UNIVERSIDAD SAN FRANCISCO DE QUITO

Colegio de Postgrados

Adquisición de imágenes de bajo costo aplicadas a la agricultura de precisión usando vehículos aéreos no tripulados

Cesar Augusto Diaz Celis

Tesis de grado presentada como requisito
Para la obtención del título de
Magíster en Sistemas de Información Geográfica

Quito, Junio de 2013

Universidad San Francisco De Quito

Colegio de Postgrados

HOJA DE APROBACION DE TESIS

Adquisición de imágenes de bajo costo aplicadas a la agricultura de precisión usando vehículos aéreos no tripulados

Cesar Augusto Diaz Celis

Richard Resl. PhD(c).,
Director de Tesis
Director del Programa de Maestría en
Sistemas de Información Geográfica _____

Pablo Cabrera. MSc.,
Miembro del Comité de Tesis _____

Stella de la Torre. PhD.,
Decana del Colegio de
Ciencias Biológicas y Ambientales _____

Víctor Viteri Breedy. PhD.,
Decano del Colegio de Postgrados _____

Quito, Junio de 2013

© DERECHOS DE AUTOR

Por medio del presente documento certifico que he leído la Política de Propiedad Intelectual de la Universidad San Francisco de Quito y estoy de acuerdo con su contenido, por lo que los derechos de propiedad intelectual del presente trabajo de investigación quedan sujetos a lo dispuesto en la política.

Asimismo, autorizo a la USFQ para que realice la digitalización y publicación de este trabajo de investigación en el repositorio virtual, de conformidad a lo dispuesto en el Art.144 de la Ley Orgánica de Educación Superior.

Firma:

_____

Nombre: CESAR AUGUSTO DIAZ CELIS.

C. I.: 86070616 de Villavicencio – Meta – Colombia.

Fecha: 14 de Febrero de 2013.

DEDICATORIA

A todos aquellos que de una u otra manera

han creído en mí y han ayudado a formar

la persona que soy, gracias.

Cesar Augusto Diaz Celis

AGRADECIMIENTOS

El autor agradece especialmente a:

A mis padres y hermanos porque siempre han creído en mí y por su apoyo incondicional.

Al Ingeniero Cesar Augusto Romero Molano por su amistad, acompañamiento e ideas.

Al MSc. Pablo Cabrera por su acompañamiento y juicios.

A la universidad San Francisco de Quito y al programa Internacional UNIGIS por darme la oportunidad de aprender más y de poder mostrar mis capacidades como profesional.

A la Universidad de los Llanos, al Instituto de Investigaciones de la Orinoquia Colombiana - IIOC y al grupo de investigación en tecnologías abiertas GITECX por la financiación y apoyo para el desarrollo de la investigación.

RESUMEN

La presente investigación muestra como el software y hardware adaptado y desarrollado permite la captura de imágenes aéreas de bajo costo usando un vehículo aéreo no tripulado comercial, esto con el fin de bajar los costos de la obtención de las imágenes.

Para lograr este resultado se definió una metodología con 9 fases que son: 1) la revisión bibliográfica, 2) la adaptación del cuadricoptero, 3) la captura y transmisión de datos GPS, 4) la captura y transmisión de imágenes aéreas, 5) la teleoperación; 6) el diseño de la interfaz gráfica de usuario para el control, la visualización y captura de las imágenes aéreas, 7) las pruebas, depuración y ajustes, 8) el análisis de resultados y 9) las conclusiones y trabajos futuros.

La totalidad del software y parte del hardware se desarrolló con herramientas Open Source, adicional a la captura de imágenes también se captura la ubicación geográfica en WGS84 de la captura de la imagen, esto con el fin de ayudar en la georeferenciación de las mismas, para la conversión de las coordenadas geográficas (WGS84) a coordenadas proyectadas (UTM) se utilizaron las ecuaciones de COTICCHIA-SURACE.

La principal conclusión de la investigación es que se pueden obtener imágenes aéreas de bajo costo, lo cual es de gran utilidad para las pequeñas y grandes empresas ya que contarían con imágenes en tiempo real con una buena resolución.

ABSTRACT

This research shows how the software and hardware adapted and developed imaging enables low-cost carriers using a commercial unmanned aerial vehicle, this in order to lower the costs of obtaining the images.

To achieve this result was defined a methodology to 9 phases which are: 1) literature review, 2) the adaptation of the quadricopter, 3) capture and GPS data, 4) the capture and transmission of aerial imagery, 5) teleoperation, 6) the design of graphical user interface for control, visualization and capture aerial images, 7) testing, debugging and adjustments, 8) the analysis of results and 9) the conclusions and future work.

The entire part of the hardware and software developed with open source tools, in addition to capturing images is also captured in WGS84 geographic location of the image capture, this in order to assist in the georeferencing them to converting the coordinates (WGS84) a projected coordinates (UTM) were used COTICCHIA-SURACE equations.

The main conclusion of the research is that you can get low-cost aerial images, which is useful for small and large companies since they will have real-time images with good resolution.

TABLA DE CONTENIDO

LISTA DE IMÁGENES

LISTA DE TABLAS

1. INTRODUCCIÓN

1.1 PRESENTACION

El uso de UAV para la captura de imágenes aéreas presenta ventajas como son: el bajo costo en la obtención de imágenes aéreas, obtención de imágenes de alta definición (esto gracias a la baja altura de vuelo y a la alta definición de las cámaras digitales que se encuentran en el mercado), vehículos fácilmente transportables, la captura de imágenes es inmediata, la recolección de la información es casi en tiempo real, entre otros.

La tecnología UAV es un complemento a los sistemas tradicionales para la captura de información del territorio, con un alto nivel de detalle a bajo costo, hoy en día estos vehículos se están usando como herramienta de captura de información en la AP, entre sus aplicaciones encontramos: la toma de imágenes aéreas las cuales permiten a los productores tomar decisiones fundamentadas que pueden mejorar el ahorro de insumos, dinero y afectando menos el ambiente natural [1]; captura de imágenes aéreas con sensores infrarrojos, luz visible, térmicas, entre otros; imágenes que se combinan con gráficos los cuales muestran como los cultivos están creciendo e identificación de las zonas de deterioro; las imágenes capturadas también se utilizan para medir los niveles de humedad en el suelo; los problemas causados por la sobre fertilización, los animales de pastoreo o las plagas; entre otros.

El creciente desarrollo de los UAV, ha hecho posible su uso para transportar sensores de teledetección, los cuales presentan las mismas ventajas que los

sistemas aéreos tripulados, añadiendo otras ventajas como son [2]: el ser operados durante intervalos de tiempo extensos, vuelo de forma autónoma, el costo de explotación y desarrollo es menor que el de un avión tripulado.

De allí la necesidad de llevar a cabo el desarrollo de este proyecto de investigación ya que la región contaría con una herramienta de bajo costo para la adquisición de imágenes aéreas, las cuales se aplicarían en la AP, esto ayudaría de gran manera a la productividad agrícola, en una región como el departamento del Meta donde se explotan grandes cultivos como son: la palma de aceite con más de 130.000 Ha primordialmente para la generación de biocombustibles[1]; el arroz con un área cosechada de 80.700 Ha en el año 2010, siendo el departamento con la mayor participación del área cosechada nacional[2]; el caucho con 638 Ha sembradas en el año 2003 y el segundo departamento con mayor área sembrada[3]; caña de azúcar; yuca; entre otros.

El objetivo del gobierno colombiano es convertir a Colombia en una potencia de biocombustibles, donde los llanos orientales, en especial el departamento del Meta tendría mucho que ver, empresas como: GPC Etanol ubicada en Puerto López, con una planta de producción de alcohol carburante a partir de yuca con capacidad de 25.000 lts/día[4]; el proyecto de BIONERGY participada en un 85% por ECOPETROL, para la construcción de un complejo industrial el cual incluye una planta de bioetanol a partir de caña de azúcar, con una capacidad de

[1] Periódico el Tiempo – 30 de Septiembre de 2009 - http://www.eltiempo.com/archivo/documento/CMS-6245190
[2] DANE – Encuesta nacional de arroz mecanizado II semestre 2010
http://www.dane.gov.co/files/investigaciones/boletines/arroz/bol_arroz_IIsem10.pdf
[3] Secretaria de planeación departamental de Cundinamarca – 2005
http://www.planeacion.cundinamarca.gov.co/BancoMedios/Documentos%20PDF/el%20caucho.pdf
[4] http://www.gpc.com.co/index.php?option=com_content&view=article&id=44&Itemid=48 consultado el 4 de Julio de 2011.

procesamiento de 2.100.000 toneladas de caña/año, este proyecto se construye entre Puerto López y Puerto Gaitán[5]; Manuelita con la producción de 100 millones de litros anuales de biodisel de palma[6]. Todas estas empresas están trabajando en la producción de etanol y biodisel a base de cultivos de palma, yuca, jatropha, entre otros productos, esto enmarcado en la política nacional de biocombustibles[7].

A nivel nacional organizaciones como: Fedepalma[8], Uniban[9], Cenibanano[10], Cenicaña[11], Riopaila Castilla[12], entre otras usan la AP para mejorar la explotación de sus cultivos, el proyecto aquí planteado sería una herramienta de gran utilidad para las empresas regionales y nacionales disminuyendo el costo por captura de información de imágenes aéreas, no solo en costo si no en el tiempo de captura ya que es casi en tiempo real y mejorando la producción agrícola apoyada en la AP la cual se beneficiaría por la adquisición de las imágenes aéreas las cuales serían muy útiles en la explotación sostenible de los cultivos.

El problema es como disminuir los tiempos y costos en la captura de imágenes aéreas para ser aplicadas en la AP, esta investigación no solo le aportaría a los grandes agricultores del departamento del Meta ya que también beneficiaria al pequeño agricultor.

[5] http://www.isoluxcorsan.com/es/isolux-corsan-en-el-mundo/america/colombia/ consultado el 15 de Julio de 2011.
[6] http://www.manuelita.com/index.php?p=productos/energiarenovable& consultado el 16 de Julio de 2011.
[7] Periódico el Espectador – 01 de Marzo de 2009- http://www.elespectador.com/impreso/biocombustibles/articuloimpreso122880-el-llano-polo-de-biocombustibles
[8] http://www.fedepalma.org/palmas.shtm#1
[9] http://www.uniban.com/home_espanol.htm
[10] http://www.cenired.org.co/?q=investigacion/cenibanano
[11] http://www.cenicana.org/publicaciones/carta_trimestral/carta_trimestral.php?opcion=6&menu=1
[12] http://www.riopaila-castilla.com/index.php?option=com_content&view=article&id=33&Itemid=38

1.2 OBJETIVOS

1.2.1 Objetivo general

Implementar un sistema de captura de imágenes aéreas de bajo costo para ser aplicadas a la agricultura de precisión usando un vehículo aéreo no tripulado.

1.2.2 Objetivos específicos

- Analizar el modo en el que se capturan las imágenes aéreas tanto para aviones tripulados y no tripulados.

- Elaborar un módulo para captura y transmisión de imágenes aéreas, basado en protocolos de comunicación inalámbrica.

- Desarrollar un módulo de captura y transmisión de datos GPS, basados en protocolos de comunicación inalámbrica.

- Controlar remotamente el vehículo aéreo no tripulado.

- Desarrollar una aplicación de escritorio para el control del vehículo aéreo no tripulado y la visualización de las imágenes aéreas.

2. REVISIÓN DE LITERATURA

2.1 VEHICULOS AEREOS NO TRIPULADOS

Son aquellos que realizan una misión o actividad sin tener tripulación a bordo, ya que el control del mismo se realiza desde tierra ya sea por medio de un controlador o de forma autónoma. Los UAVs hoy en día no solo se utilizan en el sector defensa también son usados en aplicaciones civiles como el caso de la agricultura y el medio ambiente, el desarrollo de esta tecnología ha aumentado exponencialmente en los últimos años en todo el mundo.

2.1.1 Ventajas de los UAVs

Dentro de las ventajas de los UAVs encontramos:

- No se arriesgan vidas humanas.
- Útil para aquellas zonas de difícil acceso geográfico, orden público, volcanes, incendios, concentración de radioactividad, entre otros.
- Fotografías aéreas de alta resolución.
- Disponibilidad de las fotografías aéreas en tiempo real.
- No presenta problemas por las condiciones atmosféricas (nubosidad), ya que la altura de vuelo puede ser por debajo de las nubes.
- Bajo costo en la adquisición de imágenes aéreas.
- Adquisición de imágenes aéreas en cualquier momento lo cual permite la realización de estudios multitemporales.

2.1.2 Desventajas de los UAVs

Dentro de las desventajas de los UAVs encontramos:

- Dependen de una estación en tierra.
- Vulnerabilidad.
- Limitaciones de peso de carga.
- Dificultad de integración en el espacio aéreo.

En la tabla 1 se hace una comparación entre las aeronaves tripuladas y no tripuladas.

Característica	UAV	Avión
Permanencia / autonomía	Media	Baja
Velocidad	Baja	Alta
Alcance	Media	Baja
Penetración	Alta	Alta
Maniobrabilidad	Alta	Media
Precisión	Media	Baja
Capacidad de respuesta	Media	Alta
Previsibilidad	Media	Baja
Autonomía	Media	Baja
Restricción de uso	Alta	Media
Coste de posesión	Alta	Media
Factor humano	Media	Alta
Polivalencia	Alta	Alta

Tabla 1. **Beneficio de los UAV frente a los aviones tripulados. Adaptado de Fuente:** [3].

2.1.3 Cuadricoptero

Es un UAV que puede despegar y aterrizar de forma vertical, facilitando la navegación en interiores, de la misma manera que lo hace un helicóptero, la diferencia principal es que su estructura está formada por cuatro motores que le permiten sostenerse en el aire.

Estos vehículos aéreos no tripulados poseen tres grados de libertad angular o también conocidos como ángulos de navegación que son: roll, pitch y yaw (ver imagen 1).

El movimiento de guiñada o yaw se presenta cuando el cuadricoptero gira sobre su eje vertical, esto se logra aumentando o disminuyendo en la misma proporción la potencia de giro de los rotores 1 y 3 y disminuyendo o aumentando en igual potencia de giro los motores 2 y 4 (ver imagen 2).

El movimiento de inclinación o pitch es el que permite que el cuadricoptero se desplace hacia adelante o hacia atrás, para ello el cuadricoptero requiere de aumentar la potencia en el rotor 1 que es opuesto al sentido deseado y disminuyendo en la misma magnitud el rotor 2, dejando el rotor 3 y 4 a potencia media (ver imagen 3).

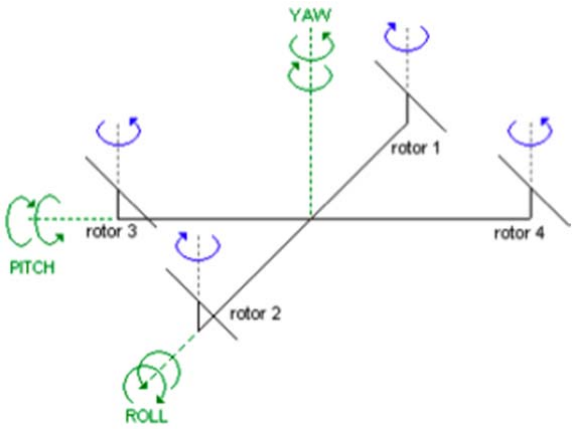

Imagen 1. Grados de libertad angular o ángulos de navegación.
Fuente: [4].

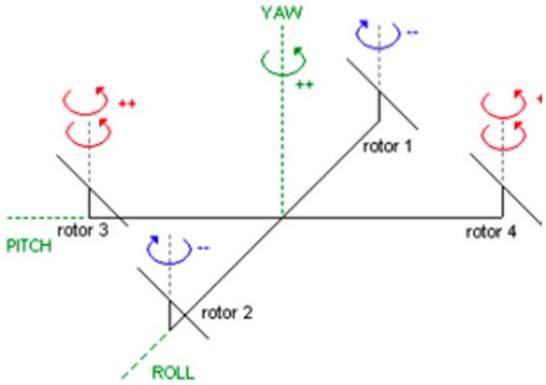

Imagen 2. Movimiento de guiñada o yaw.
Fuente: [4].

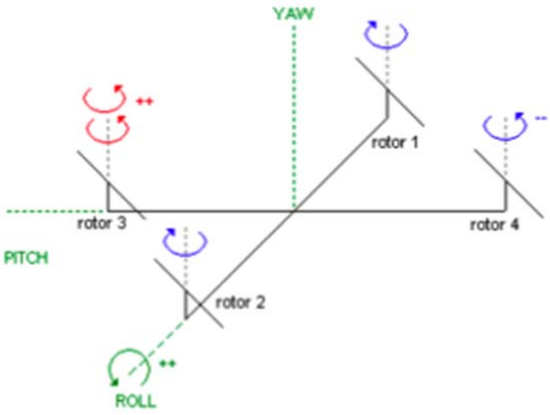

Imagen 3. Movimiento de inclinación o pitch.
Fuente: [4].

El movimiento de bamboleo o roll es cuando el cuadricoptero rota a la izquierda o a la derecha, usa el mismo principio de inclinación o pitch pero de forma lateral (ver imagen 4).

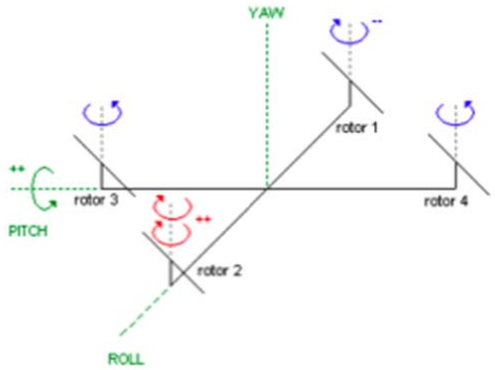

Imagen 4. Movimiento de bamboleo o roll.
Fuente: [4].

El movimiento vertical (ascender o descender) o mantenerse estacionario se logra generando con los cuatro rotores una fuerza mayor (ascender) o menor (descender) que la generada por la fuerza de gravedad (ver imagen 5).

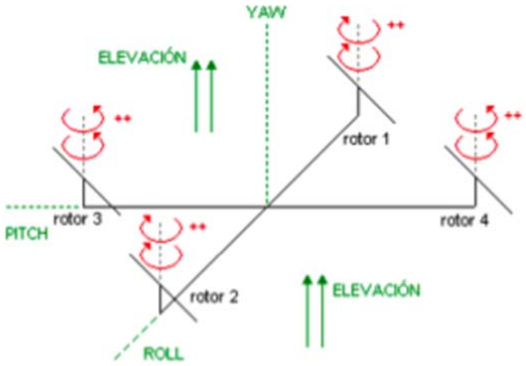

Imagen 5. Movimiento vertical (ascender, descender o estacionario).
Fuente: [4].

2.3 GPS

El sistema de posicionamiento global es un sistema de navegación por satélite que permite calcular la velocidad y posición de cualquier punto de la superficie terrestre a partir de la recepción de señales emitidas desde una constelación de

32 satélites en 6 planos orbitales [5] (ver imagen 6), el sistema GPS es controlado por el sistema de defensa de los Estados Unidos de América.

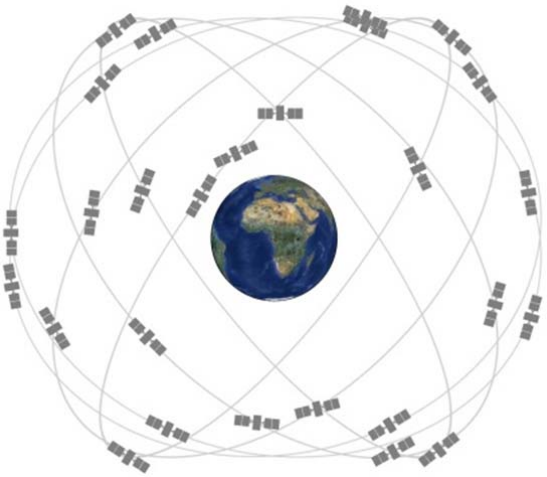

Imagen 6. Constelación de satélites sistema de posicionamiento global.
Fuente: [6].

2.4 AGRICULTURA DE PRECISIÓN (AP)

La AP busca la optimización del proceso productivo a partir del manejo de la variabilidad del agro ecosistema, esta nace del pobre entendimiento del proceso de producción que da el manejo tradicional de la agricultura el cual se basa en la generalización y en los promedios, esto resulta costoso y es causante de impactos ambientales negativos [7].

La AP está fundamentada en la variabilidad que se encuentra en todo el proceso productivo, la variabilidad se clasifica en [8]:

- **Natural:** Se debe a procesos de la naturaleza, como en el caso de los diferentes tipos de suelos, clima, topografía, especies vegetales y especies animales.

- **Inducida:** se debe a prácticas antrópicas, como en el caso de los diferentes niveles de conservación de los suelos, la microtopografía de un terreno agrícola, los tipos e intensidades de compactación. En algunas ocasiones esta diferenciación no es muy clara y la variabilidad ocurre debido a interacciones complejas entre procesos naturales y antrópicos.

- **Espacial:** puede ocurrir dentro de una finca o dentro de un mismo lote, un ejemplo de ello es diferentes contenidos de un nutriente en el suelo genera diferentes rendimientos en un cultivo.

- **Temporal:** se pueden presentar cambios temporales los cuales pueden afectar la producción, ejemplo de ello es el clima, la humedad del suelo, entre otros.

La AP pretende dar respuesta a los problemas importantes de la agricultura extensiva de hoy en día, como son la competitividad y el medio ambiente. De igual manera disponer de datos precisos tanto ecológicos, biológicos, tecnológicos y económicos que representen un agrosistema, posteriormente integrarlos en un marco conceptual que los formalice y relacione, con modelos conceptuales, para poderlos procesar mediante modelos de simulación y sistemas de información, para seleccionar las mejores alternativas de manejo, organización o comercialización a partir de datos, que aprueban transmitir la información de manera adecuada y que permita tomar decisiones en busca de la productividad y competitividad del sector agropecuario. En pocas palabras tener un conocimiento

y control de procesos empresariales mucho más ajustados a las necesidades reales del productor [9].

La utilización de esta tecnología permite la detección de variedad de ambientes en un mismo terreno, cuantificar las diferencias de productividad, así mismo diferenciar entre ambientes de rendimientos alcanzables estabilidad/riesgos, implicaciones sobre modelos de fertilización, densidad, plagas, malezas y la oportunidad de mejorar en los márgenes y sostenibilidad, para poner a prueba y mejorar los modelos utilizados para tal fin [9].

El termino AP hace referencia al manejo de las superficies agrícolas de forma individualizada y especifica. En pocas palabras es hacer lo correcto en el lugar adecuado y en el momento preciso, de forma automática, utilizando sensores, ordenadores y otros equipos electrónicos [10].

Los agentes involucrados en el desarrollo y adopción de las prácticas de la agricultura de precisión suelen dividir este conjunto de tecnologías en tres etapas diferentes, que son: 1) recolección de datos, 2) procesamiento e interpretación de la información y 3) aplicación de insumos [11] (Ver imagen 7).

