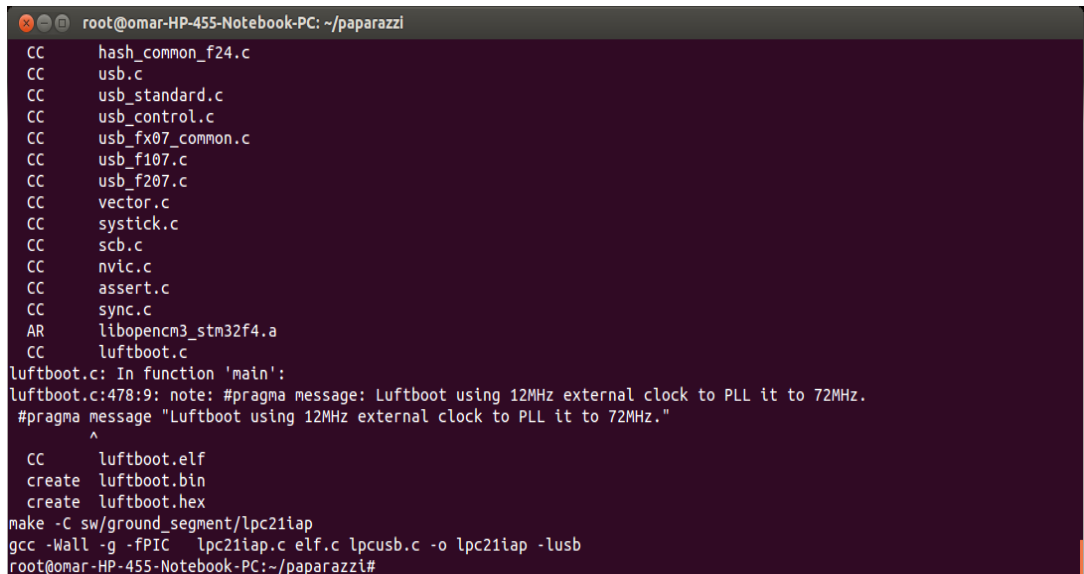
siguiendo con el procedimiento colocamos el siguiente comando en el terminal.

make clean && make && ./paparazzi



```
root@omar-HP-455-Notebook-PC: ~/paparazzi
OC pages.ml
OCI speech.mli
OC speech.ml
OCI plugin.mli
OC plugin.ml
OCI sectors.mli
OC sectors.ml
OCI map2d.mli
OC map2d.ml
OCI editFP.mli
OC editFP.ml
OCI live.mli
OC live.ml
OC particules.ml
OCI papgets.mli
OC papgets.ml
OC gcs.ml
OL gcs
make -C sw/ground_segment/multimon
CC hdlc.c
CC pprz.c
CC demod_afsk48p.c
CC demod_display.c
CC unixinput.c
```

Cuando ya se instala todo debe de salir la siguiente pantalla.



```
root@omar-HP-455-Notebook-PC: ~/paparazzi
CC hash_common_f24.c
CC usb.c
CC usb_standard.c
CC usb_control.c
CC usb_fx07_common.c
CC usb_f107.c
CC usb_f207.c
CC vector.c
CC systick.c
CC scb.c
CC nvic.c
CC assert.c
CC sync.c
AR libopencm3_stm32f4.a
CC luftboot.c
luftboot.c: In function 'main':
luftboot.c:478:9: note: #pragma message: Luftboot using 12MHz external clock to PLL it to 72MHz.
#pragma message "Luftboot using 12MHz external clock to PLL it to 72MHz."
^
CC luftboot.elf
create luftboot.bin
create luftboot.hex
make -C sw/ground_segment/lpc21iap
gcc -Wall -g -fPIC lpc21iap.c elf.c lpcusb.c -o lpc21iap -lusb
root@omar-HP-455-Notebook-PC:~/paparazzi#
```

