UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

IMPLEMENTACIÓN DE UN SISTEMA AUTÓNOMO EN UN CUADRICOPTERO MEDIANTE UN PROCESADOR ARM

TESIS
QUE PARA OBTENER EL TÍTULO DE:
INGENIERO ELÉCTRICO ELECTRÓNICO

P R E S E N T A
IRVING LUNA ORTIZ

DIRECTOR DE TESIS
M. en I. MAURICIO ONTIVEROS SALGADO

Cd Nezahualcóyotl, Estado de México, 2018

Dedicatoria

Te quiero mucho y ¿sabes? nunca habrá alguna manera de devolverte lo mucho que me has dado y de verdad es tanto que desde un principio me has dado la vida. Esta tesis es un logro más que llevo a cabo y sin lugar a dudas ha sido en gran parte por tu apoyo incondicional, no sé en donde me encontraría de no ser por tus enseñanzas, tu amor y tu compañía.

Es por eso que lo único que puedo hacer es dedicar esta tesis a una mujer que simplemente me llena de orgullo, ***MUCHAS GRACIAS POR TODO MAMÁ.***

Agradecimientos:

Le agradezco a Dios, arquitecto del universo el cual me ha permitido comprenderlo un poco más y ser parte de su perfecta armonía.

Agradezco a mis padres María Luisa y Antonio por apoyarme en todo momento, por la educación, los valores y su amor incondicional.

Les doy las gracias a mis abuelitos Gustavo y Elena por su infinito amor, sus sabios consejos y enseñanzas.

Les doy gracias a mis hermanos sanguíneos Gustavo Antonio e Itzel Arelly por su cariño y por permitirme ser el hermano qué más quieren, los voy a querer por siempre y cuentan conmigo en lo que sea.

Le doy gracias a mi familia por quererme mucho y estar juntos, siempre mandaré mis bendiciones al cielo a mi gran ángel Maribel siempre cuídanos tía, a mi tío Juan Manuel mis primos Arlette y Juan Manuel por ser mis compañeros de juego y recordar que somos 11 más los que vengan.

Agradezco a la vida de mis sobrinos amados Camilito y Toñito que son luces de esperanza para sus padres a los cuales agradezco por amarlos y tratar de darles lo mejor, prometo guiarlos para convertirlos en hombres de bien.

Agradezco a mis hermanos volcanos por estar ahí y enseñarme el valor de un primer amigo y cómo gente desconocida puede ser familia, gracias Rodrigo, David, Edgar, Miguel y Misael.

Agradezco a mis hermanos de la prepa los RU2, sin duda la mejor etapa de mi vida y ha sido genial pasar por esos momentos con ustedes y seguir siendo mi familia gracias Carlos, Julio, Tonantzin, Jorge y Serafín.

Por último a mis camaradas y grandes amigos de la universidad donde conocí familia y un gran equipo de trabajo muchas gracias Jonathan, Luis, Oscar, Erick, Hugo, Fermín, Abril, Dulce, Oscar Moreno, Mauricio Ontiveros y al Mtro. Arturo Ocampo.

Contenido

Índice de figuras

Índice de Tablas

Resumen

En el presente trabajo se describe la implementación de un sistema autónomo para un piloto automático realizado a través de algoritmos en una tarjeta de desarrollo, la cual cuenta con un microcontrolador para comandar un cuadricóptero mediante un procesador Cortex8.

El cuadricóptero es controlado por una rutina para la auto navegación después de hacer una clasificación de piezas de manufactura. Las rutinas están desarrolladas en la plataforma de Arduino, las cuales son ejecutadas en una interfaz desarrollada en Linux para ser visualizadas y posteriormente transmitir los comandos de navegación hacia el dron.

El reconocimiento de piezas de manufactura esta implementado en una tarjeta en la placa BeagleBoneBlack, que cuenta con un procesador basado en la arquitectura ARM Cortex-A8. A través de algoritmos de reconocimiento de imagen, se detectan diferentes tipos de piezas que son clasificadas de acuerdo a su forma.

Con estos procesos se pretende dar una alternativa de transporte de piezas en la industria y reducir el desgaste del personal humano en labores monótonas y bajar costes de producción.

Capítulo 1
Introducción

Desde hace algunos años, la aspiración del ser humano en diferentes etapas ha sido volar. El estudio, especulación y experimentación, desencadenaron en un sinfín de procesos que nos han llevado a seguir implementando nuevas tecnologías que aumentan la precisión y eficiencia de la aviación [1].

La aviación como ciencia, se remonta al diseño del cometa hacia el siglo V. Para el XVI, Leonardo da Vinci proponía diversos diseños aeronáuticos que impulsaron el estudio de esta ciencia durante cientos de años. La búsqueda de nuevo conocimiento, ha hecho posible la creación de transporte de grandes dimensiones, velocidad y durabilidad.