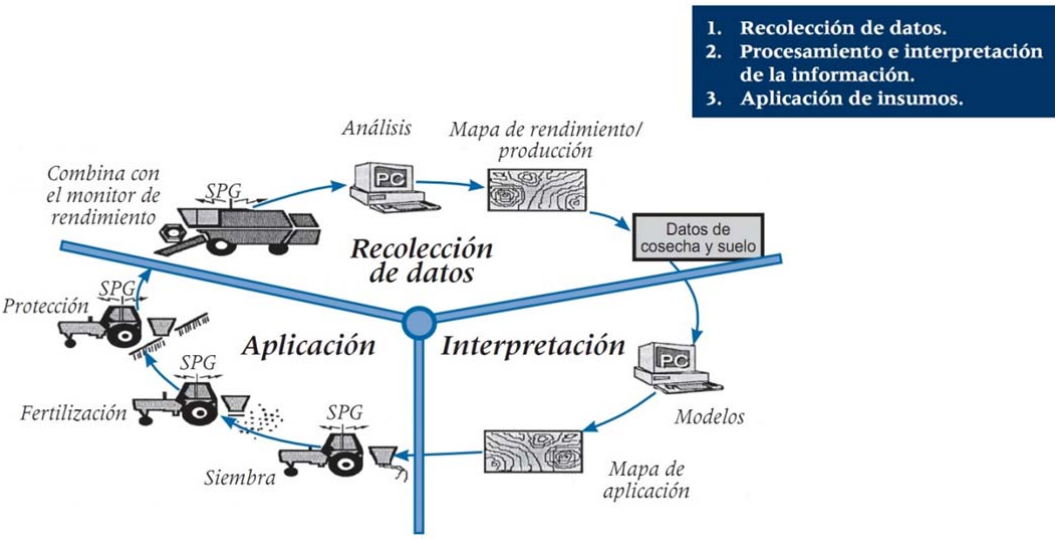

Imagen 7. Tecnologías agricultura de precisión.
Fuente: [11].

La necesidad de hacer competitivo el sector agropecuario dio paso a la utilización de este sistema alternativo sostenible, en Colombia se vienen adelantando experiencias en agricultura de precisión por parte de algunos productores comerciales con cultivos tecnificados como se puede apreciar en cultivos de palma de aceite (FEDEPALMA[13]) en donde han realizado delimitaciones de unidades de manejo agronómico, determinado el patrón de distribución espacial de enfermedades como marchites letal y anillo rojo y su relación con factores ambientales y nutricionales; experiencias similares se han obtenido en los sectores bananero (UNIBAN[14], CENIBANANO[15]) en plagas y enfermedades y el sector cañicultor (CENICAÑA[16], RIOPAILA CASTILLA[17]) en lo referente a suelos, en la facultad de Agronomía de la Universidad Nacional de Colombia, Bogotá, se han realizado investigaciones puntuales en manejo de plagas y suelos pero igualmente se han generado mejores y más confiables estadísticas

[13] http://www.fedepalma.org/palmas.shtm#1
[14] http://www.uniban.com/home_espanol.htm
[15] http://www.cenired.org.co/?q=investigacion/cenibanano
[16] http://www.cenicana.org/publicaciones/carta_trimestral/carta_trimestral.php?opcion=6&menu=1
[17] http://www.riopaila-castilla.com/index.php?option=com_content&view=article&id=33&Itemid=38

agropecuarias, mejor vigilancia epidemiológica que permite finalmente controlar y erradicar brotes de enfermedades con la georreferenciación de predios afectados, entre otras experiencias en el sector agropecuario colombiano [9, 12].

El área de mayor desarrollo dentro de la AP es el manejo de nutrientes sitio – especifico (MNSE), también llamado tecnología de dosis variables (TDV), el cual corresponde a la aplicación variable de dosis de fertilizantes de acuerdo al nivel de fertilidad de cada zona de manejo homogéneo dentro del lote, lo cual significa que no se trabaja, con una sola dosis de fertilizante, sino que con tantas dosis como áreas significativamente homogéneas existan en explotación [13].

Sin embargo en Colombia se presentan algunas barreras en el desarrollo y la utilización de la agricultura de precisión, ya que no existen metodologías para la recolección y manejo de información ajustadas a nuestra realidad, la falta de adaptación de la propia tecnología, y el bajo nivel de gestión y de capacidad administrativa de un sector importante de nuestros productores agrícolas.

Una tecnología como la descrita está lejos de ser una opción válida para nuestros agricultores, tanto por costos (a pesar de que el costo de inversión está disminuyendo a medida que aumenta la adopción de la misma), como por su adaptabilidad a nuestro medio. Se hace necesario para el país investigar el potencial de la agricultura de precisión, con diferentes niveles de desarrollo tecnológico, así como su aplicación en los distintos sistemas productivos y áreas del conocimiento agronómico. Esto implica evaluar y adaptar metodologías y tecnologías (hardware y software) aplicables a nuestra realidad agraria, que den

viabilidad a la implementación de la AP como una opción importante para el desarrollo sostenible y competitivo de la agricultura nacional [12].

2.5 IMÁGENES AEREAS APLICADAS EN LA AGRICULTURA DE PRECISIÓN

La AP pretende dar respuesta a los problemas importantes de la agricultura extensiva de hoy en día, como son la competitividad y el medio ambiente. De igual manera disponer de datos precisos tanto ecológicos, biológicos, tecnológicos y económicos que representen un agrosistema, posteriormente integrarlos en un marco conceptual que los formalice y relacione, con modelos conceptuales, para poderlos procesar mediante modelos de simulación y sistemas de información, para seleccionar las mejores alternativas de manejo, organización o comercialización a partir de datos, que aprueban transmitir la información de manera adecuada y que permita tomar decisiones en busca de la productividad y competitividad del sector agropecuario. En pocas palabras tener un conocimiento y control de procesos empresariales mucho más ajustados a las necesidades reales del productor [14].

Los UAV por su parte, son aeronaves motorizadas capaces de operar por medio de controladores situados en tierra o en el aire, sin que sea requerida la figura del piloto humano que gobierne sus mandos [15].

La obtención de imágenes aéreas bien sea por medio de un vehículo tripulado o un UAV tienen múltiples aplicaciones dentro del campo de la AP como son: obtención de imágenes en la banda visible, infrarrojo cercano y térmico para

estimar parámetros biofísicos de los cultivos: concentración de clorofila, xantofila, carotenoides y antocianinas [16]; determinación de niveles de stress en los cultivos [17]; determinación de zonas con índice de vegetación de diferencia normalizada (NDVI) [18]; determinación de zonas con diferente índice verde y su correlación con la aplicación diferenciada de fertilizante [19]; evaluación de daños en cultivos causados por plagas o enfermedades; determinación del crecimiento de la biomasa y la calidad de los cultivos [20]; análisis de estrés en cultivos; análisis de cultivos usando imágenes térmicas; agricultura de bajo costo [21]; entre otras.

Estas imágenes pueden ser capturadas desde un UAV y de esta manera obtener imágenes mucho más rápido, de menor costo y mayor precisión [21], lo cual sería un gran aporte a la toma de decisiones en la AP.

Los UAV también tienen aplicaciones no relacionadas con la AP como son: detección de zonas contaminadas; generación de mapas prediales o catastrales; Inspección de infraestructura (líneas eléctricas, oleoductos, gasoductos, entre otros); vigilancia de fronteras, patrulla marítima (inmigración ilegal, contrabando); búsqueda y rescate (naufragios, accidentes en montaña o zonas de difícil acceso, amenazas naturales [22]); reconocimiento y toma de datos en desastres naturales (huracanas, volcanes, terremotos); climatología (toma de muestras y monitorización de partículas en aerosol, monitorización de contaminación atmosférica); localización de recursos naturales (pesca [23], minería); seguimiento de especies protegidas; arqueología [24]; entre otras.

3. METODOLOGÍA

La metodología utilizada en la investigación consta de nueve fases las cuales se explican a continuación:

3.1 FASE 1: REVISIÓN BIBLIOGRÁFICA, CONTEXTUALIZACIÓN Y ANÁLISIS DETALLADO

Se realizó una búsqueda y revisión bibliográfica para el desarrollo de la investigación. Se identificaron libros, revistas y artículos con la finalidad de profundizar en los conceptos teóricos en temas como: UAV, AP, Software libre, GPS, redes inalámbricas, teleoperación, sensores para captura de imágenes aéreas.

3.2 FASE 2: ADAPTACIÓN DE UN CUADRICOPTERO (ARDRONE)

Para lo cual se realizó un cuidadoso análisis, así mismo una selección de componentes, identificación de piezas, accionamientos mecánicos y estructurales a utilizar para el mejoramiento, luego se procedió a seleccionar e instalar sensores y acondicionadores de señales. La ejecución de esta etapa permitió contar con un cuadricoptero prototipo útil como herramienta de desarrollo durante la investigación.

3.3 FASE 3: IMPLEMENTACIÓN DE UN MÓDULO DE CAPTURA Y TRANSMISIÓN DE DATOS GPS

Se realizó un cuidadoso análisis, así mismo una selección de componentes. Se identificaron piezas, accionamientos mecánicos y estructurales a utilizar en el montaje.

Para esto se realizó una investigación en el mercado de los GPS para desarrollo y los módulos de transmisión inalámbricos estos últimos encargados de enviar los datos del GPS el cual se encuentra montado en el cuadricoptero (ArDrone) a la estación base para que procese esta información.

3.4 FASE 4: SENSOR PARA CAPTURA Y TRANSMISIÓN DE IMÁGENES AÉREAS

Se investigaron los diferentes sensores para captura y transmisión de imágenes aéreas, analizando características como: tamaño, peso, resolución, consumo, capacidad de almacenamiento, entre otros.

3.5 FASE 5: MEJORAR LA COBERTURA DE TELEOPERACIÓN

Se investigaron los diferentes módulos de transmisión de datos como son: los módulos Xbee, módulos wifly, tarjetas de red inalámbricas, Nanostation, entre otros, esto con el fin de analizar características como: tamaño, peso, alcance, consumo, línea de visión, entre otros.

3.6 FASE 6: DISEÑO DE INTERFAZ GRÁFICA DE USUARIO (GUI) PARA EL CONTROL, LA VISUALIZACIÓN Y CAPTURA DE LAS IMÁGENES AÉREAS.

Se realizó una comparación de lenguajes de programación para el desarrollo de los diferentes módulos de software (control del cuadricoptero, procesamiento de imágenes aéreas (captura, transmisión, almacenaje y visualización), entre otros), manejo de GPS, todo orientado al desarrollo de aplicaciones Open Source.

En esta fase también Se llevó a cabo el análisis a herramientas existentes, argumentando sus ventajas y desventajas, lo cual sirvió de retroalimentación para la adaptación del SDK 2.0 del cuadricoptero (ArDrone).

3.7 FASE 7: PRUEBAS, DEPURACIÓN Y AJUSTES

En esta etapa se realizaron las pruebas a todos los módulos ya montados sobre el cuadricoptero (ArDrone) y en campo, esto con el fin de analizar el comportamiento de los módulos ante condiciones del mundo real como son: el viento, la temperatura, la humedad entre otros.

También se realizaron pruebas orientadas a determinar falencias sobre la lógica interna del software (aplicación). Esta fase se realizó en paralelo a todas las fases anteriores, con la finalidad de ir retroalimentando la solución final de cada fase.

3.8 FASE 8: ANÁLISIS DE RESULTADOS

Se realizó el análisis de los resultados generados por cada uno de los módulos, contrastándolos con resultados generados por soluciones comerciales, comparando tiempos de respuesta, calidad de la transmisión de datos, calidad de las imágenes capturadas, costos, aplicaciones posibles, clientes potenciales, población beneficiada, entre otros.

3.9 FASE 9: INFORME FINAL, CONCLUSIONES, TRABAJOS FUTUROS

Se generaron las conclusiones, los posibles trabajos futuros y el informe final de toda la investigación.

4. MATERIALES

4.1 EQUIPOS

El desarrollo de la investigación fue financiada por el Instituto de Investigación de la Orinoquia Colombiana - IIOC y la Universidad de los llanos, con el apoyo del grupo de Investigación en Tecnologías Abiertas – GITECX.

4.1.1 Dispositivo Móvil

El dispositivo móvil o equipo estación base utilizado en la investigación es un Ultrabook Samsung Serie 5 [25] (ver imagen 8), las especificaciones técnicas son:

- SO: Windows 7 de 64 Bits.
- Procesador: Intel Core I5 1.7 GHz.
- Pantalla: 14” HD LED.
- Memoria: DDR3 6 GB a 1600 MHz.
- Disco Duro: Sata II 1 TB a 5400 RPM, ExpressCache de 24 GB.
- Intel® Centrino® Advanced-N 6235, 2 x 2 802.11abg/n (hasta 300 Mbps), soporte para Widi.
- Bluetooth V4.0.
- Gigabit Ethernet.

Imagen 8. Ultrabook Samsung Serie 5.
Fuente: [25].

4.1.2 GPS

El modulo receptor utilizado es el GPS A2100A [26] (ver imagen 9), las especificaciones técnicas son:

- Módulo de recepción GPS basado en el integrado w/SIRF Star IV.
- Diseñado para aplicaciones en interiores.
- Soporte para sistema de aumentación basado en satélites SBAS, esto le permite un cálculo más preciso de la posición.
- SRAM.
- Posee 48 canales.
- Comunicación: UART-NMEA (default), UART -SiRF Specific SSB/OSP, SPI - NMEA/SiRF Specific (in preparation for A/B), I2C - NMEA/SiRF Specific (in preparation for B)
- Rata de baudios: 4800 (default).
- Sensitividad: -163 dBm.
- Voltaje de alimentación: 3.3V.
- Corriente de alimentación: 32mA.
- frecuencia: 1575 MHz.

- Temperatura de operación: -40 a +85ºC.
- Tamaño: 15mm x 15mm x 2,4mm
- Precisión: 2,5 metros en la horizontal.
- Velocidad de transmisión de trama: 115,2K.

Este módulo receptor se encuentra sobre la tarjeta GPS A2100A USB [27] (ver imagen 10), las especificaciones técnicas son:

- Tarjeta para adaptar el módulo GPS A2100A a montaje tipo board estándar.
- Módulo GPS A2100A.
- Interfaz USB con el FT232.
- Regulación a 3.3V.
- Leds de estado.
- Pulsador ON/OF.
- Pulsador Reset.
- puertos disponibles y conector SMA.
- Antena.

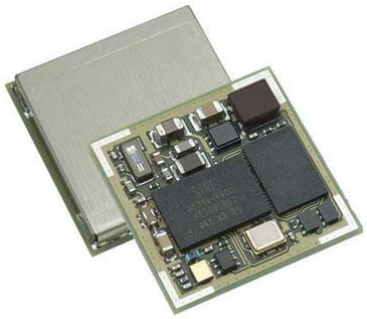

Imagen 9. GPS A2100A.
Fuente: [26].

Imagen 10. Tarjeta GPS A2100A USB.
Fuente: [27].

4.1.3 Cámara

El sensor óptico utilizado es la cámara para deportes extremos GoPro Hero 2 [28] (ver imagen 11), las especificaciones técnicas son:

- Óptica de la cámara:
    - Lente de Cristal Profesional 2x Más Nítido.
    - Foco fijo f/2.8.
    - Campo de Visión Panorámico de 170º (Incluyendo 1080p).
    - Campo de Visión Medio 127º (En 1080p).
    - Campo de Visión Angosto de 90º.
- Video:
    - RESOLUCIONES HD:
        - 1080p: 1920x1080, 30 FPS.
        - 960p: 1280x960, 48 FPS + 30 FPS.
        - 720p: 1280x720, 60 FPS + 30 FPS.
    - RESOLUCIONES DE DEFINICIÓN ESTÁNDAR:
        - WVGA: 848x480, 120 FPS + 60 FPS.
    - Sensor de Imagen CMOS 1 /2.3” CMOS de Alto Rendimiento.
    - Sensibilidad a la Luz: .84 V/lux-sec.
    - Formato de Vídeo: códec H.264, formato de archivo .mp4.
    - Control de Exposición: Centro de Promedio Ponderado Spot.
    - Balance de Blancos: Automático.
- Fotografía:
    - RESOLUCIONESS: 11MP, 8MP, 5 MP.
    - Campo de Visión: Panorámico 170º, Medio 127º.

- MODOS DE CAPTURA: Único, ráfaga de 10 fotos, Lapso de Tiempo, Auto disparador.

- Audio:
  - Mono, 48 kHz, Compresión AAC, Control Automático Ganancia.
  - Entrada de micrófono estéreo externo (3,5 mm).
- Almacenamiento máximo de 32 GB clase 4 o superior.

Imagen 11. GoPro Hero 2.
Fuente: [28].

4.1.4 Módulos de transmisión y recepción inalámbrica

El módulo que se usó para la transmisión de los datos del GPS fue el Xbee Pro serie 2B con antena en board PCB [29] (ver imagen 12), las especificaciones técnicas son:

- Módulo de comunicación ZigBee serie 2.
- 2.4GHz, 250 Kbps.
- Cumple con el estándar 802.15.4 Zigbee.
- Alcance de hasta 90 m en interiores y hasta 3200 m de alcance en exteriores con línea de vista.
- Antena tipo PCB, montaje superficial.

- Sensitividad: -102dBm.
- Potencia de transmisión: 63 mW (18dBm).
- Voltaje de operación: 2.7 -3.6VDC.

Imagen 12. **Xbee Pro serie 2B.**
Fuente: [29].

En el caso del módulo de transmisión de imágenes de la cámara a la estación base y la transmisión de órdenes a la cámara de la estación base se usó el Wi-Fi BacPac™ + Wi-Fi Kit Combinado Remoto [30] (ver imagen 13), las especificaciones técnicas son:

- Wi-Fi Remoto:
    - Controla hasta 50 cámaras al mismo tiempo con un alcance de 180 m en perfectas condiciones.
    - Control de la cámara total, incluido el encendido y el apagado, el control de la configuración y del obturador de disparo.
    - La pantalla LCD muestra lo mismo que la pantalla de estado LCD de la cámara.
    - Portátil y resistente al agua a una profundidad de 3 m.
- Wi-Fi BacPac:
    - Wi-Fi BacPac se instala en la cámara GoPro y permite controlarla desde una distancia de 180 m en condiciones óptimas.

- Conexión a tablets.
- Conexión a Smartphone.
- Conexión a equipo por medio de redes Ad Hoc.

Imagen 13. **Wi-Fi BacPac + Wi-Fi Kit Combinado Remoto.**
Fuente: [30].

4.1.5 Vehículo aéreo no tripulado

El vehículo aéreo no tripulado usado en la investigación es el Ar.Drone 1 de la empresa Parrot [31] (ver imagen 14), las especificaciones técnicas son:

- Dimensiones:
  - Con casco: 52,5 x 51,5 cm.
  - 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 máxima: 5 m/s, 18 km/h.
- Altura máxima: 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.

- Sistema informático integrado:
    - 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.
- Sistema de seguridad:
    - Casco para vuelo en interiores con cubierta para las hélices.
    - Bloqueo automático de las hélices en el caso de contacto.
    - Batería UL2054.
    - Interfaz de control con botón de emergencia para detener los motores.
- Estructura aerodinámica:
    - Cuatro hélices de alta eficiencia (especialmente diseñadas para el Parrot AR.Drone).
    - Estructura tubular de fibra de carbono.
- Motores y energía:
    - 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.

- Cámara frontal:
    - Sensor CMOS de tipo gran angular de lente diagonal. 93 grados de amplitud.
    - Resolución 640x480 píxeles (VGA).
    - Respuesta en Frecuencia: 15 FPS.
    - Codificación y transmisión en vivo de imágenes.
- Cámara vertical:
    - Sensor CMOS de alta velocidad de lente diagonal. 64° de amplitud.
    - Resolución 176x144 píxeles.
    - Respuesta en Frecuencia: 60 FPS.
    - Es utilizada como sensor de desplazamiento adicional, permite la estabilización del drone incluso con viento ligero.
- Altímetro por ultrasonido:
    - Frecuencia de emisión: 40 kHz.
    - Alcance de 6 metros utilizado para la estabilización vertical.

Imagen 14. **Ar Drone 1.**
Fuente: [31].

4.1.6 Arduino UNO R3

Se utilizó un microcontrolador ATmega para la interacción con la tarjeta GPS A2100A USB y el módulo Xbee, esto con el fin de procesar el protocolo NMEA, específicamente la trama GPGGA, para ello se utilizó el Arduino Uno R3 [32] (ver imagen 15), las especificaciones técnicas son:

- Microcontrolador ATmega 328.
- 14 pines digitales de entrada y salida.
- 6 pines de estos pueden ser usados como salidas PWM.
- 6 pines de entrada análogos.
- Cristal de 16 MHz.
- Conexión USB.
- Conexión de poder.
- Voltaje de operación: 5 voltios.
- Entrada de voltaje: 7 – 12 voltios.

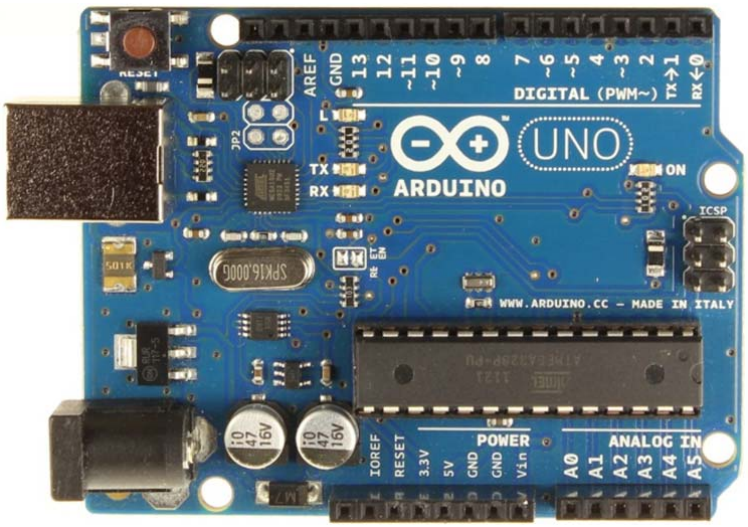

Imagen 15. **Arduino Uno R3.**
Fuente: [32].

4.2 SOFTWARE

4.2.1 Desarrollado o Adaptado en el Dispositivo Móvil

El dispositivo móvil cuenta con el sistema operativo Debian 6 Testing, librería de visión artificial OpenCV 2.4 [33 - 34], compilador GCC 4.7 y Eclipse Juno. Para el control del UAV se adaptó el SDK 2.0 [35] el cual es de código abierto, en total se crearon o adaptaron 7 clases en lenguaje C++ (ver tabla 2).

Clase	Descripción
GeograficasPlanas	Clase desarrollada para la conversión de coordenadas geográficas (WGS84) a coordenadas planas (UTM).
Archivos	Clase desarrollada para el almacenaje de datos en disco duro.
SerieLinux	Clase adaptada para la comunicación por puerto RS232, esta permite la comunicación entre el dispositivo móvil y el Explorer del Xbee.
StringMejorado	Clase desarrollada para el manejo de cadenas de caracteres.
DatosGPS	Clase desarrollada para recibir una trama GPGGA en protocolo NMEA e interpretarla, también recibe una trama creada ARDUINOGPS.
VisionArtificial	Clase desarrollada para la captura y visualización del flujo de video.
Xbee	Clase desarrollada que hereda de SerieLinux para el uso de módulos Xbee.

Tabla 2. **Clases desarrolladas o adaptadas en la aplicación del dispositivo móvil.**
Fuente: El autor.

Para el módulo de captura de datos GPS, se trabajó con el protocolo NMEA 0183 el cual es un medio que permite la comunicación entre instrumentos marítimos y la mayoría de receptores GPS, este protocolo es definido y controlado por la Organización Estadounidense National Marine Electronics Association [36]. La sentencia o trama que se trabajó en esta investigación es la GPGGA que hace

referencia a los datos del fijo del sistema de posicionamiento global (GPS) (ver tabla 3).

$GPGGA,161229.487,3723.2475,N,12158.3416,W,1,07,1.0,9.0,M, , , ,0000*18			
Nombre	**Ejemplo**	**Unidad**	**Descripción**
Message ID	$GPGGA		GGA protocol header
UTC Time	161229.487		hhmmss.sss
Latitude	3723.2475		ddmm.mmmm
N/S Indicator	N		N=North or S=south
Longitude	12158.3416		dddmm.mmmm
E/W Indicator	W		E=East or W=West
Position Fix Indicator	1		0=Fix not available or invalid; 1=GPS SPS Mode, fix valid; 2=Differential GPS, SPS Mode, fix valid; 3-5=Not supported; 6=Dead Reckoning Mode, fix valid
Satellites Used	07		Range 0 to 12
HDOP	1.0		Horizontal Dilution of Precision
MSL Altitude	9.0	meters	
Units	M	meters	
Geoid Separation		Meters	
Units	M	meters	
Age of Diff. Corr.		second	Null fields when DGPS is not used
Diff. Ref. Station ID	0000		
Checksum	*18		
<CR> <LF>			End of message termination

Tabla 3. **Formato de la trama GPGGA.**
Fuente: [37].