OBTENER EL CÓDIGO FUENTE

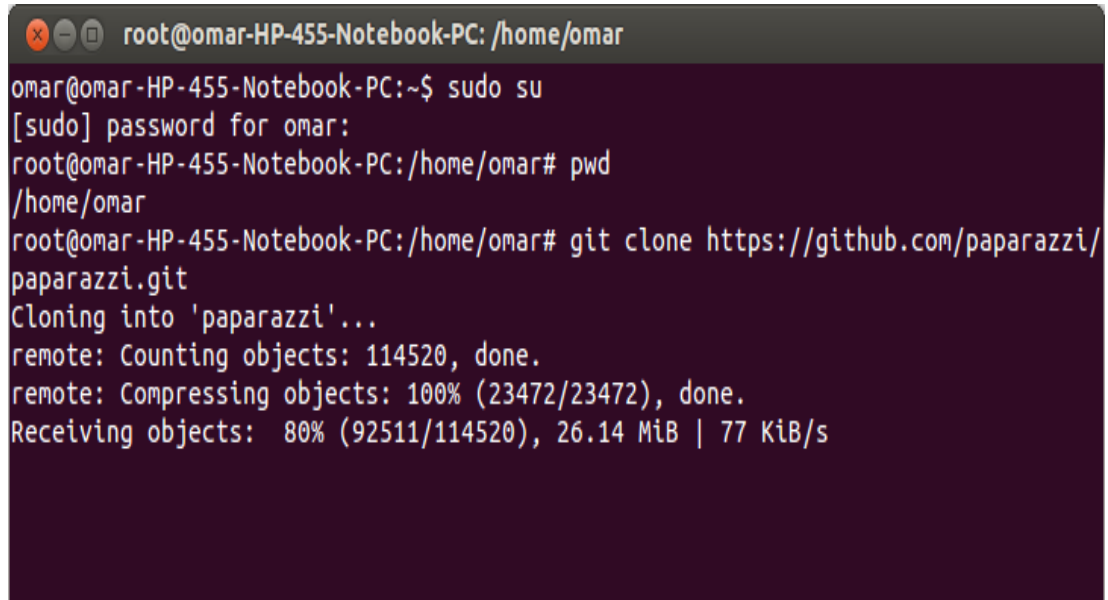
El código fuente Paparazzi está alojado en [Github](#) del wiki de . Otra opción disponible para descargar en un archivo ZIP o archivo tar es la siguiente.

<https://github.com/paparazzi/paparazzi/tags> , se recomienda clonar el repositorio con [git](#) .

Desde el directorio donde se vaya a instalar.

En este caso ejecutamos el siguiente comando en el terminal:

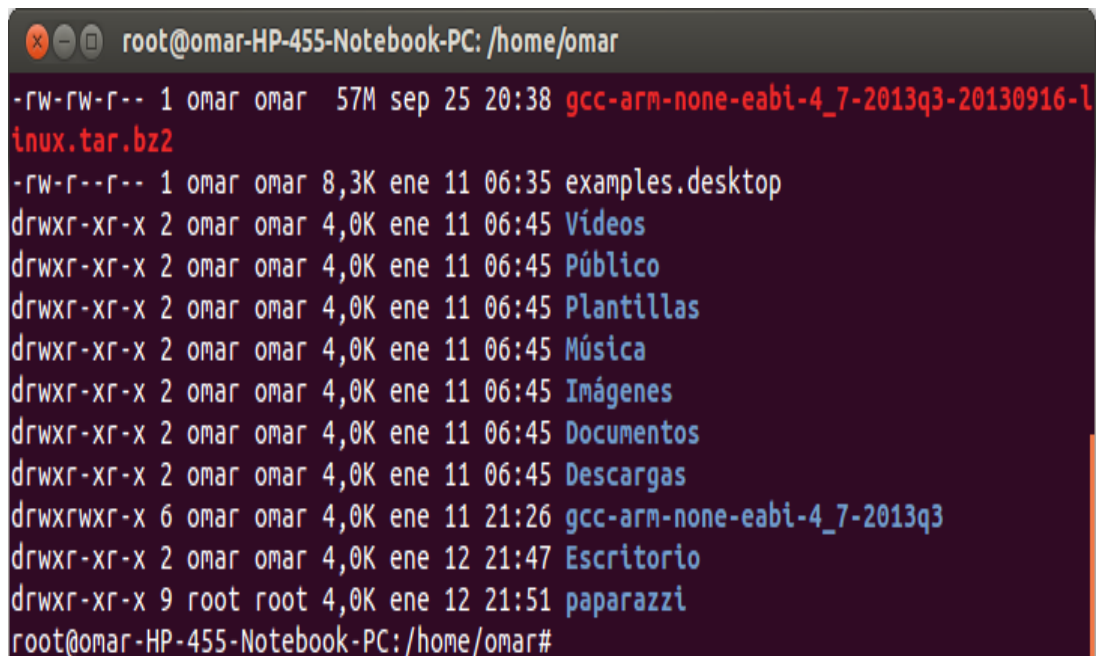
```
git clone https://github.com/paparazzi/paparazzi.git
```



```
root@omar-HP-455-Notebook-PC: /home/omar
omar@omar-HP-455-Notebook-PC:~$ sudo su
[sudo] password for omar:
root@omar-HP-455-Notebook-PC:/home/omar# pwd
/home/omar
root@omar-HP-455-Notebook-PC:/home/omar# git clone https://github.com/paparazzi/
paparazzi.git
Cloning into 'paparazzi'...
remote: Counting objects: 114520, done.
remote: Compressing objects: 100% (23472/23472), done.
Receiving objects: 80% (92511/114520), 26.14 MiB | 77 KiB/s
```