El dron surgió como alternativa donde un avión tripulado no podía acceder por sus dimensiones, costo y riesgo humano. Son definidos como vehículos de control remoto de manera automática o semiautomática, del tipo según el léxico oficial del ejército estadounidense, es definido como un vehículo terrestre, naval o aeronáutico [2].

Un dron puede ser controlado a distancia, por operadores humanos (Principio de Tele-comando) o de manera autónoma, mediante dispositivos robóticos (Principio de pilotaje automático). En la práctica se combinan estos dos modos de control. El termino formal para referirse a estos dispositivos es el de *UAV* (acrónimo de Unmanned Aerial Vehicle) o vehículo aéreo no tripulado [2].

1.1 Planteamiento del problema

A pesar de los grandes avances tecnológicos en la automatización durante el último siglo, aún existen procesos industriales que siguen utilizando la fuerza humana en operaciones monótonas y repetitivas para la segmentación y transporte de piezas manufacturadas en sistemas de producción en serie [3].

Esto trae consigo problemas físicos y psicológicos para los operadores, como trastornos musculo esqueléticos de la espalda y miembros superiores, depresión, angustia, insatisfacción, estrés laboral, entre otros [4].

1.2 Justificación

Enfocando el trabajo humano en el desarrollo intelectual, los riesgos a la salud disminuyen considerablemente, ya que el personal adquiere nuevas responsabilidades que los dotan de valor y seguridad.

Este proyecto pretende continuar con la búsqueda de tecnología automática que apoye al operador de estaciones de producción en sus labores cotidianas.

1.3 Propuesta de solución

Para contar con un dispositivo que no esté limitado a un área específica como lo son los brazos robóticos, se propone el uso de un *UAV* cuadricóptero que cuente con comunicación remota de medio alcance.

La autonomía en el pilotaje del dron provendrá de la combinación de una tarjeta de desarrollo la cuál transmitirá comandos de conducción a través de una interfaz gráfica en una *PC* (Personal Computer) hacia el dron y para elegir la ruta que le corresponde a una pieza, se implementarán algoritmos de procesamiento de imagen dentro de una microcomputadora con procesador *ARM* la cual transmitirá un comando *ASCII* hacia la tarjeta que contiene las rutinas de conducción automática, para que ejecute la correspondiente a la pieza de manufactura visualizada.

1.4 Objetivos

Objetivo 1.-Desarrollar una serie de rutinas de auto navegación dentro de una tarjeta de desarrollo que sea capaz de emular un control HID (Human Interface Device) mediante USB (Universal Serial Bus) y cuente con un puerto serial.

Objetivo 2.-Implementar algoritmos de reconocimiento de objetos utilizando visión artificial dentro de una tarjeta que cuente con un procesador ARM y ejecute un sistema operativo.

Objetivo 3.-Entablar comunicación entre un UAV y PC para transmitir comandos de conducción aérea de manera remota a medio alcance con un control HID

1.5 Metodología

Inicialmente se hace un estudio del arte para tener conocimiento de los trabajos previos de tesis realizados dentro de la UNAM y así poseer antecedentes que sirvan de apoyo y fundamento.

El trabajo comienza con la búsqueda de un *UAV* que sea estable para volar en exteriores e interiores, que cuente con comunicación WiFi y una interfaz de vuelo compatible con controles *HID*.

Posteriormente, se comparan las distintas tarjetas de desarrollo, tomando en cuenta la posibilidad de emular un control *HID*, software libre, módulo UART, memoria suficiente para almacenar las rutinas y sea compacto.

A continuación, se elige una microcomputadora de procesador *ARM* que soporte el sistema operativo Linux, que cuente con los periféricos de comunicación para la cámara unidad USB y módulo *UART* y disponga de bibliotecas y software libre.

Se comienza desarrollando programas predefinidos para determinar definir la trayectoria del dron y así lograr el vuelo independiente.

Finalmente se implementan bibliotecas de código abierto para el reconocimiento de piezas de manufactura y así seleccionar la rutina correcta.

Capítulo 2
Estado del arte

2.1 Vehículo aéreo no tripulado

Un *UAV* o vehículo aéreo no tripulado es un dispositivo aéreo que usa fuerzas aerodinámicas, puede volar automáticamente o ser pilotado remotamente y cuenta con algún medio de recuperación; se excluyen planeadores, misiles y proyectiles [2].