Como se observa la latitud y longitud está representada en grados decimales y así se transmite del UAV al dispositivo móvil, en este se realiza una conversión de coordenadas geográficas (WGS84) a universal transversal de mercator (UTM).

Partiendo de las coordenadas geográficas, donde la longitud es λ y la latitud es φ , conociendo el valor del semieje mayor (a)= 6378137, semieje menor (b)= 6356752.31424518 con estos datos se inician las operaciones para conversión de

coordenadas geográficas a UTM [38], lo primero a calcular es la geometría del elipsoide: excentricidad (1), segunda excentricidad (2), radio polar de curvatura (3) y el aplanamiento (4).

$$e = \sqrt{\frac{a^2 - b^2}{a}} \quad (1)$$

$$e' = \sqrt{\frac{a^2 - b^2}{b}} \quad (2)$$

$$c = \frac{a^2}{b} \quad (3)$$

$$\alpha = \frac{a - b}{a} \quad (4)$$

La longitud y latitud deben de estar expresada en grados decimales (5), luego se pasa de grados decimales a radianes (6).

$$GradosDecimales = grados + \frac{\min utos}{60} + \frac{segundos}{3600} \quad (5)$$

$$Radianes = \frac{GradosDecimales * \Pi}{180} \quad (6)$$

Para el cálculo del signo de la longitud si se encuentra al oeste del meridiano de Greenwich la longitud es negativa, si esta al este del meridiano de Greenwich, la longitud es positiva. Ahora se procede a calcular el huso o zona UTM (7), se toma la parte entera.

$$Huso = \frac{GradosDecimales}{6} + 31 \quad (7)$$

Ahora se obtiene el meridiano central del huso en el que se encuentran las coordenadas geográficas (8), luego se calcula la distancia angular que existe entre la longitud del punto y el meridiano central del huso (9).

$$\lambda_0 = Huso * 6 - 183 \quad (8)$$

$$\Delta\lambda = \lambda - \frac{\lambda_0 * \Pi}{180} \quad (9)$$

Para convertir de coordenadas geográficas a UTM se usan las ecuaciones de Coticchia – Surace, primero se calculan una serie de parámetros que se requieren para el uso de las ecuaciones (10 - 23).

$$A = \cos\varphi * sen\Delta\lambda \quad (10)$$

$$\xi = \frac{1}{2} * \ln\left[\frac{1+A}{1-A}\right] \quad (11)$$

$$\eta = \arctan\left(\frac{\tan\varphi}{\cos\Delta\lambda}\right) \quad (12)$$

$$\nu = \frac{c}{(1+e'^2 * \cos^2\varphi)^{1/2}} * 0.9996 \quad (13)$$

$$\zeta = \frac{e'^2}{2} * \xi^2 * \cos^2\varphi \quad (14)$$

$$A_1 = sen(2\varphi) \quad (15)$$

$$A_2 = A_1 * \cos^2\varphi \quad (16)$$

$$J_2 = \varphi + \frac{A_1}{2} \quad (17)$$

$$J_4 = \frac{3 * J_2 + A_2}{4} \quad (18)$$

$$J_6 = \frac{5 * J_4 + A_2 * \cos^2 \varphi}{3} \quad (19)$$

$$\alpha = \frac{3}{4} * e'^2 \quad (20)$$

$$\beta = \frac{5}{3} * \alpha^2 \quad (21)$$

$$\gamma = \frac{35}{27} * \alpha^3 \quad (22)$$

$$B_\phi = 0.9996 * c * (\varphi - \alpha J_2 + \beta J_4 - \gamma J_6) \quad (23)$$

Una vez calculados los parámetros anteriores se procede a resolver las ecuaciones de Coticchia – Surace (24 - 25) para obtener las coordenadas finales en UTM.

$$X = \xi * \nu * \left(1 + \frac{\zeta}{3}\right) + 500000 \quad (24)$$

$$Y = \eta * \nu * (1 + \zeta) + B_\phi \quad (25)$$

Para el caso de la solución de Y si la latitud de las coordenadas geodésicas pertenece al hemisferio sur se le debe sumar 10000000 al resultado obtenido.

4.2.2 Desarrollado o Adaptado y Embebido en el Arduino

Se utilizó un Arduino Uno R3, con el IDE 1.0.1 en AVR-GCC, en esta aplicación se usaron 2 clases adicionales para el desarrollo de la aplicación embebida (ver Tabla 3).

Clase	Descripción
SoftwareSerial	Esta clase permite la emulación de puertos seriales por software usando pines digitales del Arduino, en esta aplicación se requirió un total de 2 puertos seriales (Xbee y GPS).
TinyGPS.h	Esta clase permite la comunicación del GPS con el Arduino, para este caso la comunicación serial se realizó emulada (clase SoftwareSerial).

Tabla 3. Clases desarrolladas usadas en la aplicación embebida.
Fuente: El autor.

4.3 DATOS

Los datos que se usaron para esta investigación son:

- Fotomosaico del IGAC[18] con cubrimiento del área urbana de la ciudad de Villavicencio – Meta año 2008 (ver imagen 16).
- Imágenes de Google Earth con cubrimiento del área urbana de la ciudad de Villavicencio – Meta año 2011 (ver imagen 17).
- Fotografías aéreas capturadas con UAV modificado con cubrimiento del área urbana de la ciudad de Villavicencio – Meta año 2012 (ver imagen 18).

[18] Instituto Geografico Agustin Codazzi: es la entidad encargada de producir el mapa oficial y la cartografía básica de Colombia; elaborar el catastro nacional de la propiedad inmueble; realizar el inventario de las características de los suelos; adelantar investigaciones geográficas como apoyo al desarrollo territorial; capacitar y formar profesionales en tecnologías de información geográfica y coordinar la Infraestructura Colombiana de Datos Espaciales (ICDE). [En línea]. Disponible en: http://www.igac.gov.co/igac

Imagen 16. Imagen Fotomosaico IGAC, Villavicencio – Meta, 2008.
Fuente: IGAC.

Imagen 17. Imagen Google Earth, Villavicencio – Meta, 2011.
Fuente: Google Earth.

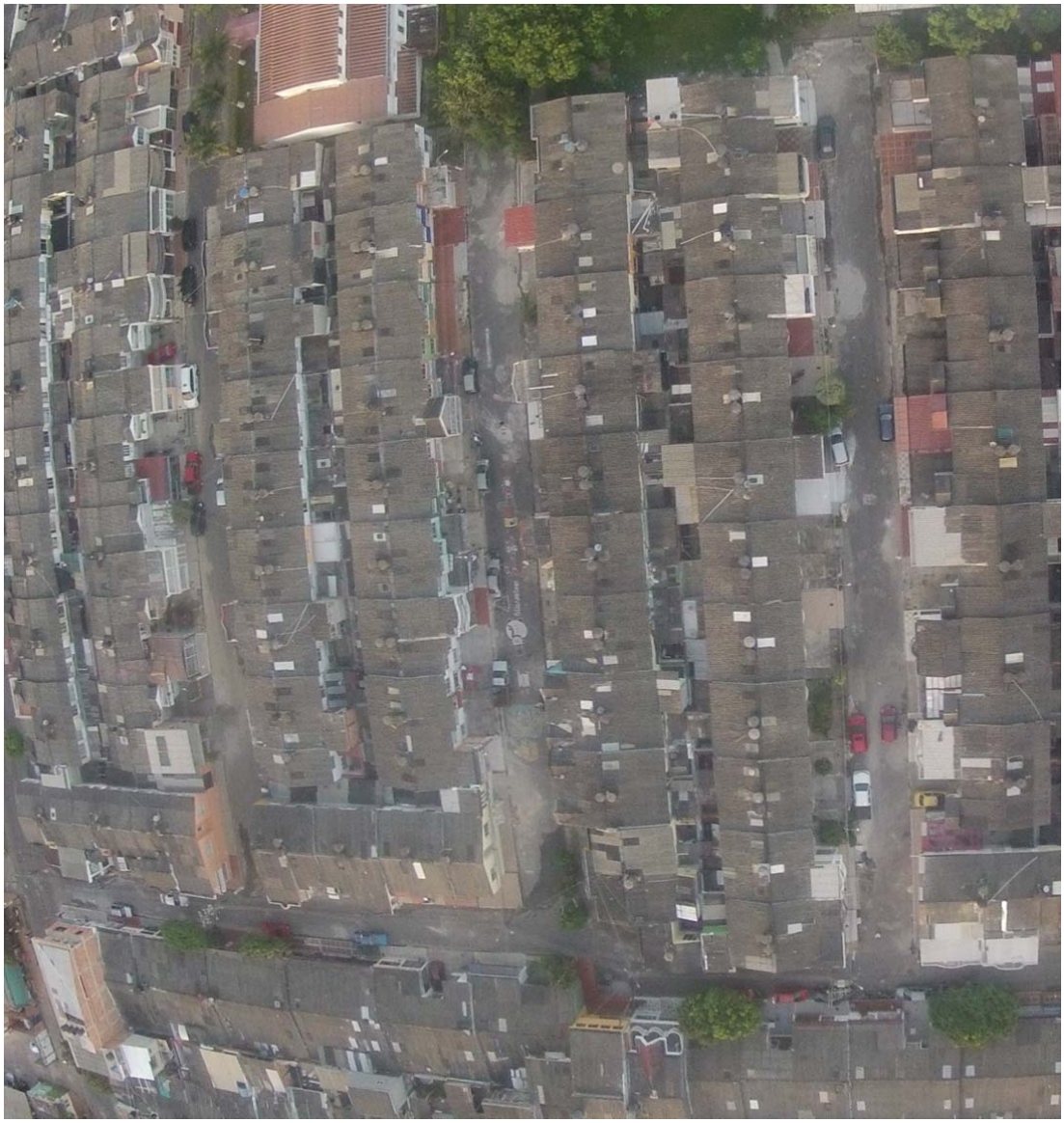

Imagen 18. Imagen tomada desde el ArDrone, con Cámara GoPro Hero 2, Villavicencio – Meta, 2012
Fuente: El autor.

4.3.1 Tratamiento de imágenes obtenidas

Las imágenes capturadas presentan distorsión radial o efecto barril, esto se debe a que la cámara GoPro Hero 2 cuenta con un lente gran angular para poder captar una mayor imagen, la distorsión hace que los objetos se vean deformados a los extremos, especialmente las construcciones, ventanas, todos aquellos que posean líneas rectas, esta distorsión es común en distancias focales cortas, para

corregir la distorsión se utilizó la herramienta Open Sourge Gimp, finalmente el área útil de la imagen capturada, donde las distorsiones geométricas son mínimas es la zona central de la imagen.

Luego de esto se procedió a georeferenciar la imagen en ArcMap utilizando la barra de Georeferencing, para ello se usó el punto capturado por el módulo de captura y transmisión GPS desarrollado y se tomaron otros puntos de control en campo para poder georeferenciar las imágenes (ver imagen 19).

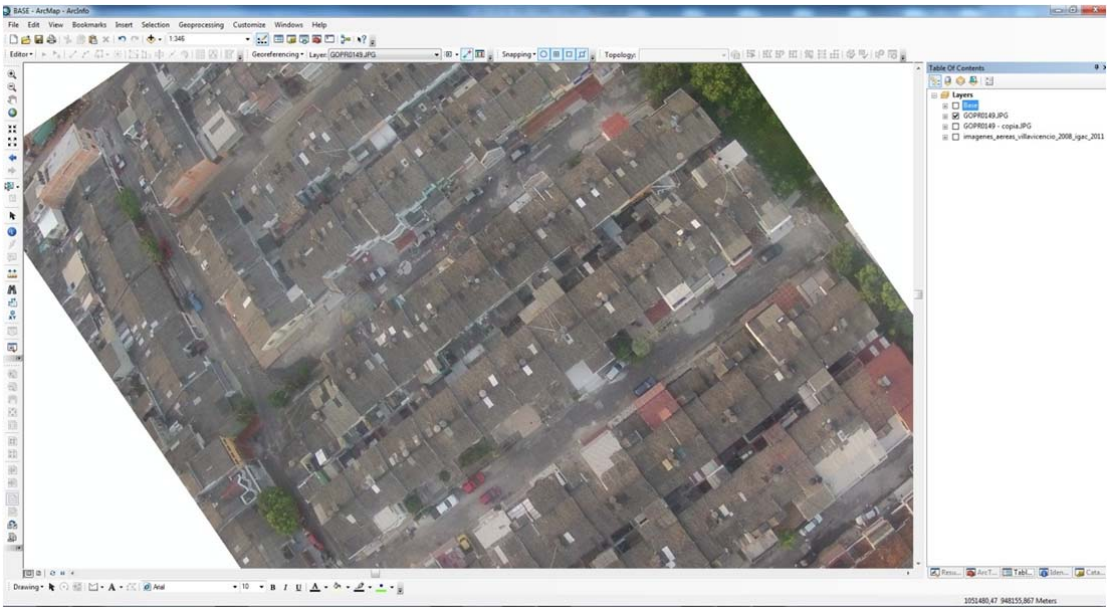

Imagen 19. Imagen georeferenciada en ArcMap usando Georeferencing
Fuente: El autor**.**

Antes de ser usadas las imágenes se requiere de la realización de ortorectificación, ya que estas presentan distorsiones causadas por el ángulo de la cámara, la distancia de la cámara a la superficie y los cambios de altura a la hora de la captura de las mismas, produciendo que la geometría difiera de la

geometría requerida, aunque en esta investigación no se realizó el proceso de ortorectificación.

5. RESULTADOS

5.1 HARDWARE Y SOFTWARE

Como resultado de la investigación se obtuvo la adaptación de un Cuadricoptero (ArDrone), el cual utiliza software y hardware diseñado para la captura de imágenes aéreas de bajo costo de buena calidad (ver imágenes 20 - 24).

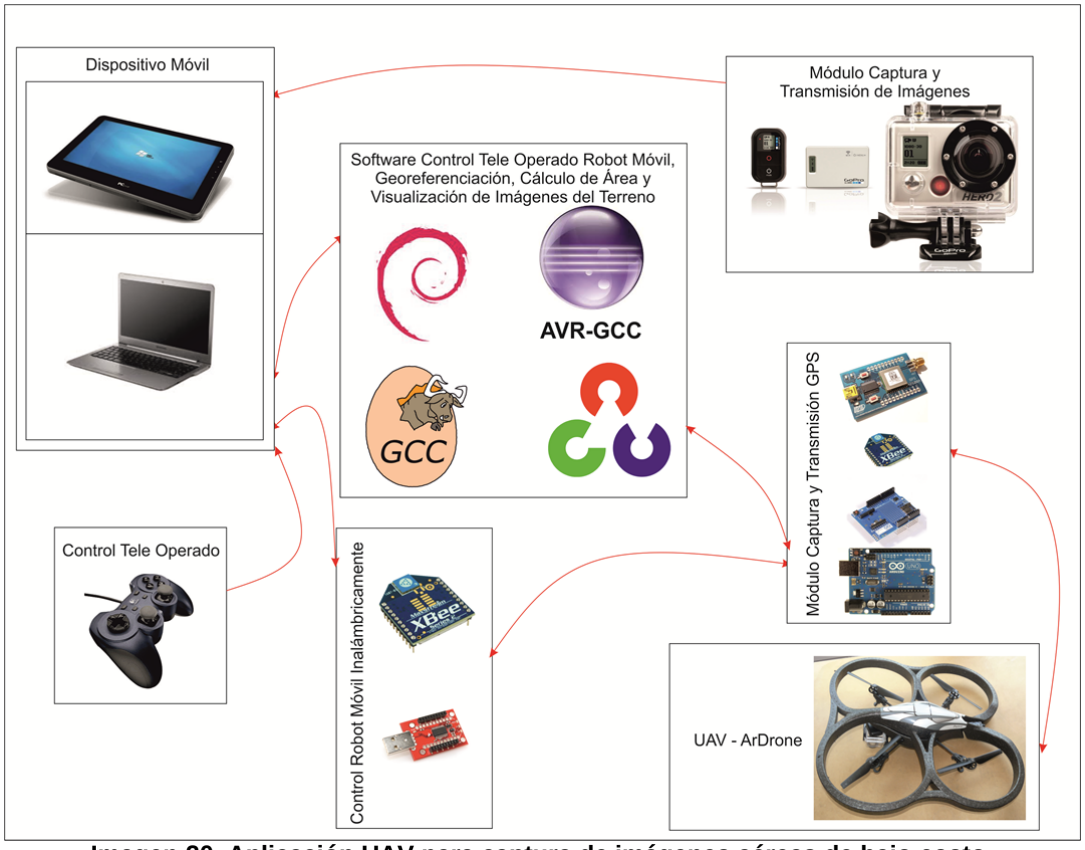

Imagen 20. **Aplicación UAV para captura de imágenes aéreas de bajo costo.**
Fuente: El autor.

Imagen 20. Montaje de la GoPro en el ArDrone
Fuente: El autor.

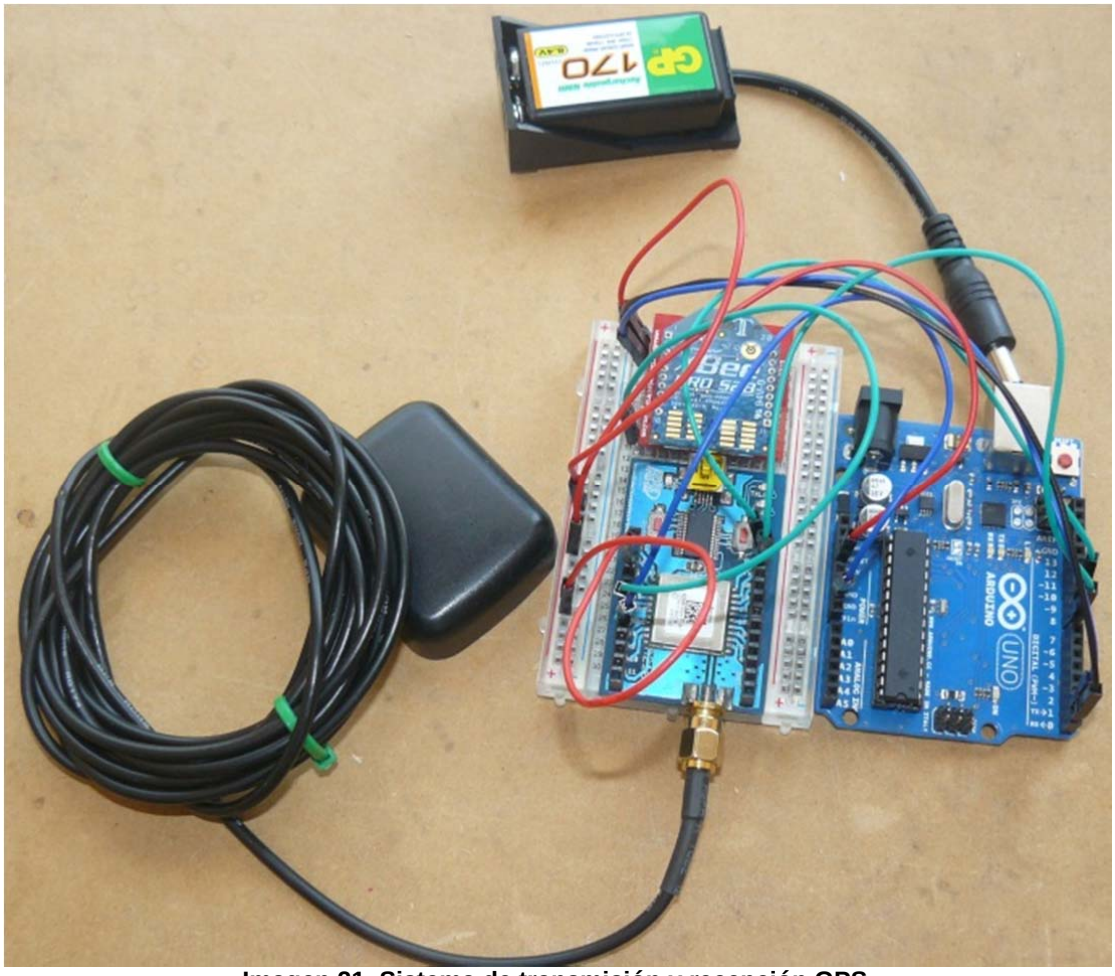

Imagen 21. Sistema de transmisión y recepción GPS
Fuente: El autor.

Imagen 22. Adaptación ArDrone, vista superior
Fuente: El autor.

Imagen 23. Adaptación ArDrone, vista inferior trasera
Fuente: El autor.

El diagrama de bloques resultante en la investigación (ver imagen 25) muestra al lado izquierdo de la imagen lo que hace referencia a la estación base y el software de la misma y en la parte derecha de la imagen lo referente al software y hardware diseñado y utilizado para el UAV.

La forma en que un usuario final interactúa con el prototipo es muy simple ya que este utiliza un GamePad y por medio de un puerto USB se comunica con el dispositivo móvil y por otro puerto USB con un módulo Xbee realizando una comunicación inalámbrica la cual utiliza el protocolo Zigbee al puerto serial del Atmega328 (Arduino Uno R3) el cual ejecuta el código embebido para la adquisición de datos GPS y por WIFI el control del UAV al igual que el control del sensor óptico.

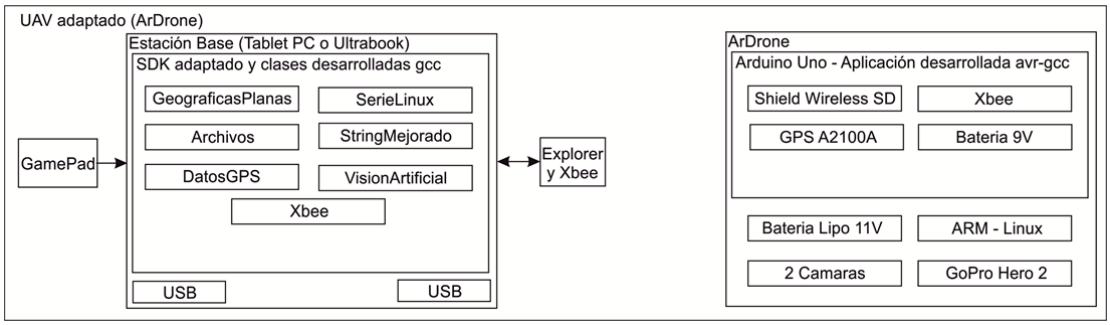

Imagen 24. **Diagrama de bloques.**
Fuente: El autor

5.2 IMÁGENES CAPTURADAS

5.2.1 comparación de imágenes

A continuación se mostraran algunas imágenes capturadas con el UAV y la cámara GoPro Hero 2 (ver imágenes 26, 28, 30, 32, 34, 36), estas imágenes se tomaron entre 80 y 110 metros de altura y una velocidad entre 30 y 40 Km/h.

Las imágenes capturadas presentan distorsión radial ya que la cámara GoPro Hero 2 cuenta con un lente gran angular, antes de ser usadas las imágenes se requiere la realización de una corrección de esta distorsión.

Las imágenes tomadas son contrastadas con las imágenes del IGAC (ver imágenes 27, 29, 35) o del Google Earth (ver imágenes 31, 33, 37) según la disponibilidad y cobertura de las anteriores. El punto central de las imágenes es el que se encuentra georeferenciado con un error aproximado entre 3 y 5 metros comparando con las imágenes del IGAC que se encuentran georeferenciadas.