Una vez terminado procedemos a verificar si se encuentra la carpeta paparazzi con el comando

ls -ltrh en el terminal



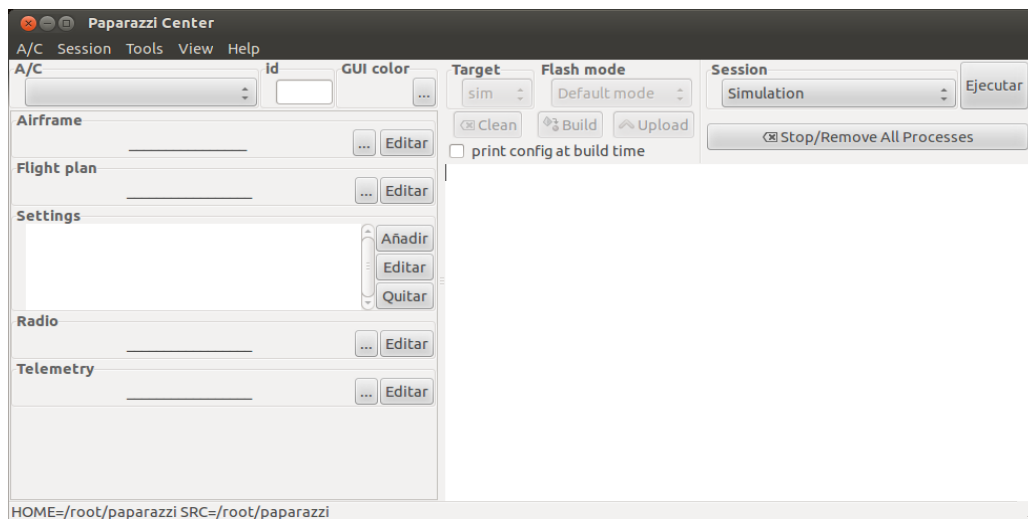
```
root@omar-HP-455-Notebook-PC: /home/omar
-rw-rw-r-- 1 omar omar 57M sep 25 20:38 gcc-arm-none-eabi-4_7-2013q3-20130916-l
inux.tar.bz2
-rw-r--r-- 1 omar omar 8,3K ene 11 06:35 examples.desktop
drwxr-xr-x 2 omar omar 4,0K ene 11 06:45 Vídeos
drwxr-xr-x 2 omar omar 4,0K ene 11 06:45 Público
drwxr-xr-x 2 omar omar 4,0K ene 11 06:45 Plantillas
drwxr-xr-x 2 omar omar 4,0K ene 11 06:45 Música
drwxr-xr-x 2 omar omar 4,0K ene 11 06:45 Imágenes
drwxr-xr-x 2 omar omar 4,0K ene 11 06:45 Documentos
drwxr-xr-x 2 omar omar 4,0K ene 11 06:45 Descargas
drwxrwxr-x 6 omar omar 4,0K ene 11 21:26 gcc-arm-none-eabi-4_7-2013q3
drwxr-xr-x 2 omar omar 4,0K ene 12 21:47 Escritorio
drwxr-xr-x 9 root root 4,0K ene 12 21:51 paparazzi
root@omar-HP-455-Notebook-PC:/home/omar#
```

Entramos a la carpeta paparazzi con **cd paparazzi/** y una vez adentro ingresamos el comando **make** y comienza a cargar paquetes nuevos como se muestra en la imagen.

```
root@omar-HP-455-Notebook-PC: /home/omar/paparazzi
BUILD test_stick
CC sdl_stick.o
CC ml_sdl_stick.o
BUILD sdl_stick.so
OC input2ivy.ml
OL input2ivy
GENERATE /root/paparazzi/var/include/messages.h
GENERATE /root/paparazzi/var/include/ubx_protocol.h
GENERATE /root/paparazzi/var/include/mtk_protocol.h
GENERATE /root/paparazzi/var/include/xsens_protocol.h
GENERATE /root/paparazzi/var/include/dl_protocol.h
GENERATE /root/paparazzi/var/include/abi_messages.h
make -C sw/ext
```

Inicio el software desde el directorio de paparazzi. Ejecutando el siguiente comando en el terminal lanzamos el aplicativo automáticamente.