Estos fueron desarrollados por la industria militar para varias labores, ya que un vehículo aéreo de configuración *UAV* ofrece una amplia gama de aplicaciones de bajo riesgo y alta precisión. Existe gran cantidad de estos dispositivos en el mercado, desde drones de grandes dimensiones para altos vuelos y grandes distancias, y de muy pequeñas dimensiones, con medidas en centímetros para viajes cortos de mucha precisión.

Sin embargo, un vehículo aéreo no tripulado no necesariamente cuenta con vuelo autónomo, sino que la mayoría de las veces está comunicada con un operador de tierra, ya sea un piloto, controladores o cualquier otro tipo de operador relacionado con el monitoreo de la UAV, como se muestra en la Figura 2.1.

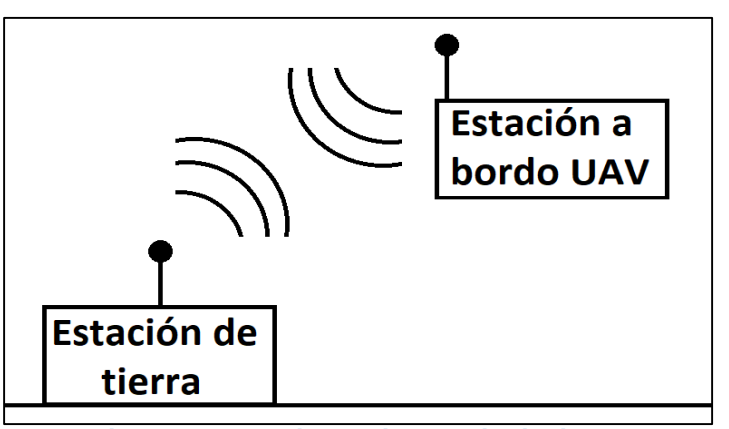

Figura 2.1 Estaciones de manejo de datos

2.1.1 Vehículos de trayectoria definida

Los *UAS* (*Unmanned Aircraft System*) son la evolución directa de los vehículos *UAV*, éstos operan con pilotaje en tierra, ya sea un humano o un sistema electrónico autónomo. Uno de estos es el de trayectoria definida, donde el control de vuelo lo lleva una estación electrónica que contiene la información necesaria para realizar una ruta específica [2].

En el canal de comunicación existen dos estaciones que manejan los datos de información del *UAV*, la estación de tierra y la estación que está a bordo de la unidad. Dependiendo de la capacidad de autonomía del *UAS*, la estación de tierra realizará más o menos funciones de forma habitual para la determinación, realización de una trayectoria o una tarea desde el inicio hasta el retorno.

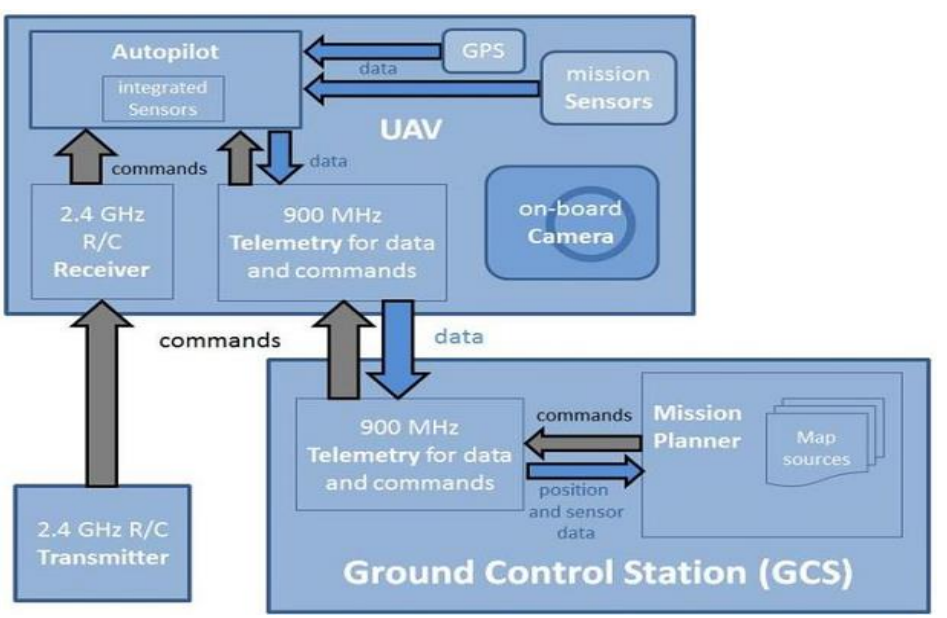

Figura 2.2 Componentes internos de un uav

Como en la industria militar, a nivel público y por parte de la iniciativa privada, se han desarrollado una gran cantidad de modelos de vehículos aéreos no tripulados para diferentes usos. El diseño depende de la configuración aerodinámica y puede dividirse en algunos *UAV* convencionales, como son [5]:

- De motor a reacción.
- De motor de propulsión a pistón.
- Dirigible.
- Cuadricóptero eléctrico.