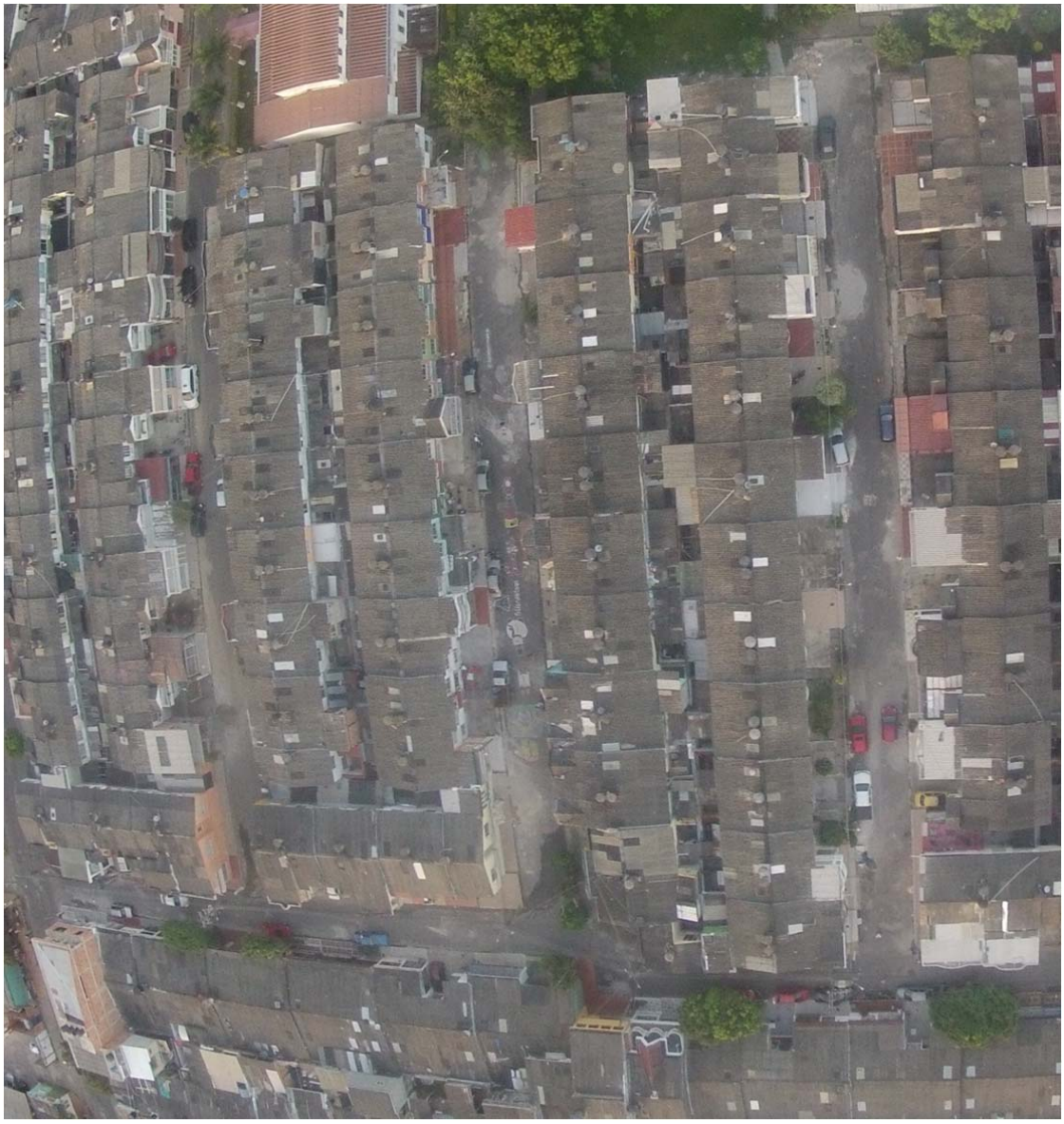

Imagen 25. Barrió Hacaritama tomada con GoPro Hero 2.
Fuente: El autor.

Imagen 26. Barrió Hacaritama.
Fuente: IGAC.

Imagen 27. Barrió Multifamiliar los Centauros tomada con GoPro Hero 2.
Fuente: El autor.

Imagen 28. Barrió Multifamiliar los Centauros.
Fuente: IGAC.

Imagen 29. Conjunto Cerrado cerca al Terminal de Transporte tomada con GoPro Hero 2.
Fuente: El autor.

Imagen 30. Conjunto Cerrado cerca al Terminal de Transporte.
Fuente: Google Earth.

Imagen 31. Conjunto Cerrado cerca al Terminal de Transporte tomada con GoPro Hero 2.
Fuente: El autor.

Imagen 32. Conjunto Cerrado cerca al Terminal de Transporte.
Fuente: Google Earth.

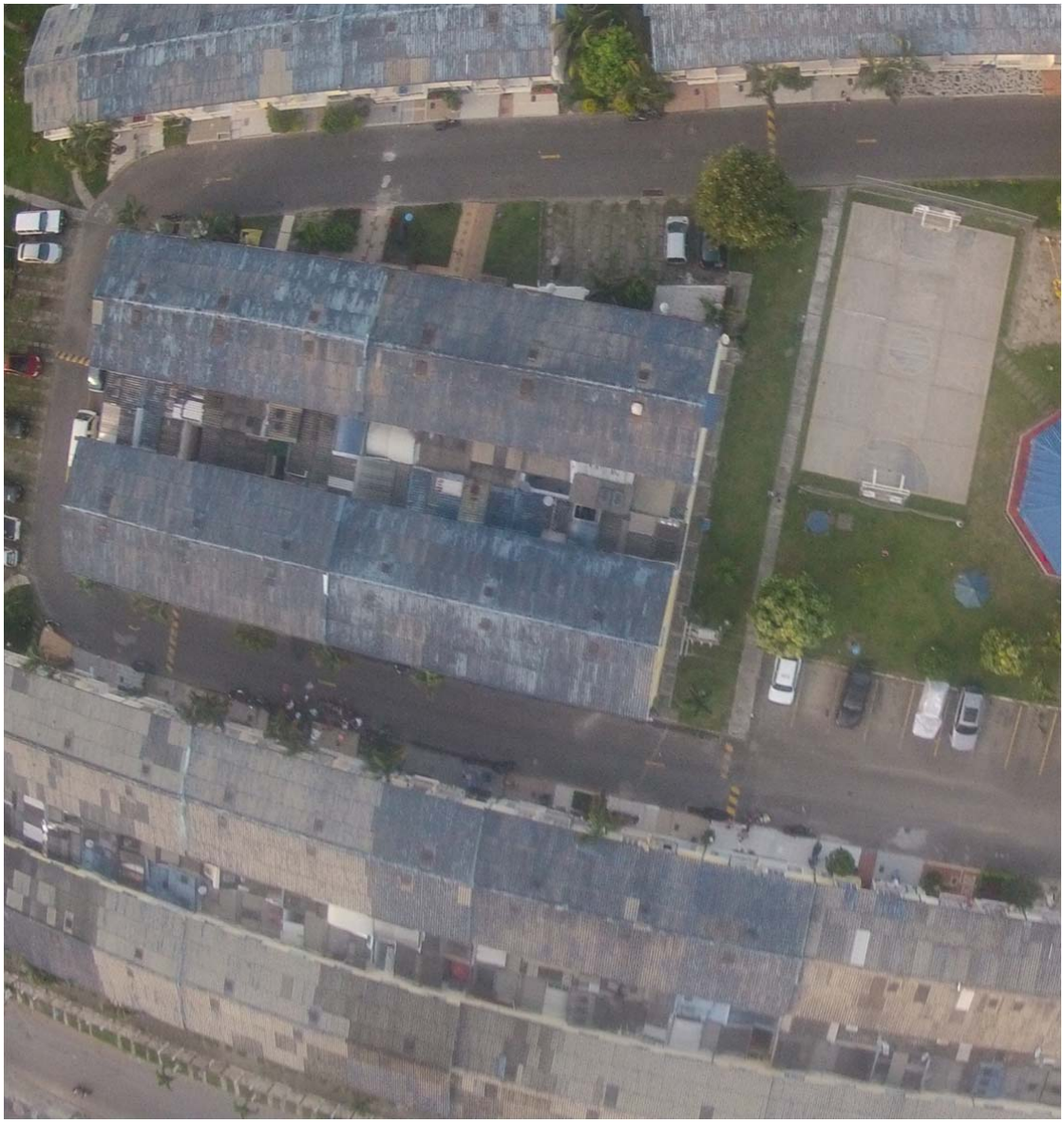

Imagen 33. Conjunto Cerrado cerca al Terminal de Transporte tomada con GoPro Hero 2.
Fuente: El autor.

Imagen 34. Conjunto Cerrado cerca al Terminal de Transporte.
Fuente: IGAC.

Imagen 35. Barrió la Alborada tomada con GoPro Hero 2.
Fuente: El autor.

Imagen 36. Barrió la Alborada.
Fuente: Google Earth.

5.2.2 Uso de las imágenes

Estas imágenes se pueden aplicar en muchos campos, ejemplo de ello es la agricultura, el catastro, la ingeniería y todo donde las imágenes puedan aportar información de utilidad. Como ejemplo se tomó una imagen de las capturadas, luego se georeferenció con cuatro puntos de control en ArcMap usando Georeferencing y luego se procedió a generar las planchas catastrales de dos manzanas de la ciudad de Villavicencio en el barrio Hacaritama (ver imágenes 38 y 39).

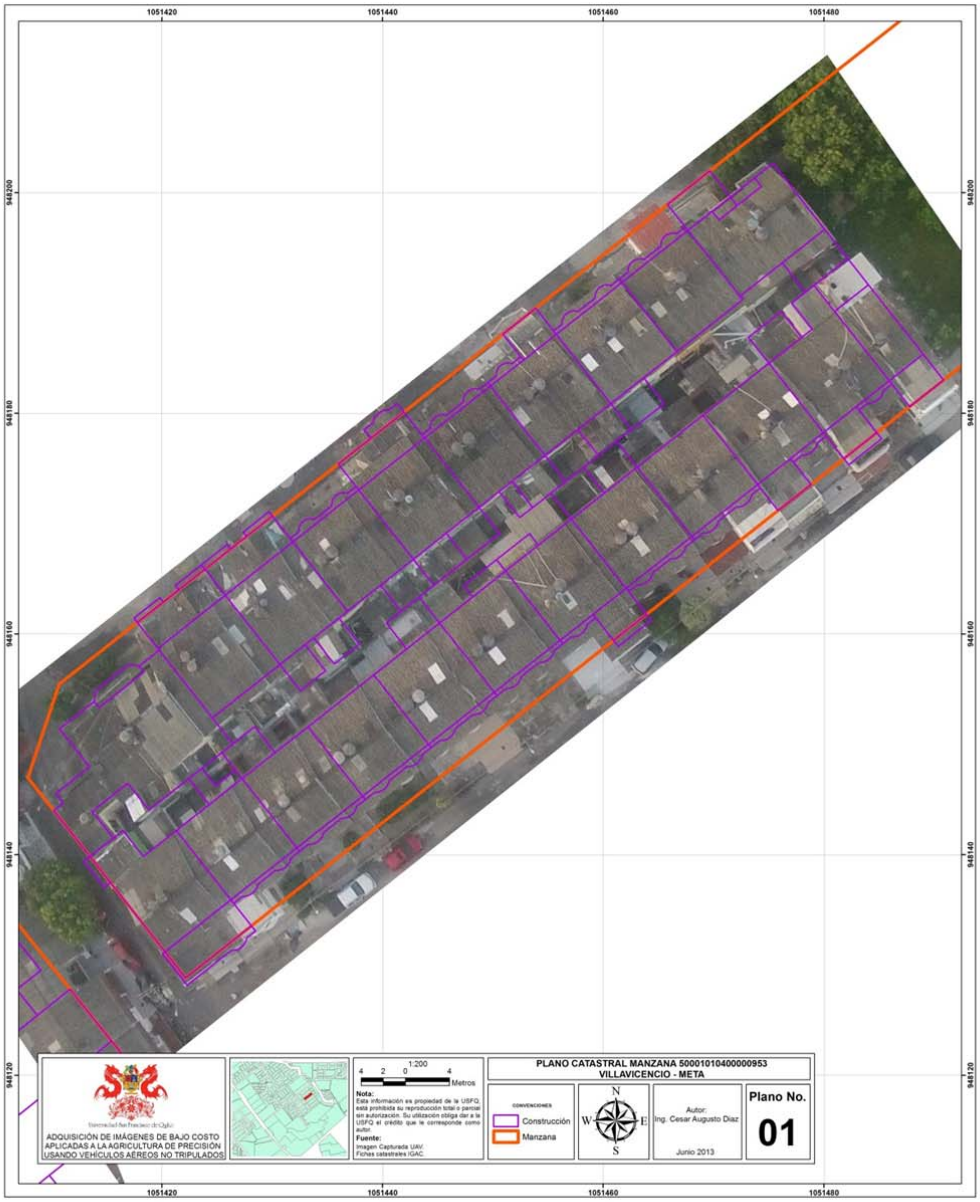

Imagen 37. Plano catastral manzana 50001010400000953
Fuente: El autor**.**

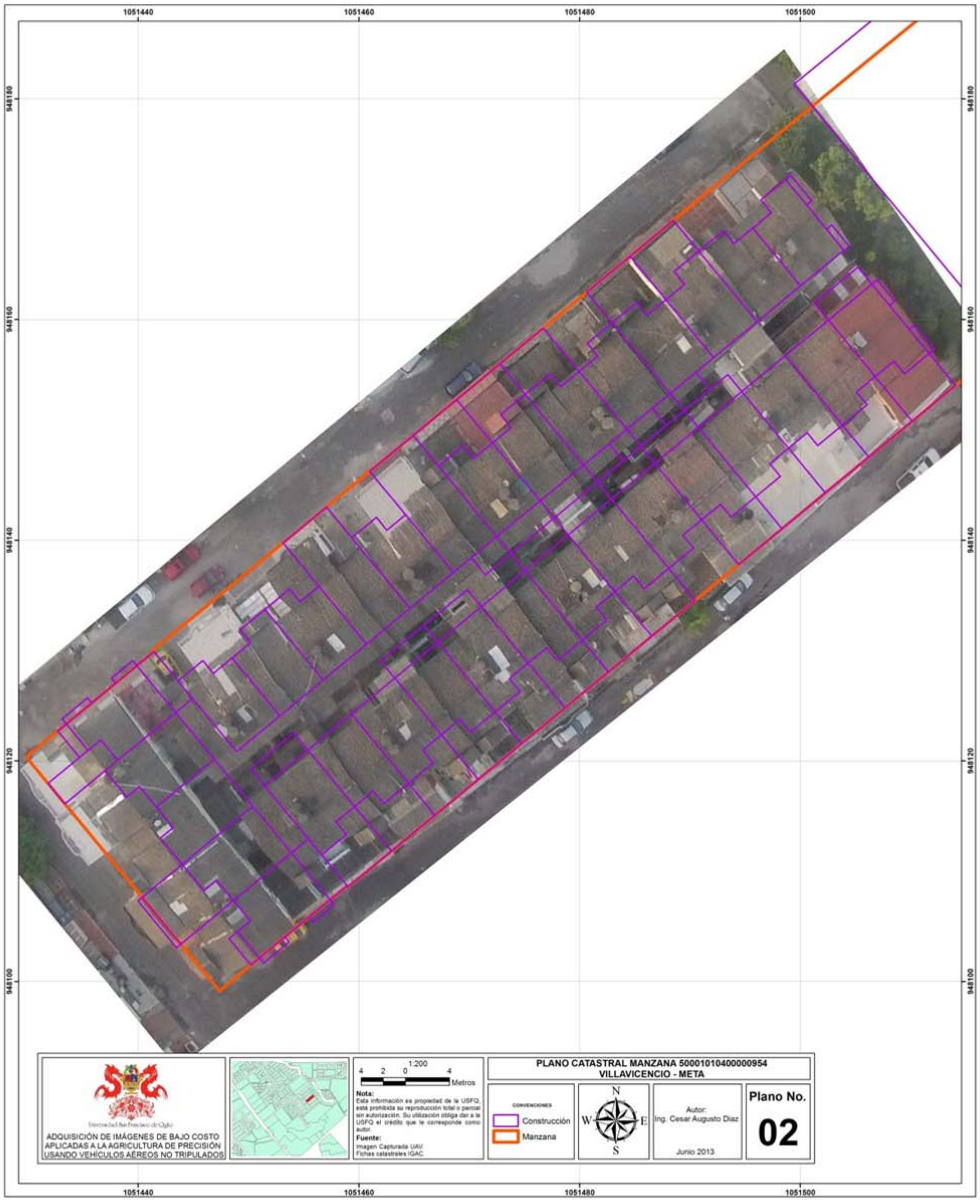

Imagen 38. Plano catastral manzana 50001010400000953
Fuente: El autor**.**

6. CONCLUSIONES

- En esta investigación se obtuvieron imágenes aéreas de bajo costo, lo cual es de gran utilidad no solo para las grandes empresas sino también para las pequeñas empresas ya que los costos de captura son reducidos a comparación de las imágenes convencionales aéreas y las imágenes satelitales.

- Las imágenes que se obtuvieron en esta investigación son de gran utilidad en los levantamientos catastrales ya que estas sirven de guía para lo construcción de este tipo de modelos.

- Los UAVs son un complemento a los sistemas tradicionales para la captura de imágenes (fotografías aéreas avión convencional e imágenes satelitales) con una alta resolución y de bajo costo.

- Los UAVs Permiten la construcción de estudios multitemporales ya que se pueden captar las imágenes en cualquier momento, en tiempo real y por debajo de las nubes, esto es de gran ayuda para la agricultura, dando un manejo apropiado a los recursos para poder satisfacer la demanda de una población que día a día crece, que mejor para todo esto que el uso de tecnologías de punta las cuales mejorarían la producción.

- Desde el punto de vista de la electrónica utilizada en el desarrollo del prototipo es importante resaltar el uso tanto de hardware libre como de

hardware privativo y en especial la gran capacidad que tiene el microprocesador Atmega328 sobre la plataforma Arduino Uno R3 y el microprocesador ARM que se encuentra en el hardware del UAV.

- Desde el punto de vista del software se destaca el uso de software libre para el diseño de la aplicación final del dispositivo móvil utilizando como sistema Operativo Debian 6 Testing y para la parte de programación la librería de visión artificial OpenCV 2.4, el Compilador GCC 4.7 y el IDE Eclipse Juno. Para la aplicación embebida en el Arduino el uso del IDE 1.0.1 del mismo y AVR-GCC.

- En nuestro país se requiere el desarrollo de algunas áreas para que la agricultura de precisión sea una solución de gran acogida y viable para el sector que lo demanda.

7. TRABAJOS FUTUROS

- Se plantea el diseño del UAV como un dispositivo que pueda navegar de forma autónoma por un área de terreno pre establecida bajo coordenadas GPS.

- Diseñar la captura de variables del ambiente como temperatura y humedad.

- Una de las limitantes del UAV seleccionado es la carga útil la cual es muy baja y no permite el agregar componentes de mayor peso, sería bueno investigar otras plataformas UAV que le den mayor autonomía para el vuelo.

- El video que transmite el Wifi Bac Pac es de muy baja calidad, sería bueno probar con otro tipo de transmisores y receptores de video.

- La cámara GoPro Hero 2 está diseñada para grabar flujo de video de muy buena calidad más no para la toma de imágenes, es bueno el estudiar otro tipo de sensores para mejorar la calidad de las imágenes capturadas.

- La cámara GoPro Hero 2 posee un lente gran angular, lo cual genera distorsión radial o efecto barril, para ello se requiere el estudio de esto y aplicar la respectiva corrección de las imágenes antes de usarse.

- Conocer más acerca del proceso de ortorectificación de imágenes con el fin de disminuir las distorsiones geométricas y obtener imágenes de mejor calidad.

- Buscar la financiación para llevar la aplicación a estándares militares, con el fin de tener una plataforma mucho más robusta.

8. BIBLIOGRAFIA

[1] García, V. J. A., Vásquez, A. L. A. 2011. Los robots en el sector agrícola. Universidad Politécnica de Madrid. Departamento de automática. Ingeniería electrónica e informática industrial. Madrid. España.

[2] Acevo, H. R. 2011. Sistemas de teledetección activos y pasivos embarcados en sistemas aéreos no tripulados para la monitorización de tierras. Tesis doctoral. Departamento de teoría de la señal y comunicaciones. Universidad Politécnica de Cataluña. España.

[3] Bendayan L. Sistema de adquisición remota de imágenes mediante vehículos aéreos no tripulados (UAV). Programa de investigación en información de la biodiversidad Amazónica – BIOINFO. Proyecto: Desarrollo de tecnologías en sistemas de información sobre biodiversidad, sociobiodiversidad y economía amazónica (SITEC). Instituto de investigación de la amazonia peruana. Perú.

[4] Redolfi, J., Henze A. 2011. Quadricóptero autónomo de arquitectura abierta, QA3. Tesis de Ingeniería. Ingeniería Electrónica. Universidad Tecnológica Nacional. Argentina.

[5] CIAT, 2007. Mejora de los sistemas de cartografía del territorio colombiano. Centro Internacional de Agricultura Tropical. [En línea].Disponible en: ftp://ftp.ciat.cgiar.org/planificacion/GEOMATICA/GPS/GPS_Modulo.pdf

[6] GPS.GOV. Información oficial del Gobierno de los Estados Unidos relativa al Sistema de Posicionamiento Global y temas afines. [En línea]. Disponible en: http://www.gps.gov/spanish.php

[7] Blackmore, B. S., Wheelwe, P. N., Morris, J., Morris, R. M., Jones, R. J. A. 1995. The role of precision farming in sustainable agriculture: A European perspective. In: Proceedinngs Precision Agriculture Conference, Madison, WI, USA. 777 – 793.

[8] Godwin, R. J., Earl, R., Taylor, J. C., Woosd, G. A., Bradley, R. I., Welsch, J. P., Richard, T., Blackmore, B. S. 2001. Precision farming of cereals: A five year experiment to develop management guidelines. Project Report No. 267. HGCA. Cranfield University.

[9] Segura, L. F., Torres, S., Vélez, J., Correa, S., Vargas, G., Pérez, P. 2010. Ejercicio EAD Agricultura de precisión. Servicio Nacional de Aprendizaje SENA. Santiago de Cali. Colombia.

[10]Lowenberg, D. J. 2001. Agricultura de precisión en EE.UU. y potencial de adopción en los paisas en desarrollo. Universidad de Purdue.

[11]Chartuni, E., de Assis, F., Marcal, D., Ruz, E. 2007. Nuevas herramientas para mejorar la gestión tecnológica en la empresa agropecuaria. Comunica. Edición No. 1, II Etapa.

[12]Leiva, F. R. 2003. La agricultura de precisión: una producción más sostenible y competitiva con visión futura. Conferencia presentada en el VIII congreso de la sociedad Colombiana de Fitomejoramiento y producción de cultivos, Bogotá, D.C.

[13]Ortega, R., Flores, L. 1999. Agricultura de Precisión: Introducción al manejo sitio - específico. Ministerio de Agricultura, Instituto de Investigaciones Agropecuarias, CRI Quilamapu-Chile.

[14]Segura, L. F., Torres, S., Vélez, J., Correa, S., Vargas, G., Pérez, P. 2010. Ejercicio EAD Agricultura de precisión. Servicio Nacional de Aprendizaje SENA. Santiago de Cali. Colombia.

[15]Bernal O. A., Orrego B. S. 2007. Diseño del ala para un vehículo aéreo no tripulado. Investigación de Pregrado. Escuela de ingenierías, Universidad EAFIT. Medellín, Colombia.

[16]Zarco Tejada P., Berni J. A. J., Suarez L., Federes E. 2008. A new era in remote sensing with unmanned robots. Laboratory for research methods in quantitative remote sensing. Instituto de agricultura sostenible (IAS). Consejo superior de investigaciones científicas (CSIC). Cordoba. España.