./paparazzi



Luego de ya instalado procedemos con la actualización de paquete de inventario y las principales dependencias de software Paparazzi. Esto tomará algún tiempo y básicamente es la ejecución de estos dos comandos en el terminal de Ubuntu.

sudo apt-get update

sudo apt-get install paparazzi-dev

ANEXO B

ESPECIFICACIONES TÉCNICAS DEL DRONE



Sensor de ultrasonidos para medición de altitud respecto al suelo.

Cámara QVGA vertical a 60 fps para medición de la velocidad respecto al suelo.

4 Tubos de fibra de carbono: peso total de 380 g con el casco de protección para el exterior, 420 g con el casco de protección para el interior.

5. Piezas de plástico nylon cargado con 30% de fibra de vidrio de alta calidad.

6. Espuma para aislar el centro de inercia de las vibraciones de los motores.

7. Casco de protección de polipropileno expandido (PPE) inyectado por un molde metálico

8. Nano revestimiento repelente a los líquidos en los sensores de ultrasonidos.

Reparable en su totalidad: todas las piezas e instrucciones de reparación disponibles en Internet.

9. Cuatro motores de rotor interno (“inrunner”) sin escobillas. 14,5 W y 28,5 cuando queda suspendido en el aire.
10. Rodamiento de bolas en miniatura.
11. Engranajes de Nylatron de bajo Ruido para reductor de hélice de 1/8,75.
12. Eje de las hélices de acero templado.
13. Cojinete de bronce auto lubricante.
14. Alta fuerza de propulsión de las hélices para mayor maniobrabilidad.
15. Microcontrolador AVR de 8 MIPS por controlador de motor.
16. Batería recargable Li-Po de 3 elementos, 1.000 mAh.; Parada de emergencia controlada por software Controlador del motor totalmente reprogramable; Controlador electrónico del motor resistente al agua.
17. Almacenamiento de vídeo sobre la marcha con Wi-Fi directamente en tu dispositivo remoto o en una unidad de memoria USB.

Estructura robusta

Intentar los trucos más audaces nunca supondrá un reto para este diseño de última generación, fabricado para durar mucho tiempo.

- Tubos de fibra de carbono: Peso total 380 g. con carcasa de exterior, 420 g con carcasa de interior.
- Fibra de alta calidad (30%) cargada con piezas plásticas de nylon.
- Espuma para aislar el centro inercial de las vibraciones del motor.
- Casco EPP inyectado por un molde de metal sinternizado.
- Nanorevestimiento que repele los líquidos en los sensores de ultrasonidos.
- Completamente reparable: Todas las piezas e instrucciones para la reparación están disponibles en Internet.

Motores

Vuela alto. Vuela rápido. Lejos del suelo.

- 4 motores “inrunner” sin escobillas. 14,5W 28.500 RPM.
- 1 Cojinete de microbola.

- 1 Engranajes de Nylatron de bajo ruido para 1 reductor de propulsión 8.75.
- 1 Eje de transmisión de acero templado.
- 1 Cojinete de bronce autolubricante.
- 1 Resistencia aerodinámica específica de alta propulsión para ofrecer una excelente maniobrabilidad.
- 1 CPU AVR de 8 MIPS por controlador de motor.
- Parada de emergencia controlada por software.
- Controlador de motor totalmente reprogramable.
- Controlador electrónico del motor.

Asistencia electrónica

La tecnología incorporada en el AR.Drone 2.0 te ofrece funciones de control de extrema precisión y estabilización automática.

- Procesador de 1 GHz y 32 bit ARM Cortex A8 con vídeo DSP. TMS320DMC64x de 800 MHz.
- Linux 2.6.32.
- RAM DDR2 de 1GB a 200MHz.
- USB 2.0 de alta velocidad para extensiones.
- Wi-Fi b g n.
- Giroscopio de 3 ejes con una precisión de 2000°/seg.
- Acelerómetro de 3 ejes con una precisión de +/- 50mg.
- Magnetómetro de 3 ejes con una precisión de 6°.
- Sensor de presión con una precisión de +/- 10 Pa.
- Sensores de ultrasonido para medir la altitud de avance.
- Cámara vertical QVGA de 60 FPS para medir la velocidad de avance.