2.1.2 Cuadricóptero eléctrico

Como se puede observar en la web, algunos vehículos son de gran tamaño, por lo que se encuentran en el mercado a un alto costo y para cuestiones académicas no son viables. Una alternativa son los cuadricópteros de motores eléctricos, este tipo de *UAV* es implementado en la realización de este proyecto.

Se emplea el AR Drone 2.0 fabricado por Parrot, el cual posee un microprocesador *ARM Cortex A8*de 32 bits para el procesamiento de las señales recibidas a través de una serie de sensores. También incluye dos cámaras que le permiten captar lo que ocurre a su alrededor. Además, la estación de control ocupa un *Smartphone* o *Tablet* mediante una aplicación desarrollada por el fabricante del *UAV* [6].

En cuanto al protocolo de comunicación, cuenta con una conexión WIFI, y un techo de altitud limitado por la propia conectividad. La integración de cámaras, sensores, tratamiento de imágenes y el SDK que distribuye gratuitamente Parrot, hacen posible el diseño de un sistema de control autónomo de trayectoria definida para el vehículo [6]. En la Figura 2.3 se presenta este vehículo.

Figura 2.3 AR Drone 2.0

2.2 Arquitectura ARM

ARM se formó en 1990 como *Advanced RISC Machines*, una empresa conjunta de Apple, Acorny VLSI. En 1987 fueron lanzados los primeros procesadores, y para1991,*ARM* introdujo la familia de procesadores *ARM6* como una empresa consolidada en el mercado, posteriormente Texas Instruments, NEC, Sharp y ST Microelectronics licenciaron el diseño de los procesadores [7].

Es una arquitectura *RISC* (Reduced Instruction Set Computer) de 32 y 64 bits. Actualmente las aplicaciones de estos procesadores incluyen teléfonos móviles, discos duros de ordenadores, asistentes digitales personales (*PDA*), tarjetas de desarrollo, entre otros [8].

El diseño basado en RISC, promueve que los procesadores *ARM* requieran una cantidad menor de transistores que los procesadores x86 CISC típicos en la mayoría de ordenadores personales. Este enfoque reduce los costos, calor y energía, lo cual es deseable para dispositivos que funcionan con baterías [8].

En cuanto a la arquitectura básica de *ARM*, puede ser representada de la siguiente forma:

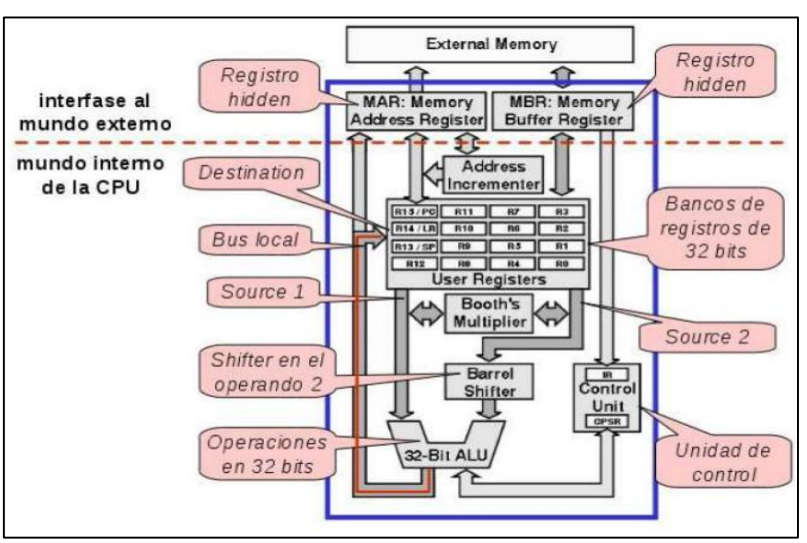

Figura 2.4 Arquitectura ARM

Donde se encuentran elementos como:

- Registros R_0 a R_{15} como apuntadores a memoria o acumuladores de 32 bits
- Unidad *ALU* (Arithmetic Logical Unit) de 32 bits
- Bloque de corrimiento paralelo a la *ALU*
- Bloque multiplicador
- Interfaz de transferencia de datos *MAR* (Memory Address Register) y *MBR* (Memory Buffer Register)
- Unidad de control

Algunas características de la arquitectura *RISC* incorporadas son:

- Arquitectura de carga (*Load-store*)donde las instrucciones de acceso a la memoria están separadas de las instrucciones de procesamiento de datos
- Formato fijo de palabra de instrucciones que simplifican la decodificación de las instrucciones
- Máquina de tres direcciones, donde un número de f bits representa al código de una operación, n bits especifica la dirección del primer operando y n bits para la dirección resultado o destino. Por ejemplo, al sumar dos números en lenguaje ensamblador da como resultado:

$$ADD\ d, S_1, S_2 \quad ; d = S_1 + S_2$$

Donde los registros son:

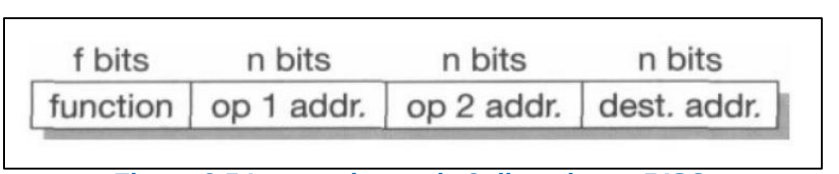

Figura 2.5 Instrucciones de 3 direcciones RISC

Además, cada dirección se especifica por registros que han sido cargados previamente con el contenido de sus direcciones de memoria correspondientes [8].

La reducción de transistores en el diseño de esta arquitectura reduce el costo de fabricación y adquisición. La incorporación y mejora de características de la arquitectura *RISC* permiten a los procesadores *ARM* adquirir equilibrio entre el alto rendimiento y el

bajo consumo de energía eléctrica manteniendo potencia óptima de ejecución [9].

2.3 Arduino

Arduino es una plataforma de prototipos electrónica de código abierto (Open-source) diseñada con hardware y software flexibles. Reconoce el entorno a través de una variedad de módulos de sensado que pueden ser incorporados a la tarjeta base. Puede interactuar con el mundo exterior mediante el control de luces, motores y diversos actuadores [26]. En la siguiente figura se muestra la amplia gama de tarjetas y extensiones de esta plataforma.

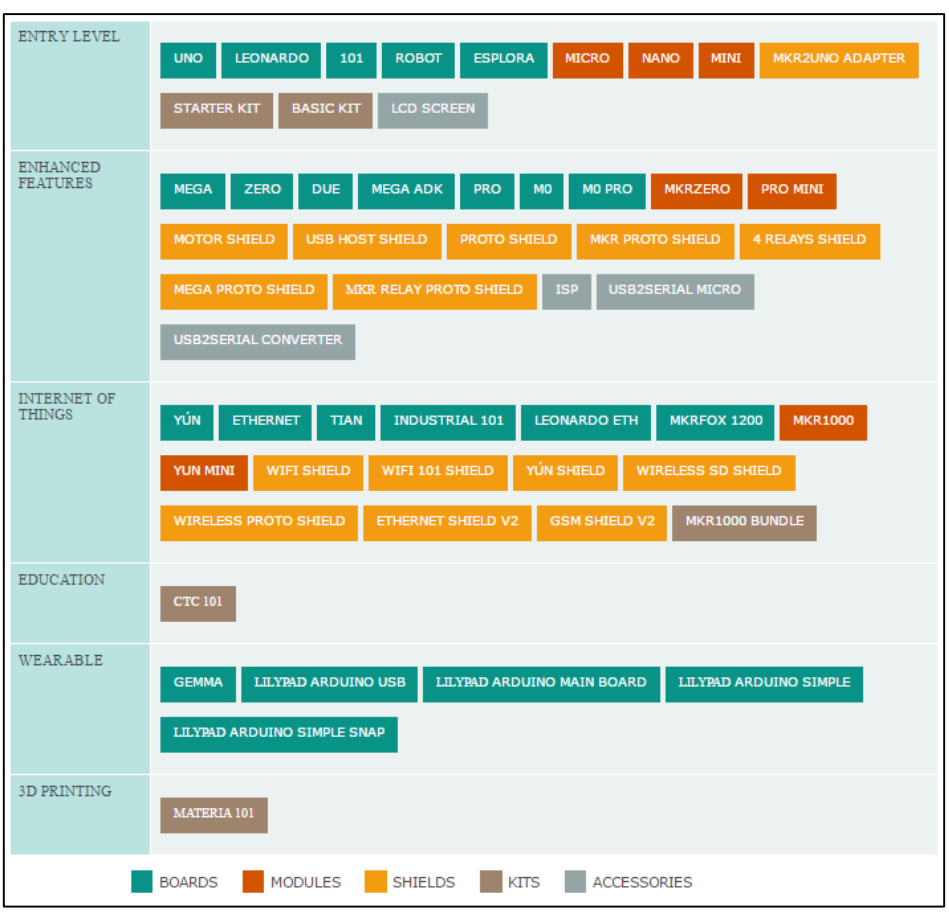

Figura 2.6 Tarjetas y extensiones Arduino

La tarjeta puede ser programada usando el *ArduinoIDE*(), el cual es distribuido por los desarrolladores. Incorpora un lenguaje

basado en *C* y *C++*, pero al incorporar un microcontrolador *Atmel*, el desarrollo de programas puede llevarse a cabo a través de *AVR C* o *Atmel Studio*. Los proyectos desarrollados pueden ser autónomos o de ejecución en un ordenador (Flash, Processing, MaxMSP, entre otros) [13].