[17]Pillai S. G., Tian L., Beal J.: 1998. Detection of nitrogen stress in corn using digital aerial imaging. ASAE Paper No. 983030 in 1998 ASAE annual international meeting. Orlando. Florida. USA.

[18] Suarez, L. 2009. Teledetección de estrés hídrico en cultivos mediante el índice de reflectancia fotoquímica (PRI). Tesis doctoral. Departamento de agronomía. Universidad de Córdoba. España.

[19] Muguerza J. 2007. Uso de herramientas de agricultura de precisión en aceitera general deheza. Uso propio y visión de servicio. En séptimo curso internacional de agricultura de precisión y segunda expo de máquinas precisas. INTA EEA Manfredi. 17 al 19 de julio 2007. Cordoba. Argentina.

[20] Herwitz, S. R., Johnson, L. F., Dunagan, S. E., Higgins, R. G., Sullivan, D. V., Zheng, J., Lobitz, B. M., Leung, J. G., Gallmeyer, B. A., Aoyagi, M., Slye, R. E., Brass, J. A. 2004. Imaging from an unmanned aerial vehicle: agricultural surveillance and decision support. Computers and Electronics in Agriculture. Volumen 44 (1): 49-61.

[21] Xiang, H., Tian, L. 2006. Development of autonomous unmanned helicopter based agricultural remote sensing system. ASABE paper no. 063097.

[22] Niethammer U., James M., Rothmund S., Travelletti J., Joswig M. 2011. UAV – based remote sensing of the Super-Sauze landslide: evaluation and results. Engineering Geology. Vol. 128 pp 2-11. [En línea]. Disponible en: http://www.sciencedirect.com/science/article/pii/S0013795211000755

[23]Kudo H., Koshino Y., Eto A., Ichimura M., Kaeriyama M. 2012. Cost-effective accurate estimates of adult chum salmon, oncorhynchus keta, abundance in a japanese river using a radio-controlled helicopter. Fisheries Research. Vol 119-120 pp 94-98. [En línea]. Disponible en: http://www.sciencedirect.com/science/article/pii/S0165783611003882

[24]Chiabrando F., Nex F., Piatti D., Rinaudo F. 2011. UAV and RPV systems for photogrammetric surveys in archaeological areas: two tests in the piedmont region (Italy). Journal of Archaeological Science. Vol. 38 pp 697-710. [En línea]. Disponible en: http://www.sciencedirect.com/science/article/pii/S0305440310003821

[25]Samsung. Series 5 Ultra Alto Rendimiento NP530U4C. Consultado el 20 de octubre de 2012. [En línea]. Disponible en: http://www.samsung.com/co/consumer/monitor-peripherals-printer/notebooks/ultrabook/NP530U4C-A03CO

[26]Maestro GPS reveiver. GPS Receivers A2100-A. Consultado el 25 de octubre de 2012. [En línea]. Disponible en: http://www.mt-system.ru/sites/default/files/docs/documents/gps_receiver_a2100_v12.pdf

[27]Didáctica electrónica. Tarjeta gps A2100A USB. Consultado el 25 de octubre de 2012. [En línea]. Disponible en: http://www.didacticaselectronicas.com/index.php?page=shop.product_details&flypage=flypage.tpl&product_id=1002&category_id=154&option=com_virtuemart&Itemid=76&vmcchk=1&Itemid=76

[28]GoPro. GoPro Hero 2. Consultado el 30 de octubre de 2012. [En línea]. Disponible en: http://es.gopro.com/cameras/hd-hero2-outdoor-edition/#specs

[29]DIGI. Xbee Pro serie 2B. Consultado el 30 de octubre de 2012. [En línea]. Disponible en: http://www.digi.com/pdf/ds_xbeezbmodules.pdf

[30]GoPro. Wi-Fi BacPac + Wi-Fi Kit Combinado Remoto. Consultado el 30 de octubre de 2012. [En línea]. Disponible en: http://es.gopro.com/hd-hero-accessories/wi-fi-bacpac-remote-combo#description

[31]Parrot. Ar Drone 1. Consultado el 30 de octubre de 2012. [En línea]. Disponible en: http://es.wikipedia.org/wiki/Parrot_AR.Drone

[32]Arduino. Arduino Uno R3. Consultado el 30 de octubre de 2012. [En línea]. Disponible en: http://arduino.cc/en/Main/ArduinoBoardUno

[33]G. Bradski, A. Kaehler, “Learning OpenCV: Computer Vision with the OpenCV Library”, O’Reilly Media, 2008.

[34]R. Laganiere, “OpenCV 2 Computer Vision Application Programming Cookbook”, Packt Publishing, 2011.

[35]AR.Drone open API platform. SDK 2.0. Consultado el 30 de octubre de 2012. [En línea]. Disponible en: https://projects.ardrone.org/

[36]National Marine Electronics Association. [En línea]. Disponible en: http://www.nmea.org/

[37]Manual de referencia NMEA. [En línea]. Disponible en: http://www.sparkfun.com/datasheets/GPS/NMEA%20Reference%20Manual-Rev2.1-Dec07.pdf

[38] G. Ortiz, “Aprende a convertir coordenadas geograficas en UTM y UTM en geograficas”. [En línea]. Disponible en: http://www.gabrielortiz.com/index.asp?Info=058a

9. ANEXOS

A.1. INSTALACIÓN DEL SDK 2.0

A.2. CLASES ADICIONALES DISPOSITIVO MOVIL

A.3. APLICACIÓN EMBEBIDA EN ARDUINO UNO R3

A.9. CONTENIDO DEL CD

A.1. INSTALACIÓN DEL SDK 2.0

1. Descargar el SDK 2.0, para ello debe de estar registrado:
   https://projects.ardrone.org/login?back_url=http%253A%252F%252Fprojects.ardrone.org%252Fattachments%252Fdownload%252F434%252FARDrone_SDK_2_0.tar.gz

2. Abrir una terminal en Linux como usuario root y ubicarse en la ruta donde quiere descomprimir el SDK.

3. Ahora se ingresa a la siguiente carpeta con la siguiente instrucción:
   cd ARDrone_SDK_2_0/Examples/Linux

4. Instalamos los siguientes paquetes con la instrucción:
   apt-get install libsdl1.2-dev libudev-dev libiw-dev libgtk2.0-dev libxml2-dev libncurses5-dev

5. Compilamos con la instrucción:
   make

6. Ingresamos a:
   cd Build/Release

7. Ahora conectamos la batería al ArDrone y lo colocamos en una superficie plana.

8. Buscamos en las conexiones inalámbricas del dispositivo móvil y lo vinculamos a la red del ArDrone.

9. En la consola ejecutamos:

   ./ardrone_navigation

10. Ya dentro de la aplicación configuramos el GamePad con los siguientes valores:

    take of/land : return : boton 10 : valor 9

    emergency/reset : space : boton 9 : valor 8

    pitch front : up : boton flecha arriba : valor -2

    pitch back : down : boton flecha abajo : valor 2

    roll left : q : boton flecha izquierda : valor -1

    roll right : d : boton flecha derecha : valor 1

    yaw left : left : boton 5 : valor 4

    yaw right : right : boton 6 : valor 5

    vertical speed up : z : boton j2 arriba : valor -4

    vertical speed down : x : boton j2 abajo : valor 4

11. Con esto ya podemos controlar el ArDrone.

A.2. CLASES ADICIONALES DISPOSITIVO MOVIL

Archivos.h Clase construida

```
/*
 * Archivos.h
 *
 *  Created on: 16/04/2012
 *      Author: diaz
 */
#ifndef __Archivos_H__
#define __Archivos_H__
#include "DatosGPS.h"
class Archivos
	{
	private:
		FILE *archivo;
	public:
		void abrirArchivo(char *nombreArchivo, char *modo);
		void cerrarArchivo(void);
		void escribirRegistroArchivo(DatosGPS datos);
		DatosGPS leerRegistroArchivo(int registro);
		long int sizeArchivo(void);
	};
#endif
```

Archivos.cpp

```
/*
 * Archivos.cpp
 *
 *  Created on: 16/04/2012
 *      Author: diaz
 */

#include "Archivos.h"

void Archivos::abrirArchivo(char *nombreArchivo, char *modo)
	{
	//compara si el modo es escritura
	if (strcmp(modo, "escritura") == 0)
		{
		//abre el archivo para escritura al final, si no existe lo crea
		this->archivo = fopen(nombreArchivo, "a+");
		}

	//compara si el mode es lectura
	else if (strcmp(modo, "lectura") == 0)
		{
		//abre el archivo para lectura, el archivo debe de existir
		this->archivo = fopen(nombreArchivo, "r");
		}

	}

void Archivos::cerrarArchivo(void)
	{
	fclose(this->archivo);
	}
```

```
void Archivos::escribirRegistroArchivo(DatosGPS datos)
	{
	fwrite(&datos, datos.sizeDatos(), 1, this->archivo);
	}

DatosGPS Archivos::leerRegistroArchivo(int registro)
	{
	DatosGPS datos;

	//si el registro solicitado existe en el archivo
	if (((registro * datos.sizeDatos()) <= sizeArchivo()) && (registro > 0))
		{

		//se ubica en el inicio del registro solicitado
		fseek(this->archivo, (registro - 1) * datos.sizeDatos(), SEEK_SET);

		//lee un registro de tipo datos del archivo
		fread(&datos, datos.sizeDatos(), 1, this->archivo);
		}
	else
		{
		printf("El registro no existe\n");
		}

	return datos;
	}

long int Archivos::sizeArchivo(void)
	{
	long int posActual = 0, size = 0;

	//almacena la posicion actual en el archivo
	posActual = ftell(this->archivo);

	//se ubica al final del archivo
	fseek(this->archivo, 0, SEEK_END);

	//almacena el tamano del archivo
	size = ftell(this->archivo);

	//se ubica en el lugar que estaba del archivo
	fseek(this->archivo, posActual, SEEK_SET);

	//retorna el tamano
	return size;
	}
```

DatosGPS.h Clase construida

```
/*
 * DatosGPS.h
 *
 *  Created on: 16/04/2012
 *      Author: diaz
 */
#ifndef __DatosGPS_H__
#define __DatosGPS_H__

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

#include "StringMejorado.h"

class DatosGPS
	{
//--------------elementos privados-----------------//
	private:
		char hora[20];
		char latitudGradosMinutos[10];
		char indicadorNS[2];
		char longitudGradosMinutos[10];
		char indicadorEW[2];
		char indicadorPosicionFija[40];
		char satelitesUsados[3];
		double precisionDilucionHorizontal;
		double altitud;
		char altitudUnidad[2];

		char latitudGradosMinutosSegundos[20];
		char longitudGradosMinutosSegundos[20];
		double latitudDecimal;
		double longitudDecimal;
		double latitudRadianes;
		double longitudRadianes;
		int huso;
		double X;
		double Y;

//--------------elementos publicos-----------------//
	public:
//--------------set-----------------//
		void setHora(char *hora);
		void setLatitudGradosMinutos(char *latitudGradosMinutos);
		void setIndicadorNS(char *indicadorNS);
		void setLongitudGradosMinutos(char *longitudGradosMinutos);
		void setIndicadorEW(char *indicadorEW);
		void setIndicadorPosicionFija(char *indicadorPosicionFija);
		void setSatelitesUsados(char *satelitesUsados);
		void setPrecisionDilucionHorizontal(double precisionDilucionHorizontal);
		void setAltitud(double altitud);
		void setAltitudUnidad(char *altitudUnidad);

		void setLatitudGradosMinutosSegundos(char *latitudGradosMinutosSegundos);
		void setLongitudGradosMinutosSegundos(char
*longitudGradosMinutosSegundos);
		void setLatitudDecimal(double latitudDecimal);
```

```
                    void setLongitudDecimal(double longitudDecimal);
                    void setLatitudRadianes(double latitudRadianes);
                    void setLongitudRadianes(double longitudRadianes);
                    void setHuso(int huso);
                    void setX(double x);
                    void setY(double y);

//--------------get-----------------//
                    char *getHora(void);
                    char *getLatitudGradosMinutos(void);
                    char *getIndicadorNS(void);
                    char *getLongitudGradosMinutos(void);
                    char *getIndicadorEW(void);
                    char *getIndicadorPosicionFija(void);
                    char *getSatelitesUsados(void);
                    double getPrecisionDilucionHorizontal(void);
                    double getAltitud(void);
                    char *getAltitudUnidad(void);

                    char *getLatitudGradosMinutosSegundos(void);
                    char *getLongitudGradosMinutosSegundos(void);
                    double getLatitudDecimal(void);
                    double getLongitudDecimal(void);
                    double getLatitudRadianes(void);
                    double getLongitudRadianes(void);
                    int getHuso(void);
                    double getX(void);
                    double getY(void);

//--------------otros-----------------//
                    long int sizeDatos(void);
                    void datosGPGGA(char *datosRecibidos, int size);
                    bool tipoDatos(char *datosRecibidos, char *tipo);
                    void datosArduinoGPS(char *datosRecibidos, int size);
          };
#endif
```

DatosGPS.cpp

```
/*
 * DatosGPSGPS.cpp
 *
 *  Created on: 16/04/2012
 *      Author: diaz
 */

#include "DatosGPS.h"

//--------------set-----------------//
void DatosGPS::setHora(char *hora)
          {
          strcpy(this->hora, hora);
          }

void DatosGPS::setLatitudGradosMinutos(char *latitudGradosMinutos)
          {
          strcpy(this->latitudGradosMinutos, latitudGradosMinutos);
          }

void DatosGPS::setIndicadorNS(char *indicadorNS)
```

```
        {
        strcpy(this->indicadorNS, indicadorNS);
        }

void DatosGPS::setLongitudGradosMinutos(char *longitudGradosMinutos)
        {
        strcpy(this->longitudGradosMinutos, longitudGradosMinutos);
        }

void DatosGPS::setIndicadorEW(char *indicadorEW)
        {
        strcpy(this->indicadorEW, indicadorEW);
        }

void DatosGPS::setIndicadorPosicionFija(char *indicadorPosicionFija)
        {
        strcpy(this->indicadorPosicionFija, indicadorPosicionFija);
        }

void DatosGPS::setSatelitesUsados(char *satelitesUsados)
        {
        strcpy(this->satelitesUsados, satelitesUsados);
        }

void DatosGPS::setPrecisionDilucionHorizontal(double precisionDilucionHorizontal)
        {
        this->precisionDilucionHorizontal = precisionDilucionHorizontal;
        }

void DatosGPS::setAltitud(double altitud)
        {
        this->altitud = altitud;
        }

void DatosGPS::setAltitudUnidad(char *altitudUnidad)
        {
        strcpy(this->altitudUnidad, altitudUnidad);
        }

void DatosGPS::setLatitudGradosMinutosSegundos(char *latitudGradosMinutosSegundos)
        {
        strcpy(this->latitudGradosMinutosSegundos, latitudGradosMinutosSegundos);
        }

void DatosGPS::setLongitudGradosMinutosSegundos(char *longitudGradosMinutosSegundos)
        {
        strcpy(this->longitudGradosMinutosSegundos, longitudGradosMinutosSegundos);
        }

void DatosGPS::setLatitudDecimal(double latitudDecimal)
        {
        this->latitudDecimal = latitudDecimal;
        }

void DatosGPS::setLongitudDecimal(double longitudDecimal)
        {
        this->longitudDecimal = longitudDecimal;
        }
```

```
void DatosGPS::setLatitudRadianes(double latitudRadianes)
	{
	this->latitudRadianes = latitudRadianes;
	}

void DatosGPS::setLongitudRadianes(double longitudRadianes)
	{
	this->longitudRadianes = longitudRadianes;
	}

void DatosGPS::setHuso(int huso)
	{
	this->huso = huso;
	}

void DatosGPS::setX(double x)
	{
	this->X = x;
	}

void DatosGPS::setY(double y)
	{
	this->Y = y;
	}

//--------------get-----------------//
char *DatosGPS::getHora(void)
	{
	return this->hora;
	}

char *DatosGPS::getLatitudGradosMinutos(void)
	{
	return this->latitudGradosMinutos;
	}

char *DatosGPS::getIndicadorNS(void)
	{
	return this->indicadorNS;
	}

char *DatosGPS::getLongitudGradosMinutos(void)
	{
	return this->longitudGradosMinutos;
	}

char *DatosGPS::getIndicadorEW(void)
	{
	return this->indicadorEW;
	}

char *DatosGPS::getIndicadorPosicionFija(void)
	{
	return this->indicadorPosicionFija;
	}

char *DatosGPS::getSatelitesUsados(void)
	{
	return this->satelitesUsados;
```

```
        }

double DatosGPS::getPrecisionDilucionHorizontal(void)
        {
        return this->precisionDilucionHorizontal;
        }

double DatosGPS::getAltitud(void)
        {
        return this->altitud;
        }

char *DatosGPS::getAltitudUnidad(void)
        {
        return this->altitudUnidad;
        }

char *DatosGPS::getLatitudGradosMinutosSegundos(void)
        {
        return this->latitudGradosMinutosSegundos;
        }

char *DatosGPS::getLongitudGradosMinutosSegundos(void)
        {
        return this->longitudGradosMinutosSegundos;
        }

double DatosGPS::getLatitudDecimal(void)
        {
        return this->latitudDecimal;
        }

double DatosGPS::getLongitudDecimal(void)
        {
        return this->longitudDecimal;
        }

double DatosGPS::getLatitudRadianes(void)
        {
        return this->latitudRadianes;
        }

double DatosGPS::getLongitudRadianes(void)
        {
        return this->longitudRadianes;
        }

int DatosGPS::getHuso(void)
        {
        return this->huso;
        }

double DatosGPS::getX(void)
        {
        return this->X;
        }

double DatosGPS::getY(void)
        {
```

```
	return this->Y;
	}

//--------------otros-----------------//
long int DatosGPS::sizeDatos(void)
	{
	return sizeof(DatosGPS);
	}

void DatosGPS::datosGPGGA(char *DatosGPSRecibidos, int size)
	{

	StringMejorado sm;

	//usada para el token
	char *token;

	char datoTemporal[50];

	char *tempDatosGPSRecibidos;

	tempDatosGPSRecibidos = sm.normalizarDatos(DatosGPSRecibidos, (char*)",,", (char*)",",size);

	printf("%s\n", tempDatosGPSRecibidos);

	//instruccion para separar por tokens $GPGGA
	token = strtok(tempDatosGPSRecibidos, ",");

	//instruccion para separar por tokens hhmmss.sss
	token = strtok(NULL, ",");

	//construccion del string para la hora
	sprintf(datoTemporal, "%s:%s:%s", sm.substr(token, 0, 2), sm.substr(token, 2, 2), sm.substr(token, 4, 2));

	//se guarda la hora en el atributo de la clase DatosGPS
	this->setHora(datoTemporal);

	//instruccion para separar por tokens latitud en ddmm.mmmm
	token = strtok(NULL, ",");

	//se guarda la latitud en el atributo de la clase DatosGPS
	this->setLatitudGradosMinutos(token);

	//instruccion para separar por tokens N=norte o S=sur
	token = strtok(NULL, ",");

	//se guarda el indicador NS en el atributo de la clase DatosGPS
	this->setIndicadorNS(token);

	//instruccion para separar por tokens longitud en ddmm.mmmm
	token = strtok(NULL, ",");

	//se guarda la longitud en el atributo de la clase DatosGPS
	this->setLongitudGradosMinutos(token);

	//instruccion para separar por tokens E=este o W=oeste
	token = strtok(NULL, ",");
```

```
//se guarda el indicador EW en el atributo de la clase DatosGPS
this->setIndicadorEW(token);

//instruccion para separar por tokens el indicador de posicion fija
token = strtok(NULL, ",");

//se revisa el valor del token y se asigna el mensaje correspondiente
if (token[0] == '0')
        {
        //se guarda el indicador de posicion fija en el atributo de la clase DatosGPS
        this->setIndicadorPosicionFija((char*)"No esta disponible o no es valido");
        }
else if (token[0] == '1')
        {
        //se guarda el indicador de posicion fija en el atributo de la clase DatosGPS
        this->setIndicadorPosicionFija((char*)"Modo GPS SPS punto valido");
        }
else if (token[0] == '2')
        {
        //se guarda el indicador de posicion fija en el atributo de la clase DatosGPS
        this->setIndicadorPosicionFija((char*)"GPS diferencial modo SPS punto valido");
        }
else if (token[0] == '3')
        {
        //se guarda el indicador de posicion fija en el atributo de la clase DatosGPS
        this->setIndicadorPosicionFija((char*)"Modo GPS PPS punto valido");
        }
else
        {
        //se guarda el indicador de posicion fija en el atributo de la clase DatosGPS
        this->setIndicadorPosicionFija((char*)"Indicador indefindo");
        }

//instruccion para separar por tokens satelites usados
token = strtok(NULL, ",");

//se guarda satelites usados en el atributo de la clase DatosGPS
this->setSatelitesUsados(token);

//instruccion para separar por tokens precision dilucion horizontal
token = strtok(NULL, ",");

//se guarda la precision dilucion horizontal en el atributo de la clase DatosGPS
this->setPrecisionDilucionHorizontal(atof(token));

//instruccion para separar por tokens altitud
token = strtok(NULL, ",");

//se guarda la altitud en el atributo de la clase DatosGPS
this->setAltitud(atof(token));

//instruccion para separar por tokens altitud unidad
token = strtok(NULL, ",");

//se guarda la altitud unidad en el atributo de la clase DatosGPS
this->setAltitudUnidad(token);
}
```

```
bool DatosGPS::tipoDatos(char *DatosGPSRecibidos, char *tipo)
	{
	int pos = 0, posInicial = 0;

	//verifica donde empieza en la cadena el simbolo $ y ese valor es la posicion inicial
	//por lo general la cadena tiene un cr y un lf al inicio y al final
	for (posInicial = 0; posInicial < (int)strlen(DatosGPSRecibidos); posInicial++)
		{
		if (DatosGPSRecibidos[posInicial] == '$')
			{
			break;
			}
		}

	//verifica si los caracteres iniciales del dato recibido equivalen al tipo de dato especificado
	for (pos = 0 + posInicial; pos < (int)strlen(tipo) + posInicial; pos++)
		{
		//compara caracter a caracter de ser diferente retorna falso
		if (DatosGPSRecibidos[pos] != tipo[pos - posInicial])
			{
			return false;
			}
		}
	//en caso de ser iguales
	return true;
	}
void DatosGPS::datosArduinoGPS(char *datosRecibidos, int size)
{

	//usada para el token
	char *token;

	char *tempToken;

	StringMejorado sm;

	char datoTemporal[50];

	//instruccion para separar por tokens $GPGGA
	token = strtok(datosRecibidos, ",");

	//instruccion para separar por tokens satelites usados
	token = strtok(NULL, ",");

	this->setSatelitesUsados(token);

	//instruccion para separar por tokens latitud en decimal
	token = strtok(NULL, ",");

	//cambia el . por , para trabajar el decimal
	tempToken=sm.normalizarDatos(token, (char*)".", (char*)",",strlen(token));

	//se guarda la latitud en el atributo de la clase DatosGPS
	this->setLatitudDecimal(atof(tempToken));

	//instruccion para separar por tokens longitud en decimal
	token = strtok(NULL, ",");

	//cambia el . por , para trabajar el decimal
```

```
tempToken=sm.normalizarDatos(token, (char*)".", (char*)",",strlen(token));

//se guarda la longitud en el atributo de la clase DatosGPS
this->setLongitudDecimal(atof(tempToken));

//instruccion para separar por tokens la fecha y hora
token = strtok(NULL, ",");

//se guarda la fecha y la hora
this->setHora(token);
}
```

GeograficasPlanas.h Clase construida

```
/*
 * GeograficasPlanas.h
 *
 *  Created on: 16/04/2012
 *      Author: diaz
 */

#ifndef __GeograficasPlanas_H__
#define __GeograficasPlanas_H__

#include <math.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

class GeograficasPlanas
        {
//--------------elementos privados-----------------//
        private:
                //semieje mayor WGS 84
                static const double a = 6378137;

                //semieje menor WGS 84
                static const double b = 6356752.31424518;

                //excentricidad
                double e;

                //segunda excentricidad
                double ep;

                //cuadrado de la segunda excentricidad
                double ep2;

                //radio polar de la curvatura
                double c;

                //longitud en grados minutos y segundos equivalente a lambda
                char *longitudGradosMinutosSegundos; //="73Âº35\'4.1\"W";

                //latitud en grados minutos y segundos equivalente a fi
                char *latitudGradosMinutosSegundos; //="4Âº4\'26.57\"N";

                //longitud en decimal
                double longitudDecimal;

                //latitud en decimal
                double latitudDecimal;

                //valor de PI
                static const double PI = 3.1415926;

                //longitud en radianes
                double longitudRadianes;

                //latitud en radianes
                double latitudRadianes;
```

```
                //zona UTM o huso
                int huso;

                //la letra lambda simboliza el meridiano central del huso
                int lambdaPrima;

                //las letras deltaLambda simbolizan la distancia angular que existe entre la longitud
del punto con el que se
                //opera y el meridiano central del huso
                double deltaLambda;

                double A;

                double xi;

                double eta;

                double ni;

                double zeta;

                double A1;

                double A2;

                double J2;

                double J4;

                double J6;

                double alfa;

                double beta;

                double gamma1;

                double Bfi;

                //coordenada x en planas
                double X;

                //coordenada y en planas
                double Y;

//----------otros----------//

                //calculos de la geometria del elipsoide
                void geometriaElipsoide(void);

                //calculos longitud y latitud
                void longitudLatitud(void);

                //convierte coordenadas grados minutos segundos a decimal
                double coordenadaGeograficaDecimal(char *coordenda);

                //convierte coordenadas decimal a grados minutos segundos
                char *coordenadaDecimalGeografica(char *coordendas);
```

```
                //convierte coordenadas decimal a grados minutos segundos
                char *coordenadaDecimalGeografica(double coordendas);

                //convierte coordenadas decimal (ddmm.mmmm indicador) a grados minutos
segundos
                char *coordenadaDecimalGeografica(char *coordendas, char *indicador);

                //calcula el indicador de la latitud segun signo +N -S
                char *indicadorLatitud(double coordenadas);

                //calcula el indicador de la longitud segun signo +E -O
                char *indicadorLongitud(double coordenadas);

                //convierte coordenadas decimal en radianes
                double coordenadaDecimalRadianes(double coordenada);

                //calcula el signo de la longitud
                double signoLongitud(double coordenada, char *coordenadaGeografica);

                //calculos de la zona UTM o huso
                void zonaUTM(void);

                //calculo ecuaciones de Coticchia-Surace para el problema directo (Geograficas a
UTM)
                void ecuacionesCoticchiaSurace(void);

//--------------elementos publicos-----------------//
        public:

//--------------set-----------------//

                void setLongitudGradosMinutosSegundos(char *longitud);
                void setLatitudGradosMinutosSegundos(char *latitud);
                void setLongitudDecimal(double longitudDecimal);
                void setLatitudDecimal(double latitudDecimal);
                void setLongitudRadianes(double longitudRadianes);
                void setLatitudRadianes(double latitudRadianes);
                void setHuso(int huso);
                void setX(double x);
                void setY(double y);

//--------------get-----------------//

                char *getLongitudGradosMinutosSegundos(void);
                char *getLatitudGradosMinutosSegundos(void);
                double getLongitudDecimal(void);
                double getLatitudDecimal(void);
                double getLongitudRadianes(void);
                double getLatitudRadianes(void);
                int getHuso(void);
                double getX(void);
                double getY(void);

//--------------constructores-----------------//

                //constructor coordenadas decimales
                GeograficasPlanas(char *latitudDecimal, char *longitudDecimal);

                //constructor coordenadas decimales
```

```
		GeograficasPlanas(double latitudDecimal, double longitudDecimal);

		//constructor coordenadas en formato ddmm.mmmm
		GeograficasPlanas(char *latitudDecimal, char *indicadorNS, char *longitudDecimal,
char *indicadorEW);

	//constructor coordenadas grados minutos segundos
	//GeograficasPlanas (char longitud[], char latitud[]);

//--------------otros-----------------//

		//imprime todos los parametros
		void imprimirParametros();
	};

#endif
```

GeograficasPlanas.cpp

```
/*
 * GeograficasPlanas.cpp
 *
 *  Created on: 16/04/2012
 *      Author: diaz
 */

#include "StringMejorado.h"

#include "GeograficasPlanas.h"

//--------------constructores-----------------//

GeograficasPlanas::GeograficasPlanas(char *latitudDecimal, char *indicadorNS, char
*longitudDecimal, char *indicadorEW)
	{
	//calcula la latitud en grados minutos segundos a partir del decimal
	this->latitudGradosMinutosSegundos = this-
>coordenadaDecimalGeografica(latitudDecimal, indicadorNS);

	//calcula la longitud en grados minutos segundos a partir del decimal
	this->longitudGradosMinutosSegundos = this-
>coordenadaDecimalGeografica(longitudDecimal, indicadorEW);

	//calcula geometria del elipsoide
	this->geometriaElipsoide();

	//calcula longitud latitud en radianes y signo de la longitud
	this->longitudLatitud();

	//calcula la zona utm
	this->zonaUTM();

	//calculas ecuaciones de coticchia surace
	this->ecuacionesCoticchiaSurace();

	}

GeograficasPlanas::GeograficasPlanas(char *latitudDecimal, char *longitudDecimal)
	{
	//calcula la latitud en grados minutos segundos a partir del decimal
```

```
	this->latitudGradosMinutosSegundos = this-
>coordenadaDecimalGeografica(latitudDecimal);

	//calcula la longitud en grados minutos segundos a partir del decimal
	this->longitudGradosMinutosSegundos = this-
>coordenadaDecimalGeografica(longitudDecimal);

	//calcula geometria del elipsoide
	this->geometriaElipsoide();

	//calcula longitud latitud en radianes y signo de la longitud
	this->longitudLatitud();

	//calcula la zona utm
	this->zonaUTM();

	//calculas ecuaciones de coticchia surace
	this->ecuacionesCoticchiaSurace();
	}

GeograficasPlanas::GeograficasPlanas(double latitudDecimal, double longitudDecimal)
	{
	//printf ("%s%s\n", this->coordenadaDecimalGeografica(latitudDecimal),
indicadorLatitud(latitudDecimal));

	char *latitud, *longitud;

	//calcula la latitud en grados minutos segundos a partir del decimal
	sprintf(latitud,"%s%s",this-
>coordenadaDecimalGeografica(latitudDecimal),indicadorLatitud(latitudDecimal));

	this->latitudGradosMinutosSegundos=latitud;

	//calcula la longitud en grados minutos segundos a partir del decimal
	sprintf(longitud,"%s%s",this-
>coordenadaDecimalGeografica(longitudDecimal),indicadorLongitud(longitudDecimal));

	this->longitudGradosMinutosSegundos=longitud;

	//calcula geometria del elipsoide
	this->geometriaElipsoide();

	//calcula longitud latitud en radianes y signo de la longitud
	this->longitudLatitud();

	//calcula la zona utm
	this->zonaUTM();

	//calculas ecuaciones de coticchia surace
	this->ecuacionesCoticchiaSurace();
	}


//--------------set-----------------//

void GeograficasPlanas::setLongitudGradosMinutosSegundos(char
*longitudGradosMinutosSegundos)
	{
	strcpy(this->longitudGradosMinutosSegundos, longitudGradosMinutosSegundos);
```

```
	}

void GeograficasPlanas::setLatitudGradosMinutosSegundos(char
*latitudGradosMinutosSegundos)
	{
	strcpy(this->latitudGradosMinutosSegundos, latitudGradosMinutosSegundos);
	}

void GeograficasPlanas::setLongitudDecimal(double longitudDecimal)
	{
	this->longitudDecimal = longitudDecimal;
	}

void GeograficasPlanas::setLatitudDecimal(double latitudDecimal)
	{
	this->latitudDecimal = latitudDecimal;
	}

void GeograficasPlanas::setLongitudRadianes(double longitudRadianes)
	{
	this->longitudRadianes = longitudRadianes;
	}

void GeograficasPlanas::setLatitudRadianes(double latitudRadianes)
	{
	this->latitudRadianes = latitudRadianes;
	}

void GeograficasPlanas::setHuso(int huso)
	{
	this->huso = huso;
	}

void GeograficasPlanas::setX(double x)
	{
	this->X = x;
	}

void GeograficasPlanas::setY(double y)
	{
	this->Y = y;
	}

//--------------get-----------------//
char *GeograficasPlanas::getLongitudGradosMinutosSegundos(void)
	{
	return this->longitudGradosMinutosSegundos;
	}

char *GeograficasPlanas::getLatitudGradosMinutosSegundos(void)
	{
	return this->latitudGradosMinutosSegundos;
	}

double GeograficasPlanas::getLongitudDecimal(void)
	{
	return this->longitudDecimal;
	}
```

```
double GeograficasPlanas::getLatitudDecimal(void)
	{
	return this->latitudDecimal;
	}

double GeograficasPlanas::getLongitudRadianes(void)
	{
	return this->longitudRadianes;
	}

double GeograficasPlanas::getLatitudRadianes(void)
	{
	return this->latitudRadianes;
	}

int GeograficasPlanas::getHuso(void)
	{
	return this->huso;
	}

double GeograficasPlanas::getX(void)
	{
	return this->X;
	}

double GeograficasPlanas::getY(void)
	{
	return this->Y;
	}

//--------------otros-----------------//

void GeograficasPlanas::geometriaElipsoide(void)
	{
	//calculo de la excentricidad
	this->e = sqrt(pow(this->a, 2) - pow(this->b, 2)) / this->a;

	//calculo de la segunda excentricidad
	this->ep = sqrt(pow(this->a, 2) - pow(this->b, 2)) / this->b;

	//calculo del cuadrado de la segunda excentricidad
	this->ep2 = pow(this->ep, 2);

	//calculo radio polar de la curvatura
	this->c = pow(this->a, 2) / this->b;

	//calculo del aplanamiento
	//alfa=(a-b)/a;
	}

void GeograficasPlanas::longitudLatitud(void)
	{
	//convierte de grados minutos y segundos a decimal
	this->latitudDecimal = this->coordenadaGeograficaDecimal(this-
>latitudGradosMinutosSegundos);

	//convierte de grados minutos y segundos a decimal
	this->longitudDecimal = this->coordenadaGeograficaDecimal(this-
>longitudGradosMinutosSegundos);
```

```
	//convierte de decimal a radianes
	this->longitudRadianes = coordenadaDecimalRadianes(this->longitudDecimal);

	//convierte de decimal a radianes
	this->latitudRadianes = coordenadaDecimalRadianes(this->latitudDecimal);

	//determina el signo de la longitud
	this->longitudDecimal = signoLongitud(this->longitudDecimal, this-
>longitudGradosMinutosSegundos);

	this->longitudRadianes = signoLongitud(this->longitudRadianes, this-
>longitudGradosMinutosSegundos);
	}

double GeograficasPlanas::coordenadaGeograficaDecimal(char *coordenada)
	{
	//usada para el token y separar en grados minutos y segundos
	char *coordenadaTexto;

	char tempCoordenada[50];

	strcpy(tempCoordenada, coordenada);

	//guarda la coordenada en decimal
	double decimal = 0;

	//usada para generar los divisores 1, 60 y 3600, potencias de 60
	int potencia = 0;

	//instruccion para separar po tokens
	coordenadaTexto = strtok(tempCoordenada, "Âº,\"'");

	//cuando longitudTexto sea null es por que no encontro mas tokens
	while (coordenadaTexto != NULL)
		{
		//realiza las operaciones y acumula la coordenada en decimal
		decimal = decimal + (atof(coordenadaTexto) / pow(60, potencia));

		//aumenta el valor de la potencia
		potencia++;
		//busca el siguiente token
		coordenadaTexto = strtok(NULL, "Âº,\"'");
		}
	return decimal;
	}

//calcula el indicador de la latitud segun signo +N -S
char *GeograficasPlanas::indicadorLatitud(double coordenadas)
	{
	if (coordenadas<0)
		{
		return (char *)"S";
		}
	else
		{
		return (char *)"N";
		}
	}
```

```
//calcula el indicador de la longitud segun signo +E -O
char *GeograficasPlanas::indicadorLongitud(double coordenadas)
	{
	if (coordenadas<0)
		{
		return (char *)"O";
		}
	else
		{
		return (char *)"E";
		}
	}

char *GeograficasPlanas::coordenadaDecimalGeografica(char *coordenada)
	{

	//usada para el token y separar en grados minutos y segundos
	char *coordenadaTexto;

	//usada para no modificar la coordenada original
	char tempCoordenada[50];

	//los grados de la coordenada
	int grados = 0;

	//los minutos de la coordenada
	int minutos = 0;

	//los segundos de la coordenada
	double segundos = 0;

	//parte decimal de la coordenada
	double valor = 0;

	//la coordenada en grados minutos segundos
	char *resultado = new char[50];

	//saca una copia de la coordenada original
	strcpy(tempCoordenada, coordenada);

	//instruccion para separar por tokens
	coordenadaTexto = strtok(tempCoordenada, " ");

	//extrae la parte entera de la coordenada equivalente a los grados
	grados = (int) atof(coordenadaTexto);

	//extrae la parte decimal de la coordenda
	valor = (atof(coordenadaTexto) - grados) * 60;

	//extrae la parte entera de la coordenada equivalente a los minutos
	minutos = (int) valor;

	//equivalente a los segundos
	segundos = (valor - minutos) * 60;

	//cuando coordenadaTexto sea null es por que no encontro mas tokens
	while (coordenadaTexto != NULL)
		{
```

```
                //convierte a string las variables y concatena
                sprintf(resultado, "%iÂº%i\'%f\"%s", grados, minutos, segundos, tempCoordenada);

                //busca el siguiente token
                coordenadaTexto = strtok(NULL, " ");
                }

        return resultado;
        }

char *GeograficasPlanas::coordenadaDecimalGeografica(double coordenada)
        {
        //los grados de la coordenada
        int grados = 0;

        //los minutos de la coordenada
        int minutos = 0;

        //los segundos de la coordenada
        double segundos = 0;

        //parte decimal de la coordenada
        double valor = 0;

        //la coordenada en grados minutos segundos
        char *resultado = new char[50];

        //si el valor es negativo volverlo positivo
        if (coordenada<0)
                {
                coordenada=coordenada*-1;
                }


        //extrae la parte entera de la coordenada equivalente a los grados
        grados = (int) coordenada;

        //extrae la parte decimal de la coordenda
        valor = (coordenada - grados) * 60;

        //extrae la parte entera de la coordenada equivalente a los minutos
        minutos = (int) valor;

        //equivalente a los segundos
        segundos = (valor - minutos) * 60;


        //convierte a string las variables y concatena
        sprintf(resultado, "%iÂº%i\'%.2f\"", grados, minutos, segundos);

        return resultado;
        }

char *GeograficasPlanas::coordenadaDecimalGeografica(char *coordenada, char *indicador)
        {
        StringMejorado sm;
        //la coordenada en grados minutos segundos
        char *resultado = new char[50];
```

```
	//convierte a string las variables y concatena
	sprintf(resultado, "%sÂº%s\'%f\"%s", sm.substr(coordenada, 0, 2), sm.substr(coordenada,
2, 2), atof(sm.substr(coordenada, 5, 4)) * 0.006, indicador);

	return resultado;
	}

double GeograficasPlanas::coordenadaDecimalRadianes(double coordenada)
	{
	return coordenada * PI / 180;
	}

double GeograficasPlanas::signoLongitud(double coordenada, char *coordenadaGeografica)
	{
	if (strpbrk(coordenadaGeografica, "W"))
		{
		return coordenada * (-1);
		}
	else
		{
		return coordenada;
		}
	}

void GeograficasPlanas::zonaUTM(void)
	{
	//calculo de la zona UTM o huso
	this->huso = (this->longitudDecimal / 6) + 31;

	//calculo del meridiano central del huso
	this->lambdaPrima = this->huso * 6 - 183;

	//calculo de la distancia angular que existe entre la longitud del punto con el que se opera
y el meridiano central del huso
	this->deltaLambda = this->longitudRadianes - (this->lambdaPrima * PI / 180);
	}

void GeograficasPlanas::ecuacionesCoticchiaSurace(void)
	{
	this->A = cos(this->latitudRadianes) * sin(this->deltaLambda);

	this->xi = 0.5 * log((1 + this->A) / (1 - this->A));

	this->eta = atan(tan(this->latitudRadianes) / cos(this->deltaLambda)) - this-
>latitudRadianes;

	this->ni = (this->c / pow((1 + this->ep2 * pow(cos(this->latitudRadianes), 2)), 0.5)) * 0.9996;

	this->zeta = (this->ep2 / 2) * pow(this->xi, 2) * pow(cos(this->latitudRadianes), 2);

	this->A1 = sin(2 * this->latitudRadianes);

	this->A2 = this->A1 * pow(cos(this->latitudRadianes), 2);

	this->J2 = latitudRadianes + (this->A1 / 2);

	this->J4 = (3 * this->J2 + this->A2) / 4;

	this->J6 = (5 * this->J4 + this->A2 * pow(cos(this->latitudRadianes), 2)) / 3;
```

```
	this->alfa = (double) 3 / (double) 4 * this->ep2;

	this->beta = (double) 5 / (double) 3 * pow(this->alfa, 2);

	this->gamma1 = (double) 35 / (double) 27 * pow(this->alfa, 3);

	this->Bfi = 0.9996 * this->c * (this->latitudRadianes - this->alfa * this->J2 + this->beta * this->J4 - this->gamma1 * this->J6);

	this->X = this->xi * this->ni * (1 + (this->zeta / 3)) + 500000;

	this->Y = this->eta * this->ni * (1 + this->zeta) + this->Bfi;

	if (strpbrk(this->latitudGradosMinutosSegundos, "S"))
		{
		this->Y = this->Y + 10000000;
		}
	}

void GeograficasPlanas::imprimirParametros(void)
	{
	printf("Semieje mayor:%.15lf\n", this->a);
	printf("Semieje menor:%.15lf\n", this->b);
	printf("Excentricidad:%.15lf\n", this->e);
	printf("Segunda Excentricidad:%.15lf\n", this->ep);
	printf("Segunda Excentricidad al Cuadrado:%.15lf\n", this->ep2);
	printf("Radio Polar de la Curvatura:%.15lf\n", this->c);
	printf("Longitud:%s\n", this->longitudGradosMinutosSegundos);
	printf("Latitud:%s\n", this->latitudGradosMinutosSegundos);
	printf("Longitud Decimal:%.15lf\n", this->longitudDecimal);
	printf("Latitud Decimal:%.15lf\n", this->latitudDecimal);
	printf("Longitud Radianes:%.15lf\n", this->longitudRadianes);
	printf("Latitud Radianes:%.15lf\n", this->latitudRadianes);
	printf("Zona UTM o Huso:%i\n", this->huso);
	printf("Meridiano Central del Huso:%i\n", this->lambdaPrima);
	printf("Distancia Angular:%.15lf\n", this->deltaLambda);
	printf("A:%.15lf\n", this->A);
	printf("xi:%.15lf\n", this->xi);
	printf("eta:%.15lf\n", this->eta);
	printf("ni:%.15lf\n", this->ni);
	printf("zeta:%.15lf\n", this->zeta);
	printf("A1:%.15lf\n", this->A1);
	printf("A2:%.15lf\n", this->A2);
	printf("J2:%.15lf\n", this->J2);
	printf("J4:%.15lf\n", this->J4);
	printf("J6:%.15lf\n", this->J6);
	printf("alfa:%.15lf\n", this->alfa);
	printf("beta:%.15lf\n", this->beta);
	printf("gamma:%.15lf\n", this->gamma1);
	printf("Bfi:%.15lf\n", this->Bfi);
	printf("X:%.15lf\n", this->X);
	printf("Y:%.15lf\n", this->Y);
	}
```

SerieLinux.h Clase Adaptada

```
/*
 *  LnxCOMM is a Serial Port Library
 *  Copyright (C) 2008 2010 Fernando Pujaico Rivera <fernando.pujaico.rivera@gmail.com>
 *
 *  This program is free software; you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation; either version 2, or (at your option)
 *  any later version.
 *
 *  This program is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with this program; if not, write to the Free Software
 *  Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA.
 *
 */

/** \file
 *  \brief Archivo usado para SO Linux.
 *
 *  Este archivo contiene la definiciÃ³n de las funciones de
 *  de acceso al puerto serie, destinadas a un S.O. tipo Gnu-Linux
 *
 */

/** \defgroup HeaderLinux Funciones para Gnu-Linux
 * @{
 */

#ifndef __serielinux_H__
#define __serielinux_H__

#include <stdio.h>   /* Standard input/output definitions. */
#include <string.h>  /* String function definitions. */
#include <unistd.h>  /* UNIX standard function definitions. */
#include <fcntl.h>   /* File control definitions. */
#include <sys/ioctl.h>
#include <sys/time.h>

#include <termios.h> /* POSIX terminal control definitions. */

#ifndef FALSE
#define FALSE 0
#endif

#ifndef TRUE
#define TRUE  1
#endif

#ifndef BOOL
#define BOOL  int
#endif

#define INVALID_HANDLE_VALUE -1
#define NONE      0
```

```
#define RTSCTS    1
#define HARD      1
#define XONXOFF   2
#define SOFT      2


typedef struct termios DCB;

typedef int HANDLE;

class SerieLinux
        {

        BOOL ERROR_CONFIGURE_PORT;

        public:

                HANDLE Open_Port(char COMx[]);

                DCB Get_Configure_Port(HANDLE fd);

                DCB Configure_Port(HANDLE fd, unsigned int BaudRate, char CharParity[]);

                int Set_Configure_Port(HANDLE fd, DCB newtio);

                long Write_Port(HANDLE fd, char Data[], int SizeData);

                long Read_Port(HANDLE fd, char *Data, int SizeData);

                long Gets_Port(HANDLE fd, char *Data, int SizeData);

                long Getc_Port(HANDLE fd, char *Data);

                int Kbhit_Port(HANDLE fd);

                int Close_Port(HANDLE fd);

                int Set_Hands_Haking(HANDLE fd, int FlowControl);

                int Set_RThreshold(HANDLE fd, int n);

                int Set_BaudRate(HANDLE fd, unsigned int BaudRate);

                int Set_Time(HANDLE fd, unsigned int Time);

                int IO_Blocking(HANDLE fd, int Modo);

                int Clean_Buffer(HANDLE fd);

                int Setup_Buffer(HANDLE fd, unsigned long InQueue, unsigned long OutQueue);

        };

#endif
```