Algunas ventajas son:

- **Bajo costo de adquisición:** La versión económica e encuentra por debajo de los $500 MXN.
- **Multiplataforma:** El software puede ser ejecutado en sistemas operativos Windows, Macintosh OSX y GNU/Linux.
- **Entorno de programación simple y clara:** El entorno de programación de Arduino es fácil de usar para principiantes, pero suficientemente flexible para usuarios avanzados. Está basado en el entorno Processing, de manera que estudiantes aprendiendo a programar en ese entorno estarán familiarizados con el aspecto y la imagen.
- **Software extensible:** El software Arduino está publicado como herramientas de código abierto. El lenguaje puede ser expandido mediante librerías *C++,* y estudiar los detalles técnicos en lenguaje AVR C.
- **Hardware extensible:** Ya que está basado en microcontroladores de *ATMEGA,* los módulos pueden ser extendidos y acoplados según la aplicación.

2.4 Procesamiento de imágenes

El procesamiento digital de imágenes es un campo de investigación abierto. El constante progreso en esta área no ha sido por sí mismo, sino en conjunto con otras áreas con las cuales está relacionada como las matemáticas, la computación, y el conocimiento cada vez mayor de ciertos órganos del cuerpo humano que intervienen en la percepción y en la manipulación de las imágenes. Aunado a esto, la inquietud del hombre por imitar y usar ciertas características del ser humano como apoyo en la solución de problemas. El avance del Procesamiento Digital de Imágenes se ve reflejado en la medicina, la astronomía, geología, microscopía, etc. Información meteorológica, transmisión y despliegue agilizado de imágenes por Internet las cuales tienen sustento gracias a estos avances.

El término "imagen monocromática" o imagen simplemente, se refiere a una función de intensidad de luz bidimensional f(x,y), donde “X” y “Y” indican las coordenadas espaciales y el valor de f en cualquier punto (x,y) es proporcional a la luminosidad (o nivel de gris) de la imagen en dicho punto. Una imagen digital es una imagen (función) f(x,y) que ha sido discretizada tanto en coordenadas espaciales como en luminosidad. Una imagen digital puede ser considerada como una matriz cuyos índices de renglón y columna identifican un punto (un lugar en el espacio bidimensional) en la imagen y el correspondiente valor de elemento de matriz identifica el nivel de gris en aquel punto. Los elementos de estos arreglos digitales son llamados elementos de imagen o pixeles [10] donde el pixel es la unidad mínima de visualización de una imagen digital.

En el tratamiento de imágenes se pueden distinguir tres etapas principales:

1. Adquisición de la imagen.

2. Procesamiento de la imagen.

3. Presentación al observador.

2.4.1 Bibliotecas de código abierto OpenCV

OpenCV es una biblioteca de visión por computadora de código libre disponible en http;//opencv.org, en 1999 Gary Bradsky trabajando con Intel corporation lanzo OpenCV con la esperanza de acelerar la visión por computadora y la inteligencia artificial para proveer una infraestructura sólida para cualquiera que esté trabajando en el campo.

La biblioteca está escrita en c y c++ y corre bajo Linux, Windows y mac sistema operativo x esto está activando un desarrollo en interfaces para python, java, matlab y otros lenguajes incluyendo la portabilidad de la biblioteca hacia android e IOS para aplicaciones móviles [11].

Capítulo 3
Hardware y Software del sistema

3.1 AR Drone 2.0

En este apartado se describe el UAV implementado en este proyecto y se profundiza en sus características, selección y ventajas que ofrecen para el sistema propuesto.

Se usa el AR Drone 2.0 Elite Edition de la marca francesa Parrot. En la siguiente figura se muestran las secciones esenciales que lo componen [6].

Figura 3.1 Secciones del AR Drone 2.0

Las secciones del hardware se dividen en: captura de video, electrónica, estructura y motores. Estas mismas serán descritas en los siguientes subtemas.

3.1.1 Captura de video del AR Drone

El AR Drone 2.0 Elite cuenta con 2 cámaras de video instaladas en la parte frontal e inferior, las cuales están posicionadas para

obtener un alcance mayor sobre la visión de la periferia de la región donde se encuentre.