SerieLinux.cpp

```
#include "SerieLinux.h"

/**
```

```
 * @}
 */

/** \fn HANDLE Open_Port(char COMx[])
 *  \brief Abre el puerto de comunicaciones.
 *  \param COMx Es el puerto a abrir.<br>
 *  En Gnu-Linux: "/dev/ttyS0","/dev/ttyS1","/dev/ttyACM0","/dev/ttyUSB0",... <br>
 *  En Windows: "COM1","COM2","COM3","COM4",... <br>
 *  \return El manejador del puerto, en caso de error devuelve INVALID_HANDLE_VALUE.
 *  \ingroup HeaderLinux
 */


HANDLE SerieLinux::Open_Port(char COMx[])
        {
        int fd; // File descriptor for the port.

        ERROR_CONFIGURE_PORT=FALSE;

        fd = open(COMx, O_RDWR | O_NOCTTY); //| O_NDELAY);

        if (fd < 0)
                {
                printf("open_port:fd=%d: No se puede abrir %s\n", fd, COMx);
                return INVALID_HANDLE_VALUE;
                }
        printf("open_port:fd=%d: Abierto puerto %s\n", fd, COMx);
        tcflush(fd, TCIOFLUSH);
        return (fd);
        }

/** \fn DCB Get_Configure_Port(HANDLE fd)
 *  \brief Devuelve la configuraciÃ³n actual del Puerto serie.
 *  \param fd Es el manejador del puerto.
 *  \return Una estructura de tipo DCB con la configuraciÃ³n actual del puerto
 *  serie, ademÃ¡s carga ERROR_CONFIGURE_PORT con FALSE, en caso de error carga
 *  ERROR_CONFIGURE_PORT con TRUE.
 *  \ingroup HeaderLinux
 */
DCB SerieLinux::Get_Configure_Port(HANDLE fd)
        {
        struct termios oldtio;
        if (tcgetattr(fd, &oldtio) != 0) /* almacenamos la configuraciÃ³n actual del puerto */
                {
                printf("Error pidiendo la configuraciÃ³n de puerto serie.\n");
                ERROR_CONFIGURE_PORT = TRUE;
                return oldtio;
                }
        ERROR_CONFIGURE_PORT = FALSE;
        return oldtio;
        }

/** \fn DCB Configure_Port(HANDLE fd,unsigned int BaudRate,char CharParity[])
 *  \brief Configura el puerto serie.
 *  \param fd Es el manejador del puerto.
 *  \param BaudRate Es la velocidad en baudios del puerto.
 *  \param CharParity Indica el nÃºmero de bits en la transmisiÃ³n y la paridad
"8N1","8E1","7E1","7O1" y "7S1".
 *  \return Una estructura de tipo DCB con la configuraciÃ³n del puerto
```

```
 *  serie, ademÃ¡s carga ERROR_CONFIGURE_PORT con FALSE, en caso de error carga
 *  ERROR_CONFIGURE_PORT con TRUE.
 *  \ingroup HeaderLinux
 */

DCB SerieLinux::Configure_Port(HANDLE fd, unsigned int BaudRate, char CharParity[])
        {
        DCB newtio;
        bzero(&newtio, sizeof(newtio)); //limpiamos struct para recibir los
                                                                        //nuevos parÃ¡metros del
puerto.
        //tcflush(fd, TCIOFLUSH);

        //CLOCAL  : conexion local, sin control de modem.
        //CREAD   : activa recepcion de caracteres.
        newtio.c_cflag = CLOCAL | CREAD;

        cfsetispeed(&newtio, BaudRate);
        cfsetospeed(&newtio, BaudRate);

        if (strncmp(CharParity, "8N1", 3) == 0) //CS8     : 8n1 (8bit,no paridad,1 bit de parada)
                {
                newtio.c_cflag &= ~PARENB;
                newtio.c_cflag &= ~CSTOPB;
                newtio.c_cflag &= ~CSIZE;
                newtio.c_cflag |= CS8;
                }
        if (strncmp(CharParity, "8E1", 3) == 0)
                {
                newtio.c_cflag |= PARENB;
                newtio.c_cflag &= ~PARODD;
                newtio.c_cflag &= ~CSTOPB;
                newtio.c_cflag &= ~CSIZE;
                newtio.c_cflag |= CS8;
                }
        if (strncmp(CharParity, "7E1", 3) == 0)
                {
                newtio.c_cflag |= PARENB;
                newtio.c_cflag &= ~PARODD;
                newtio.c_cflag &= ~CSTOPB;
                newtio.c_cflag &= ~CSIZE;
                newtio.c_cflag |= CS7;
                }
        if (strncmp(CharParity, "7O1", 3) == 0)
                {
                newtio.c_cflag |= PARENB;
                newtio.c_cflag |= PARODD;
                newtio.c_cflag &= ~CSTOPB;
                newtio.c_cflag &= ~CSIZE;
                newtio.c_cflag |= CS7;
                }
        if (strncmp(CharParity, "7S1", 3) == 0)
                {
                newtio.c_cflag &= ~PARENB;
                newtio.c_cflag &= ~CSTOPB;
                newtio.c_cflag &= ~CSIZE;
                newtio.c_cflag |= CS8;
                }
```

```
        // IGNPAR  : ignora los bytes con error de paridad.
        // ICRNL   : mapea CR a NL (en otro caso una entrada CR del otro ordenador
        // no terminarÃ-a la entrada) en otro caso hace un dispositivo en bruto
        // (sin otro proceso de entrada).
        newtio.c_iflag = 0;
        //newtio.c_iflag = IGNPAR;
        //newtio.c_iflag |= ICRNL;

        //Salida en bruto.
        newtio.c_oflag = 0;

        //ICANON  : activa entrada canÃ³nica(Modo texto).
        //desactiva todas las funcionalidades del eco, y no envÃ-a seÃ±ales al
        //programa llamador.
        //newtio.c_lflag = ICANON;
        newtio.c_lflag = 0;

        // inicializa todos los caracteres de control.
        // Los valores por defecto se pueden encontrar en /usr/include/termios.h,
        // y vienen dadas en los comentarios, pero no los necesitamos aquÃ-.

        newtio.c_cc[VTIME] = 0; /* temporizador entre caracter, no usado */
        newtio.c_cc[VMIN] = 1; /* bloquea lectura hasta llegada de un caracter */

        if (tcsetattr(fd, TCSANOW, &newtio) != 0)
                {
                printf("ERROR: No se pudo poner la configuraciÃ³n del puerto serie\n");
                ERROR_CONFIGURE_PORT = TRUE;
                return newtio;
                }

        return newtio;
        }

/** \fn int Set_Configure_Port(HANDLE fd,DCB PortDCB)
 *  \brief Coloca la configuraciÃ³n en el puerto serie a partir de una estructura DCB.
 *  \param fd Es el manejador del puerto.
 *  \param PortDCB Es la configuraciÃ³n del puerto.
 *  \return TRUE Si todo fue bien o FALSE si no lo fue.
 *  \ingroup HeaderLinux
 */

int SerieLinux::Set_Configure_Port(HANDLE fd, DCB newtio)
        {
        // ahora limpiamos el buffer de entrada y salida del modem y activamos
        // la configuraciÃ³n del puerto
        //tcflush(fd, TCIOFLUSH);

        if (tcsetattr(fd, TCSANOW, &newtio) != 0)
                {
                printf("ERROR: No se pudo poner configuraciÃ³n del puerto serie\n");
                ERROR_CONFIGURE_PORT = TRUE;
                return FALSE;
                }
        ERROR_CONFIGURE_PORT = FALSE;

        return TRUE;
        }
```

```
/** \fn long Write_Port(HANDLE fd,char Data[],int SizeData)
 *  \brief Escribe en el puerto serie.
 *  \param fd Es el manejador del puerto.
 *  \param Data Es el vector de datos a mandar.
 *  \param SizeData es el tamaÃ±o del vector de datos.
 *  \return En caso de Ã©xito, se devuelve el nÃºmero de bytes escritos (cero
 *          indica que no se ha escrito nada). En GNU-LINUX en caso de error se devuelve -1.
 *  \ingroup HeaderLinux
 */
long SerieLinux::Write_Port(HANDLE fd, char Data[], int SizeData)
        {
        return write(fd, Data, SizeData);
        }

/** \fn long Read_Port(HANDLE fd,char *Data,int SizeData)
 *  \brief Recibe datos en el puerto serie.
 *  \param fd Es el manejador del puerto.
 *  \param Data Es el vector donde se almacenarÃ¡n los datos recibidos.
 *  \param SizeData Es el nÃºmero de datos que se desea recibir, este nÃºmero tiene
 *  que ser menor o igual que el tamaÃ±o del vector Data.
 *  \return En caso de Ã©xito, se devuelve el nÃºmero de bytes recibidos (cero
 *          indica que no se ha recibido nada). En GNU-LINUX en caso de error, se devuelve -1.
 *  \ingroup HeaderLinux
 */

long SerieLinux::Read_Port(HANDLE fd, char *Data, int SizeData)
        {
        struct termios newtio;
        struct timeval inic, fin, temp;
        float tiempo, t;
        int bytes;
        int ESTADO;
        int TEMPO;

        if (tcgetattr(fd, &newtio) != 0)
                return -1;
        else
                {
                ESTADO = newtio.c_cc[VMIN];
                TEMPO = newtio.c_cc[VTIME];
                if ((ESTADO == 0) && (TEMPO == 0))
                        {
                        return read(fd, Data, SizeData);
                        }
                else
                        {
                        if (TEMPO == 0)
                                {
                                do
                                        {
                                        ioctl(fd, FIONREAD, &bytes);
                                        }while (bytes < SizeData);
                                return read(fd, Data, SizeData);
                                }
                        else
                                {
                                gettimeofday(&inic, NULL);
                                tiempo = newtio.c_cc[VTIME];
                                do
```

```
                                        {
                                        gettimeofday(&fin, NULL);
                                        temp.tv_sec = fin.tv_sec - inic.tv_sec;
                                        temp.tv_usec = fin.tv_usec - inic.tv_usec;
                                        t = ((temp.tv_usec / 1000.0) + temp.tv_sec * 1000.0) /
100.0;
                                        } while ((t < tiempo * SizeData) && (Kbhit_Port(fd) <
SizeData));
                                if (Kbhit_Port(fd) != 0)
                                        return read(fd, Data, SizeData);
                                else
                                        return 0;
                                }
                        }
                }
        }

/** \fn long Gets_Port(HANDLE fd,char *Data,int SizeData)
 *  \brief Recibe datos en el puerto serie,lee hasta encontrar un 0x0A,0x0D
 *         (rellenando el siguiente byte con un cero - "solo si existe")
 *         o hasta completar SizeData caracteres.
 *  \param fd Es el manejador del puerto.
 *  \param Data Es el vector donde se almacenarÃ¡n los datos recibidos.
 *  \param SizeData Es el tamaÃ±o mÃ¡ximo del vector.
 *  \return En caso de Ã©xito, se devuelve el nÃºmero de bytes recibidos (los
 *          nÃºmeros serÃ¡n siempre mayores o iguales a cero).
 *  \ingroup HeaderLinux
 */

long SerieLinux::Gets_Port(HANDLE fd, char *Data, int SizeData)
        {
        struct termios newtio;
        long n = 0, i = 0;
        struct timeval inic, fin, temp;
        float tiempo, t;

        if (tcgetattr(fd, &newtio) != 0)
                return -1;

        for (i = 0; i < SizeData; i++)
                {
                gettimeofday(&inic, NULL);
                tiempo = (float) newtio.c_cc[VTIME];
                do
                        {
                        gettimeofday(&fin, NULL);
                        temp.tv_sec = fin.tv_sec - inic.tv_sec;
                        temp.tv_usec = fin.tv_usec - inic.tv_usec;
                        t = ((temp.tv_usec / 1000.0) + temp.tv_sec * 1000.0) / 100.0;
                        } while ((t < tiempo) && (Kbhit_Port(fd) == 0));

                Data[i] = 0;
                if ((Kbhit_Port(fd) != 0) || (tiempo == 0))
                        read(fd, &Data[i], 1);

                if (((Data[i] == 13) || (Data[i] == 10) || (Data[i] == 0)) && (i != 0))
                        {
                        n = i + 1;
                        if (n < SizeData)
```

```
                                        Data[n] = 0;
                                i = SizeData;
                                }
                        }

                return n;
                }

/** \fn Getc_Port(HANDLE fd,char *Data)
 *  \brief Recibe un caracter en el puerto serie.
 *  \param fd Es el manejador del puerto.
 *  \param Data Es el dato(8-bit) a mandar.
 *  \return En caso de Ã©xito, se devuelve el nÃºmero de bytes recibidos (cero
 *          indica que no se ha recibido nada). En GNU-LINUX en caso de error, se devuelve -1.
 *  \ingroup HeaderLinux
 */

long SerieLinux::Getc_Port(HANDLE fd, char *Data)
        {
        struct termios newtio;
        struct timeval inic, fin, temp;
        float tiempo, t;

        if (tcgetattr(fd, &newtio) != 0)
                return -1;

        gettimeofday(&inic, NULL);
        tiempo = (float) newtio.c_cc[VTIME];
        do
                {
                gettimeofday(&fin, NULL);
                temp.tv_sec = fin.tv_sec - inic.tv_sec;
                temp.tv_usec = fin.tv_usec - inic.tv_usec;
                t = ((temp.tv_usec / 1000.0) + temp.tv_sec * 1000.0) / 100.0;
                } while ((t < tiempo) && (Kbhit_Port(fd) == 0));

        if ((Kbhit_Port(fd) != 0) || (tiempo == 0))
                return read(fd, Data, 1);
        else
                return 0;
        }

/** \fn int Kbhit_Port(HANDLE fd)
 *  \brief Indica el nÃºmero de caracteres disponibles en el buffer de entrada del puerto serie.
 *  \param fd Es el manejador del puerto.
 *  \return El nÃºmero de caracteres en el buffer de entrada del puerto serie.
 *  \ingroup HeaderLinux
 */

int SerieLinux::Kbhit_Port(HANDLE fd)
        {
        int bytes;
        ioctl(fd, FIONREAD, &bytes);
        return bytes;
        }

/** \fn int Close_Port(HANDLE fd)
 *  \brief Cierra el puerto serie.
 *  \param fd Es el manejador del puerto.
```

```
 *  \return TRUE si se ha cerrado el Puerto y FALSE en el caso contrario.
 *  \ingroup HeaderLinux
 */

int SerieLinux::Close_Port(HANDLE fd)
	{
	if (fd != INVALID_HANDLE_VALUE)
		{ // Close the communication port.
		// Ahora limpiamos el buffer de entrada y salida del puerto y activamos
		// la configuraciÃ³n del puerto.
		//tcflush(fd, TCIOFLUSH);
		if (close(fd) != 0)
			{
			printf("Error cerrando el puerto serie\n");
			return FALSE;
			}
		else
			{
			fd = INVALID_HANDLE_VALUE;
			return TRUE;
			}
		}
	return FALSE;
	}

/** \fn int Set_Hands_Haking(HANDLE fd,int FlowControl)
 *  \brief Configura el control de flujo en el puerto serie.
 *  \param fd Es el manejador del puerto.
 *  \param FlowControl
 *		0	Ninguno<br>
 *		1	RTS/CTS<br>
 *		2	Xon/Xoff<br>
 *		3	DTR/DSR
 *  \return TRUE si todo fue bien y FALSE si no lo fue.
 *  \ingroup HeaderLinux
 */

int SerieLinux::Set_Hands_Haking(HANDLE fd, int FlowControl)
	{
	struct termios newtio;
	tcgetattr(fd, &newtio); /* almacenamos la configuraciÃ³n actual del puerto */
	switch (FlowControl)
		{
		case 0: //NONE
			{
			newtio.c_cflag &= ~CRTSCTS;
			newtio.c_iflag &= ~(IXON | IXOFF | IXANY);
			newtio.c_cc[VSTART] = 0; /* Ctrl-q */
			newtio.c_cc[VSTOP] = 0; /* Ctrl-s */
			break;
			}
		case 1: //RTS/CTS - HARD
			{
			newtio.c_cflag |= CRTSCTS;
			newtio.c_iflag &= ~(IXON | IXOFF | IXANY);
			newtio.c_cc[VSTART] = 0; /* Ctrl-q */
			newtio.c_cc[VSTOP] = 0; /* Ctrl-s */
			break;
			}
```

```
                case 2: //XON/XOFF - SOFT
                        {
                        newtio.c_cflag &= ~CRTSCTS;
                        newtio.c_iflag |= (IXON | IXOFF); //| IXANY);
                        newtio.c_cc[VSTART] = 17; /* Ctrl-q */
                        newtio.c_cc[VSTOP] = 19; /* Ctrl-s */
                        break;
                        }
                }
        tcsetattr(fd, TCSANOW, &newtio);
        return 0;
        }

/** \fn int Set_RThreshold(HANDLE fd,int n)
 *  \brief configura el numero mÃ-nimo de caracteres que permitirÃ¡
 *         que se ejecute la lectura del puerto
 *  \param fd Es el manejador del puerto.
 *  \param n  Es el numero de caracteres que activarÃ¡ la lectura.
 *  \return TRUE si todo fue bien y FALSE si no lo fue.
 *  \ingroup HeaderLinux
 */

int SerieLinux::Set_RThreshold(HANDLE fd, int n)
        {
        return TRUE;
        }

/** \fn int Set_BaudRate(HANDLE fd,unsigned int BaudRate)
 *  \brief Configura la  velocidad puerto serie.
 *  \param fd Es el manejador del puerto.
 *  \param BaudRate Es la velocidad en baudios del puerto.
 *  \return TRUE si todo fue bien y FALSE si no lo fue.
 *  \ingroup HeaderLinux
 */

int SerieLinux::Set_BaudRate(HANDLE fd, unsigned int BaudRate)
        {
        struct termios newtio;

        if (tcgetattr(fd, &newtio) != 0)
                {
                printf("Error obteniendo configuraciÃ³n del puerto\n");
                return FALSE;
                }

        cfsetispeed(&newtio, BaudRate);
        cfsetospeed(&newtio, BaudRate);

        if (tcsetattr(fd, TCSANOW, &newtio) != 0)
                {
                printf("Error configurando el BaudRate\n");
                return FALSE;
                }

        return TRUE;
        }

/** \fn int Set_Time(HANDLE fd,unsigned int Time)
 *  \brief Configura temporizador para la lectura y escritura en el puerto serie.
```

```
 *  \param fd Es el manejador del puerto.
 *  \param Time Tiempo (T) entre bits, T=Time*0.1s, para tamaÃ±o total de time-out
 *              en la lectura y escritura.<br>
 *              Timeout = (Time *0.1* number_of_bytes) seg
 *  \return TRUE si todo fue bien y FALSE si no lo fue.
 *  \ingroup HeaderLinux
 */

int SerieLinux::Set_Time(HANDLE fd, unsigned int Time) //t =Time*0.1 s)
        {
        struct termios newtio;
        /* almacenamos la configuracion actual del puerto */
        if (tcgetattr(fd, &newtio) != 0)
                {
                printf("Error obteniendo configuraciÃ³n time-out actual\n");
                return FALSE;
                }

        newtio.c_cc[VTIME] = Time;/*temporizador entre caracter*/
        newtio.c_cc[VMIN] = 1; /*bloquea lectura hasta llegada de un caracter  */

        if (tcsetattr(fd, TCSANOW, &newtio) != 0)
                {
                printf("Error estableciendo nueva configuraciÃ³n time-out\n");
                return FALSE;
                }

        return TRUE;
        }

/** \fn int IO_Blocking(HANDLE fd,int Modo)
 *  \brief Configura si la lectura y escritura de datos se ejecutarÃ¡ en modo bloqueante.
 *  \param fd Es el manejador del puerto.
 *  \param Modo Escoge el tipo de bloqueo.<br>
 *   TRUE : Modo bloqueante.<br>
 *   FALSE: Modo no bloqueante.
 *  \return TRUE si todo fue bien y FALSE si no lo fue.
 *  \ingroup HeaderLinux
 */
int SerieLinux::IO_Blocking(HANDLE fd, int Modo)
        {
        struct termios newtio;
        /* almacenamos la configuracion actual del puerto */
        if (tcgetattr(fd, &newtio) != 0)
                {
                printf("Error obteniendo configuraciÃ³n time-out actual\n");
                return FALSE;
                }

        if (Modo == FALSE)
                {
                newtio.c_cc[VTIME] = 0; /* Temporizador entre caracter.*/
                newtio.c_cc[VMIN] = 0; /* No bloquea lectura hasta llegada de un caracter. */
                }
        if (Modo == TRUE)
                {
                newtio.c_cc[VTIME] = 0; /* Temporizador entre caracter.*/
                newtio.c_cc[VMIN] = 1; /* Bloquea lectura hasta llegada de un caracter. */
                }
```

```
	if (tcsetattr(fd, TCSANOW, &newtio) != 0)
		{
		printf("Error estableciendo nueva configuraciÃ³n bloqueante/no-bloqueante.\n");
		return FALSE;
		}

	return TRUE;
	}

/** \fn int Clean_Buffer(HANDLE fd)
 *  \brief Termina las operaciones de lectura y escritura pendientes y limpia
 *         las colas de recepciÃ³n y de transmisiÃ³n.
 *  \param fd Es el manejador del puerto.
 *  \return TRUE si todo fue bien y FALSE si no lo fue.
 *  \ingroup HeaderLinux
 */

int SerieLinux::Clean_Buffer(HANDLE fd)
	{
	if (tcflush(fd, TCIOFLUSH) != 0)
		{
		printf("Error Limpiando el Buffer  de entrada y salida.\n");
		return FALSE;
		}
	return TRUE;
	}

/** \fn int Setup_Buffer(HANDLE fd,unsigned long InQueue,unsigned long OutQueue)
 *  \brief Especifica el tamaÃ±o en Bytes del buffer de entrada y salida.
 *  \param fd Es el manejador del puerto.
 *  \param InQueue Especifica el tamaÃ±o en Bytes del buffer de entrada, se
 *               Recomienda el uso de numero pares.
 *  \param OutQueue Especifica el tamaÃ±o en Bytes del buffer de salida, se
 *               Recomienda el uso de numero pares.
 *  \return TRUE si todo fue bien y FALSE si no lo fue.
 *
 */

int SerieLinux::Setup_Buffer(HANDLE fd, unsigned long InQueue, unsigned long OutQueue)
	{
	return TRUE;
	}

/*****************************************************************************/
#ifdef ENABLE_SERIAL_PORT_EVENT
#include <pthread.h>

void SERIAL_PORT_EVENT(HANDLE * hPort);

void *Thread_Port(void *hPort)
{
	int n=0;
	HANDLE *fd;
	fd=(HANDLE *)hPort;

	printf("SERIAL_PORT_EVENT [OK]\n");

	do {
```

```
                    if(Kbhit_Port(*fd)!=0)
                    SERIAL_PORT_EVENT(fd);
          }while(TRUE);
}
/** \fn pthread_t Create_Thread_Port(HANDLE *fd)
 *  \brief Se usa para crear un hilo que ejecuta la funciÃ³n:<br>
 *         void SERIAL_PORT_EVENT(HANDLE *hPort) <br>
 *         Esta se ejecuta cuando se produce el evento de recepciÃ³n de un
 *         caracter por el puerto serie.
 *  \param fd Es el manejador del puerto serie.
 *  \return El manejador del hilo creado.
 *  \ingroup HeaderLinux
 */
pthread_t Create_Thread_Port(HANDLE *fd)
{
          pthread_t idHilo;
          pthread_create (&idHilo, NULL, Thread_Port, fd);
          return idHilo;
}
#endif
```

StringMejorado.h Clase construida

```
/*
 * StringMejorado.h
 *
 *  Created on: 16/04/2012
 *      Author: diaz
 */

#ifndef __StringMejorado_H__
#define __StringMejorado_H__

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

//tomado de http://www.forosdelweb.com/f96/substr-para-c-543410/

class StringMejorado
	{
	public:
		char * substr(char *cadena, int comienzo, int longitud);
		char *normalizarDatos(char *datosRecibidos, char *textoCambiar, char
*textoCambio, int sizeDatosRecibidos);

	};

#endif
```

StringMejorado.cpp

```
/*
 * StringMejorado.cpp
 *
 *  Created on: 16/04/2012
 *      Author: diaz
 */

#include "StringMejorado.h"

char *StringMejorado::substr(char *cadena, int comienzo, int longitud)
	{
	if (longitud == 0)
		{
		longitud = strlen(cadena) - comienzo;
		}

	char *nuevo = (char*) malloc(sizeof(char) * (longitud + 1));
	nuevo[longitud] = '\0';
	strncpy(nuevo, cadena + comienzo, longitud);

	return nuevo;
	}

char *StringMejorado::normalizarDatos(char *datosRecibidos, char *textoCambiar, char
*textoCambio, int sizeDatosRecibidos)
	{
	char *pch, *temporal = new char[sizeDatosRecibidos + 20];

	strcpy(temporal, datosRecibidos);
```

```
        pch = strstr(temporal, textoCambiar);

        while (pch != NULL)
                {
                sprintf(temporal, "%s%s%s", substr(temporal, 0, strlen(temporal) - strlen(pch)),
textoCambio, substr(pch, 2, strlen(pch) - 2));
                pch = strstr(temporal, textoCambiar);
                }
        return temporal;
        }
```

VisionArtificial.h Clase construida

```
/*
 * VisionArtificial.h
 *
 *  Created on: 15/04/2012
 *      Author: diaz
 */

#include <stdio.h>

#include "opencv2/opencv.hpp"

using namespace cv;


class VisionArtificial
	{
	public:
/*----------------publico-------------------------*/

	//usado con camaras web o capturadoras de video
	VisionArtificial (int camara);


	//usado con camaras de red o streaming
	VisionArtificial (string camara);

/*----------------get-------------------------*/
	VideoCapture getFlujoVideo (void);
	Mat getCuadroAnteriorBGR (void);
	Mat getCuadroSiguienteBGR (void);
	Mat getCuadroAnteriorGRIS (void);
	Mat getCuadroSiguienteGRIS (void);



/*----------------set-------------------------*/
	void setCuadroAnteriorBGR (Mat cuadroAnteriorBGR);
	void setCuadroSiguienteBGR (Mat cuadroSiguienteBGR);
	void setCuadroAnteriorGRIS (Mat cuadroAnteriorGRIS);
	void setCuadroSiguienteGRIS (Mat cuadroSiguienteGRIS);


/*----------------otros-------------------------*/
	//calcula la diferencia entre la imagen anterior y la siguiente
	Mat diferencia (Mat cuadroAnterior, Mat cuadroSiguiente);

	//captura el cuadro siguiente, el actual lo convierte en el anterior
	Mat capturarCuadroSiguiente (void);

	//cierra el flujo de video
	void cerrarFlujoVideo (void);

	//convierte una imagen bgr a escala de grises
	Mat cuadroBGR_GRIS (Mat imagenBGR);

	//detecta bordes de una imagen en escala de grises
	Mat deteccionBorde (Mat imagenGRIS, int filaKernel, int columnaKernel, float sigmaX, float
sigmaY);
```

```
        //inerta texto en una imagen
        Mat descriptor (Mat imagen, char *mensaje, int valor, int x, int y, Scalar color);

        //inerta texto en una imagen
        Mat descriptor (Mat imagen, char *mensaje, float valor, int x, int y, Scalar color);

        //inerta texto en una imagen
        Mat descriptor (Mat imagen, char *mensaje, char *valor, int x, int y, Scalar color);

        //invierte el color de una imagen recorriendola pixel por pixel
        Mat invertirImagen (Mat imagen);

        //detectar color por canal primario
        Mat detectarColor (Mat imagen, int intensidad, int porcentajeIntensidad);

        //definir parametros para guardar flujo de video
        void parametrosGuardarFlujoVideo (string nombreArchivo, string codec, int fps, Size size);

        //graba el flujo de video
        void grabarFlujoVideo (Mat imagen);


private:
/*----------------privado-------------------------*/
        VideoCapture flujoVideo;
        VideoWriter guardarFlujoVideo;
        Mat cuadroAnteriorBGR;
        Mat cuadroSiguienteBGR;
        Mat cuadroAnteriorGRIS;
        Mat cuadroSiguienteGRIS;
        CvFont fuente;

        string conexion;
        };
```

VisionArtificial.cpp

```
/*
 * VisionArtificial.cpp
 *
 *  Created on: 15/04/2012
 *      Author: diaz
 */

#include "VisionArtificial.h"

/*----------------constructores-------------------------*/
VisionArtificial::VisionArtificial (int camara)
   {
        //definicion de fuente para escritura de descriptores
   cvInitFont(&fuente, CV_FONT_HERSHEY_SIMPLEX, 0.4, 0.4, 0, 1, CV_AA);

   //objeto para captura del flujo de video
   this->flujoVideo.open(camara);

   //verifica si no se pudo abrir el flujo de video
   if(!this->flujoVideo.isOpened())
      {
      printf ("No se pudo abrir el flujo de video");
```

```
        }
    }

//con camaras de red o streaming
VisionArtificial::VisionArtificial (string camara)
    {
            this->conexion=camara;

    //objeto para captura del flujo de video
    this->flujoVideo.open(this->conexion);

    //verifica si no se pudo abrir el flujo de video
    if(!this->flujoVideo.isOpened())
        {
        printf ("No se pudo abrir el flujo de video");
        }
    else
            {
            printf ("Se inicio el flujo de video");
            }
    }

/*----------------get-------------------------*/
VideoCapture VisionArtificial::getFlujoVideo (void)
    {
    return this->flujoVideo;
    }

Mat VisionArtificial::getCuadroAnteriorBGR (void)
    {
    return this->cuadroAnteriorBGR;
    }

Mat VisionArtificial::getCuadroSiguienteBGR (void)
    {
    return this->cuadroSiguienteBGR;
    }

Mat VisionArtificial::getCuadroAnteriorGRIS (void)
    {
    return this->cuadroAnteriorGRIS;
    }

Mat VisionArtificial::getCuadroSiguienteGRIS (void)
    {
    return this->cuadroSiguienteGRIS;
    }

/*----------------set-------------------------*/
void VisionArtificial::setCuadroAnteriorBGR (Mat cuadroAnterior)
    {
    this->cuadroAnteriorBGR=cuadroAnteriorBGR;
    }

void VisionArtificial::setCuadroSiguienteBGR (Mat cuadroSiguienteBGR)
    {
    this->cuadroSiguienteBGR=cuadroSiguienteBGR;
    }
```

```
void VisionArtificial::setCuadroAnteriorGRIS (Mat cuadroAnteriorGRIS)
   {
   this->cuadroAnteriorGRIS=cuadroAnteriorGRIS;
   }

void VisionArtificial::setCuadroSiguienteGRIS (Mat cuadroSiguienteGRIS)
   {
   this->cuadroSiguienteGRIS=cuadroSiguienteGRIS;
   }

/*----------------otros-------------------------*/

Mat VisionArtificial::diferencia (Mat cuadroAnterior, Mat cuadroSiguiente)
   {
   //crea una matriz para el resultado
   Mat resultado;

   //si las imagenes existen procesa
   if (cuadroAnterior.empty()==false)
      {
      //copia la estructura de la matriz anterior a resultado
      //resultado=this->cuadroAnterior;

      //calcula la diferencia absoluta entre la matriz anterior y la actual
      absdiff(cuadroAnterior, cuadroSiguiente, resultado);
      //retorna el resultado
      return resultado;
      }

   return cuadroSiguiente;

   }

Mat VisionArtificial::capturarCuadroSiguiente (void)
   {
   //guarda el cuadro anterior, para ello clona cuadrosiguiente que pasa a ser el cuadroanterior
   this->cuadroAnteriorBGR=this->cuadroSiguienteBGR.clone();

   //convierte a grises el cuadro anterior en bgr
   this->cuadroAnteriorGRIS=this->cuadroBGR_GRIS(this->cuadroAnteriorBGR);

   //captura el cuadro siguiente
   this->flujoVideo>>this->cuadroSiguienteBGR;

   //convierte a grises el cuadro siguiente en bgr
   this->cuadroSiguienteGRIS=this->cuadroBGR_GRIS(this->cuadroSiguienteBGR);

   //retorna el cuadro siguiente
   return this->cuadroSiguienteBGR;
   }

void VisionArtificial::cerrarFlujoVideo (void)
   {
   //cierra el flujo de video
   this->flujoVideo.release();
   }

Mat VisionArtificial::cuadroBGR_GRIS (Mat imagenBGR)
   {
```

```
   Mat resultado;

   //verifica que exista una imagenbgr
   if (imagenBGR.empty()==false)
      {
      //convierte la imagen de bgr a grises
      cvtColor(imagenBGR, resultado, CV_BGR2GRAY);
      }

   return resultado;
   }

Mat VisionArtificial::deteccionBorde (Mat imagenGRIS, int filaKernel, int columnaKernel, float
sigmaX, float sigmaY)
   {
   Mat resultado;

   //suavizacion de imagen utilizando filtro gaussiano
   //imagenorigen, imagendestino, kernell del filtro, sigmax, sigmay
   GaussianBlur(imagenGRIS, resultado, Size(filaKernel,columnaKernel), sigmaX, sigmaY);

   //encuentra los bordes de una imagen utilizando
   //imagenorigen, imagendestino,
   Canny(resultado, resultado, 0, 30, 3);

   return resultado;
   }

Mat VisionArtificial::descriptor (Mat imagen, char *mensaje, int valor, int x, int y, Scalar color)
         {
         //convertir numeros a string para enviar a los descriptores
         char numeroString[12];

         //concatenar el mensaje y el numero al string
         char mensajeString[80];

         //copiar el mensaje a mensajestring
         strcpy(mensajeString, mensaje);

         //convertir la variable numerica a string
         sprintf(numeroString, "%i", valor);

         //concatenar el mensaje con el numero en string
         strcat(mensajeString, numeroString);

         //pegar el texto en la imagen
         //imagen, texto a pegar, coordenada dentro de la imagen, fuente, tam fuente, color letra, ,
         putText(imagen, mensajeString, Point(x,y), CV_FONT_HERSHEY_COMPLEX, 0.5, color,
1, 1);

         return imagen;
         }

Mat VisionArtificial::descriptor (Mat imagen, char *mensaje, float valor, int x, int y, Scalar color)
         {
         //convertir numeros a string para enviar a los descriptores
         char numeroString[12];

         //concatenar el mensaje y el numero al string
```

```
	char mensajeString[80];

	//copiar el mensaje a mensajestring
	strcpy(mensajeString, mensaje);

	//convertir la variable numerica a string
	sprintf(numeroString, "%.2f", valor);

	//concatenar el mensaje con el numero en string
	strcat(mensajeString, numeroString);

	//pegar el texto en la imagen
	//imagen, texto a pegar, coordenada dentro de la imagen, fuente, tam fuente, color letra, ,
	putText(imagen, mensajeString, Point(x,y), CV_FONT_HERSHEY_COMPLEX, 0.5, color,
1, 1);

	return imagen;
	}

Mat VisionArtificial::descriptor (Mat imagen, char *mensaje, char *valor, int x, int y, Scalar color)
	{
	//concatenar el mensaje y el valor string
	char mensajeString[500];

	//copiar el mensaje a mensajestring
	strcpy(mensajeString, mensaje);

	//concatenar el mensaje con el valor string
	strcat(mensajeString, valor);

	//printf ("%s\n",valor);
	//pegar el texto en la imagen
	//imagen, texto a pegar, coordenada dentro de la imagen, fuente, tam fuente, color letra, ,
	putText(imagen, mensajeString, Point(x,y), CV_FONT_HERSHEY_COMPLEX, 0.5, color,
1, 1);

	return imagen;
	}

//invierte el color de una imagen recorriendola pixel por pixel
Mat VisionArtificial::invertirImagen (Mat imagen)
	{
	//almacena los valores de la imagen b,g,r
	unsigned char *datosImagen=(unsigned char *)imagen.data;

	int fila=0, columna=0;

	//for para recorrer las filas
	for (fila=0; fila<imagen.rows; fila++)
		{
		//for para recorrer las columnas
		for (columna=0; columna<imagen.cols; columna++)
			{
			//canal azul b
			datosImagen[fila*imagen.step+columna*imagen.channels()+0]=255-
datosImagen[fila*imagen.step+columna*imagen.channels()+0];

			//canal verde g
```

```
                        datosImagen[fila*imagen.step+columna*imagen.channels()+1]=255-
datosImagen[fila*imagen.step+columna*imagen.channels()+1];

                        //canal rojo r
                        datosImagen[fila*imagen.step+columna*imagen.channels()+2]=255-
datosImagen[fila*imagen.step+columna*imagen.channels()+2];

                        }
                }
        return imagen;
        }

//detectar color por canal primario
Mat VisionArtificial::detectarColor (Mat imagen, int intensidad, int porcentajeIntensidad)
        {
        //almacena los valores de la imagen b,g,r
        unsigned char *datosImagen=(unsigned char *)imagen.data;

        int fila=0, columna=0;

        //for para recorrer las filas
        for (fila=0; fila<imagen.rows; fila++)
                {
                //for para recorrer las columnas
                for (columna=0; columna<imagen.cols; columna++)
                        {

                        //intensidad trabajada y umbral de trabajo
                        if
(datosImagen[fila*imagen.step+columna*imagen.channels()+2]>intensidad &&

        !((datosImagen[fila*imagen.step+columna*imagen.channels()+0]>datosImagen[fila*imagen.
step+columna*imagen.channels()+2]/porcentajeIntensidad)       ||

        (datosImagen[fila*imagen.step+columna*imagen.channels()+1]>datosImagen[fila*imagen.s
tep+columna*imagen.channels()+2]/porcentajeIntensidad)))
                                {
                                //canal azul b
                                datosImagen[fila*imagen.step+columna*imagen.channels()+0]=0;

                                //canal verde g
                                datosImagen[fila*imagen.step+columna*imagen.channels()+1]=0;

                                //canal rojo r

        datosImagen[fila*imagen.step+columna*imagen.channels()+2]=255;
                                }
                        else
                                {
                                //canal azul b
                                datosImagen[fila*imagen.step+columna*imagen.channels()+0]=0;

                                //canal verde g
                                datosImagen[fila*imagen.step+columna*imagen.channels()+1]=0;

                                //canal rojo r
                                datosImagen[fila*imagen.step+columna*imagen.channels()+2]=0;
                                }
                        }
```

```
		}
	return imagen;
	}

//definir parametros para guardar flujo de video
void VisionArtificial::parametrosGuardarFlujoVideo (string nombreArchivo, string codec, int fps,
Size size)
	{

	//listado de codecs
	string codecs[8]={"MPEG-1", //8.3
					"MOTION-JPEG", //11.3
					"MPEG-4.2", //3.9
					"MPEG-4.3", //4.9
					"MPEG-4", //4.6
					"H263", //4.7
					"H263I", //no funciona
					"FLV1" //5.5
					};

	//valores de los codecs
	char valoresCodec[8][4]={{'P','I','M','1'},
						{'M','J','P','G'},
						{'M','P','4','2'},
						{'D','I','V','3'},
						{'D','I','V','X'},
						{'U','2','6','3'},
						{'I','2','6','3'},
						{'F','L','V','1'},
						};

	//ubicacion del codec
	int pos=0, valorCodec=0;

	//busca el codec definido en el listado de codecs y devuelve la posicion
	for (pos=0; pos<8; pos++)
		{
		//compara el codecs de la lista con el codec solicitado
		if (codecs[pos].compare(codec)==0)
			{
			//asigna el valor de la posicion del codec encontrado en el listado de
codecs
			valorCodec=pos;
			break;
			}
		}

	//asigna los parametros para la grabacion del flujo de video
	guardarFlujoVideo = VideoWriter(nombreArchivo,
CV_FOURCC(valoresCodec[valorCodec][0], valoresCodec[valorCodec][1],
					valoresCodec[valorCodec][2],
valoresCodec[valorCodec][3]), fps, size);
	}

//graba el flujo de video
void VisionArtificial::grabarFlujoVideo (Mat imagen)
	{
	guardarFlujoVideo << imagen;
	}
```

Xbee.h Clase construida

```
/*
 * Xbee.h
 *
 *  Created on: 16/04/2012
 *      Author: diaz
 */

#ifndef __Xbee_H__
#define __Xbee_H__

#include "SerieLinux.h"

class Xbee
	{
	private:
		HANDLE fd;
		SerieLinux serial;
		char *datos;

	public:
		void abrirXbee(char *puerto, unsigned int velocidad, char *paridad, int size);
		void escribirXbee(char *datos);
		char *leerXbee(int cantidadCaracteres);
		void cerrarXbee();
	};

#endif
```

Xbee.cpp

```
/*
 * Xbee.cpp
 *
 *  Created on: 16/04/2012
 *      Author: diaz
 */

#include "Xbee.h"

void Xbee::abrirXbee(char *puerto, unsigned int velocidad, char *paridad, int size)
	{
	//abre el puerto serie segun linux
	this->fd = serial.Open_Port(puerto);

	//configura el puerto
	serial.Configure_Port(this->fd, velocidad, paridad);

	//tamaÃ±o de la trama a leer
	datos = new char[size];

	}

void Xbee::escribirXbee(char *datos)
	{
	int n;
	n = serial.Write_Port(this->fd, datos, strlen(datos));
	}

char* Xbee::leerXbee(int cantidadCaracteres)
```

```
	{
	//verifica si el buffer del puerto tiene datos
	if (serial.Kbhit_Port(this->fd) > 0)
		{
		serial.Gets_Port(this->fd, datos, cantidadCaracteres);
		return datos;
		}
	return (char *)"";
	}

void Xbee::cerrarXbee(void)
	{
	//cierra el puerto
	serial.Close_Port(this->fd);
	}
```

A.3. APLICACIÓN EMBEBIDA EN ARDUINO UNO R3

```
/*
shield Wireless SD
---------------------------------------------------------------------------
el interuptor en usb es para programar el arduino
el interuptor en micro es para operar el xbee y la sd sin conexion por usb

Arduino                 Shield Wireless SD
---------------------------------------------------------------------------
Digital pin 4           - CS de la sd
Digital pin 10          - salida del arduino a la SD
Digital pin 0           - RX del Xbee
Digital pin 1           - TX del xbee
si usa la SD no puede usar de los pines del 11 al 13

Arduino                 GPS A2100A
---------------------------------------------------------------------------
Digital pin 7 RX        - TX del GPS
Digital pin 8 TX        - RX del GPS
3.3 V                   - 3.3 V
GND                     - GND

*/

//biblioteca emulacion puertos series
#include <SoftwareSerial.h>

//bibliotecas manejo GPS
#include <TinyGPS.h>

//bibliotecas para manejo de SD
#include <SD.h>

//objeto emula puerto serie por los pines digitales 8(RX) y 9(TX) para el GPS A2100A
SoftwareSerial serialGPS(7, 8);

//trama que llega por el XBEE
/*
pos 0: captura punto GPS
*/
char trama[1];

//variable para determinar el envio
bool envioGPS=false;

//Objeto para manejo de archivos
File archivo;

//objeto para manejo de GPS
TinyGPS gps;

//prototipo de funciones GPS
static void gpsdump(TinyGPS &gps);
static bool feedgps();
static String print_float(float val, float invalid, int len, int prec);
static String print_int(unsigned long val, unsigned long invalid, int len);
```

```
static String print_date(TinyGPS &gps);
static String print_str(const char *str, int len);

//prototipo de funciones SD
void escribirArchivoSD (const char * nombreArchivo, String dato);
void leerArchivoSD (const char * nombreArchivo);

void setup ()
 {
 //velocidad del puerto serie. Xbee configurados a 115200
 Serial.begin(115200);

 //pin de salida del arduino al modulo para la SD
 pinMode(10, OUTPUT);

 //validacion de inicializacion de la SD trabaja con el pin 4 para el CS de la SD
 if (!SD.begin(4))
  {
  Serial.println("Inicializacion fallada!");
  return;
  }
 }

void loop ()
 {
 //verifica si llego algo a la entrada del puerto serie
 //trama de instrucciones desde el dispositivo movil al UAV
 if (Serial.available())
  {
  //lee 1 byte del puerto, tamaño de la trama
  Serial.readBytes(trama,1);

  envioGPS=false;
  }

  if trama[0]=='1')
   {
   envioGPS=true;
   }

 if (envioGPS==true)
  {
  serialGPS.listen();

  bool nuevoDato = false;

  unsigned long tiempoInicial= millis();

  while (millis()-tiempoInicial<2000)
   {
   if (feedgps())
    {
    nuevoDato=true;
    }
   }

  if (nuevoDato==true)
   {
   //llama la funcion que interpreta los datos del gps
```

```
      gpsdump(gps);
      }

    envioGPS=false;
    }
  }


void escribirArchivoSD (const char * nombreArchivo, String dato)
  {
  //abre un archivo para escritura en la SD
  archivo = SD.open(nombreArchivo, FILE_WRITE);

  //si abrio el archivo
  if (archivo)
    {
    //escribe en el puerto serie
    Serial.print("escribiendo en el archivo...");

    //escribe en el archivo
    archivo.println(dato);

    //cerrar archivo
    archivo.close();

    //escribe en el puerto
    Serial.println("Finalizado.");
    }

  //si no abrio el archivo
  else
    {
    //imprime en el puerto
    Serial.println("error al abrir el archivo");
    }
  }

void leerArchivoSD (const char * nombreArchivo)
  {
  //abrir un archivo
  archivo = SD.open(nombreArchivo);

  //si abrio el archivo
  if (archivo)
    {
    //escribe en el puerto serie
    Serial.println("Leyendo archivo");

    //leer todo el archivo
    while (archivo.available())
      {
      //lee y escribe al puerto serie
      Serial.write(archivo.read());
      }

    //cierra el archivo
    archivo.close();
    }
```

```
  //si no abrio el archivo
  else
    {
    //imprime en el puerto serie
    Serial.println("erro abriendo el archivo");
    }
  }

//metodos del GPS
static void gpsdump(TinyGPS &gps)
  {
  float flat, flon;
  unsigned long age, date, time, chars = 0;
  unsigned short sentences = 0, failed = 0;
  static const float LONDON_LAT = 51.508131, LONDON_LON = -0.128002;
  String satelites="", latitud="", longitud="", fixAge="", fechaHora="", trama="";
  char *nombreArchivo="datos.txt";

  satelites=print_int(gps.satellites(), TinyGPS::GPS_INVALID_SATELLITES, 5);

  gps.f_get_position(&flat, &flon, &age);

  latitud=print_float(flat, TinyGPS::GPS_INVALID_F_ANGLE, 9, 5);

  longitud=print_float(flon, TinyGPS::GPS_INVALID_F_ANGLE, 10, 5);

  fechaHora=print_date(gps);

  Serial.println("$ArduinoGPS,"+satelites+","+latitud+","+longitud+","+fechaHora);
  }

static String print_int(unsigned long val, unsigned long invalid, int len)
  {
  char sz[32];
  if (val == invalid)
    {
    strcpy(sz, "*******");
    }
  else
    {
    sprintf(sz, "%ld", val);
    }
  String s =sz;
  return s;
  }

static String print_float(float val, float invalid, int len, int prec)
  {
  char sz[32];
  String s="";
  if (val == invalid)
    {
    strcpy(sz, "*******");
    s=sz;
    }
  else
    {
    //dato float, longitud, precision, char
    dtostrf(val, len, prec, sz);
```

```
    s=sz;
    return s;
    }
  }

static String print_date(TinyGPS &gps)
  {
  //variables para capturar
  int year;
  byte month, day, hour, minute, second, hundredths;
  unsigned long age;
  char sz[32];

  //extraes fecha y hora del GPS
  gps.crack_datetime(&year, &month, &day, &hour, &minute, &second, &hundredths, &age);

  //si los datos son invalidos
  if (age == TinyGPS::GPS_INVALID_AGE)
    {
    strcpy(sz, "********");
    }

  //si los datos son validos
  else
    {
    hour=hour-5;
    if (hour<0)
      {
      hour=18-hour;
      }
    sprintf(sz, "%02d/%02d/%02d %02d:%02d:%02d   ", year, month, day, hour, minute, second);
    }

  String s = sz;
  return s;
  }

static String print_str(const char *str, int len)
  {
  String s="";
  int slen = strlen(str);
  s=str;
  return s;
  }

static bool feedgps()
  {
  while (serialGPS.available())
    {
    if (gps.encode(serialGPS.read()))
      {
      return true;
      }
    }
  return false;
  }
```

A.9. CONTENIDO DEL CD

Adicionales/

Contiene Aplicaciones, Codigo fuente y SDK.

Documentos/

Contiene la tesis en formatos PDF y DOC.

Imágenes/

Imágenes en alta resolución usadas en la tesis.

Pdf/

Contiene los PDFs de los mapas generados.