La cámara frontal del cuadricóptero es HD 720P (1280*720 pixeles) de 30 cuadros por segundo (fps), ángulo de 92° para visión panorámica, perfil básico de codificación H264, difusión en tiempo real de baja latencia y almacenamiento de vídeo sobre la marcha.

La segunda cámara localizada en la base del cuadricóptero es una QVGA vertical de (640*480 pixeles) de 60 fps, tiene el propósito de aumentar el margen de visión y ayudar en la secuencia de aterrizaje [5].

3.1.2 Electrónica del AR Drone

Según los datos técnicos especificados por el fabricante, incorpora un microprocesador *ARM* Córtex A8 de 1 GHz a 32 bits con sistema operativo Linux 2.6.32, memoria RAM DDR2 de 1 GB a 200 MHz, conexión USB 2.0 de alta velocidad para grabar video o fotos en una memoria extraíble y un conector WIFI b/g/n.

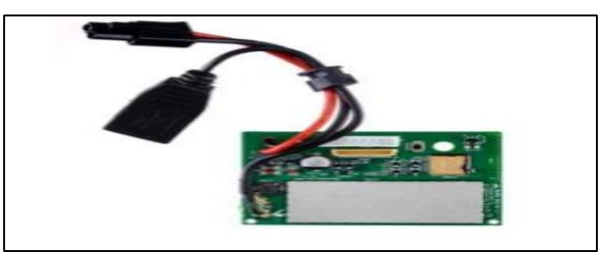

Figura 3.2 Microprocesador del AR Drone 2.0

Además, cuenta con sensores para la estabilidad de vuelo, como son:

- **Acelerómetro digital:** De 3 ejes con precisión de ±50mg para monitorizar los movimientos de posición.
- **Sensores ultrasónicos y presión:** Para el control de la altitud
- **Magnetómetro:** De 3 ejes con 6º de precisión para la orientación
- **Giroscopio:** De 3 ejes a 2000°/s para el cabeceo, giro y viraje

- **Cámara QVGA vertical:** Para el aterrizaje, relaciona la velocidad que existe desde el cuadricóptero con respecto del suelo [6].
- Estos sensores proporcionan la información necesaria para la navegación y telemetría. La combinación de los datos recopilados también brinda los ángulos de Euler, que son utilizados para la estabilización que realiza el movimiento alrededor del eje X
- **Yaw (Guiñada):** Es el ángulo ψ que realiza el movimiento respecto al eje Z [5].

A continuación, se presentan estos movimientos y su representación en función de los ángulos de Euler:

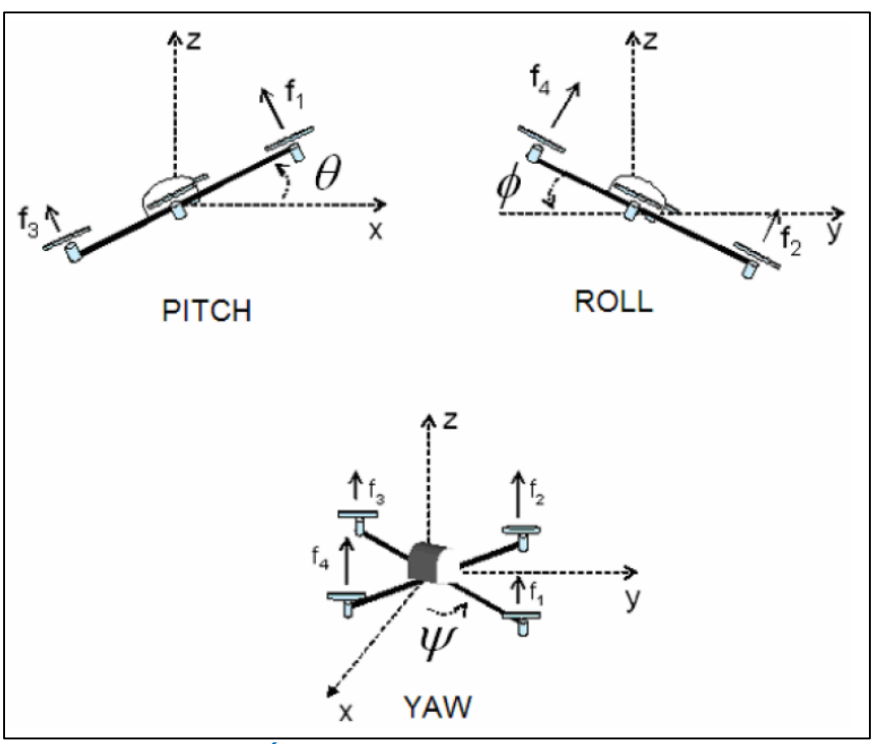

Figura 3.3 Ángulos de Euler en un cuadricóptero

El modelo dinámico para el movimiento, consta de dos sistemas de ecuaciones representados de manera matricial:

$$\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = -g * \begin{bmatrix}0\\0\\1\end{bmatrix} + \frac{T}{m} * \begin{bmatrix}\cos(\Psi)\sin(\theta)\cos(\phi) + \sin(\Psi)\sin(\phi)\\ \sin(\Psi)\sin(\theta)\cos(\phi) - \cos(\Psi)\sin(\phi)\\ \cos(\theta)\cos(\phi)\end{bmatrix} - \frac{1}{m}\begin{bmatrix}A_x & 0 & 0\\ 0 & A_y & 0\\ 0 & 0 & A_z\end{bmatrix}$$

El primer sistema matricial proporciona el avance lineal en tres dimensiones del sistema inercial como resultado de las aceleraciones en sus tres componentes $\ddot{x}, \ddot{y}, \ddot{z}$ aerodinámicas.

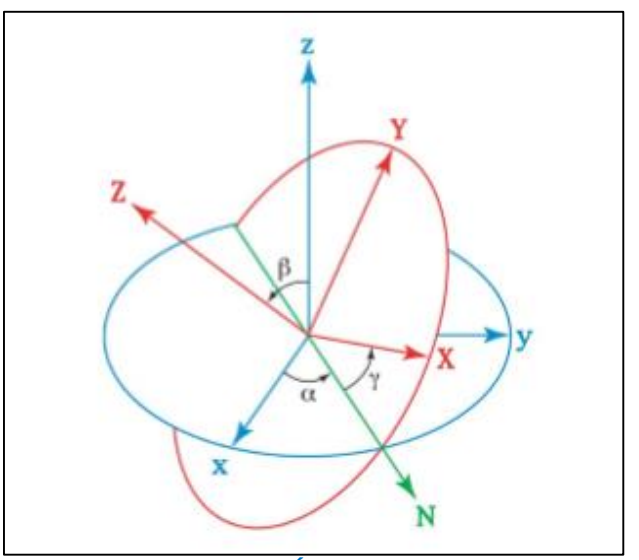

Figura 3.4 Ángulos de Euler

Roll (Alabeo): Es el ángulo ϕ

Además

- g la aceleración gravitatoria,
- T el empuje de los cuatro rotores,
- m la masa del dron,
- A_x, A_y, A_z son los coeficientes de arrastre generado por la fricción con el aire

Por otra parte, los movimientos angulares del cuadricóptero son representados por la forma matricial:

$$\begin{bmatrix}\ddot{\phi}\\ \ddot{\theta}\\ \ddot{\psi}\end{bmatrix} = J^{-1}\left(\begin{bmatrix}\tau_\phi\\ \tau_\theta\\ \tau_\psi\end{bmatrix} - C[n, \dot{n}] * \begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix}\right)$$

Donde:

- J^{-1}esto representa a la matriz inversa Jacobina, que convierte los momentos inerciales en no inerciales.
- $[\tau_\phi, \tau_\theta, \tau_\psi]$ los pares inducidos por cada dirección angular
- $C[n, \dot{n}]$ es la matriz que representa los efectos de giro dinámicos y centrípetos en el desplazamiento en función del termino Coriolis, el cual es la representación del movimiento de un objeto sobre el radio de un disco en función de su aceleración y distancia del centro al perímetro [28, 29,30].

3.1.3 Motores del AR Drone.

El sistema de propulsión del cuadricóptero consta de 4 motores de rotor interno conocidos como “in runner”, una de sus características es el funcionamiento sin escobillas y consume 14.5 W cuando está en movimiento y hasta 28.5W cuando se mantiene suspendido en el aíre, su rodamiento está compuesto por un sistema de bolas en miniatura [6].

Figura 3.5 Motor del AR Dron 2.0

Cuenta con un acoplamiento de engranajes de Nylatron de bajo ruido para reducir la dimensión de la hélice, un eje de acero templado para sostener las hélices y cojinetes de bronce auto lubricado para reducir consumo de potencia. Esto proporciona alta fuerza de propulsión y mayor maniobrabilidad.

Figura 3.6 Engranajes, soporte y hélice del Ar Drone 2.0

El control que enlaza los sensores y actuadores es un microcontrolador AVR de 8*MIPS* (Millones de Instrucciones Por Segundo) por cada motor, además son resistentes al agua [6]. Los microcontroladores implementados se muestran a continuación:

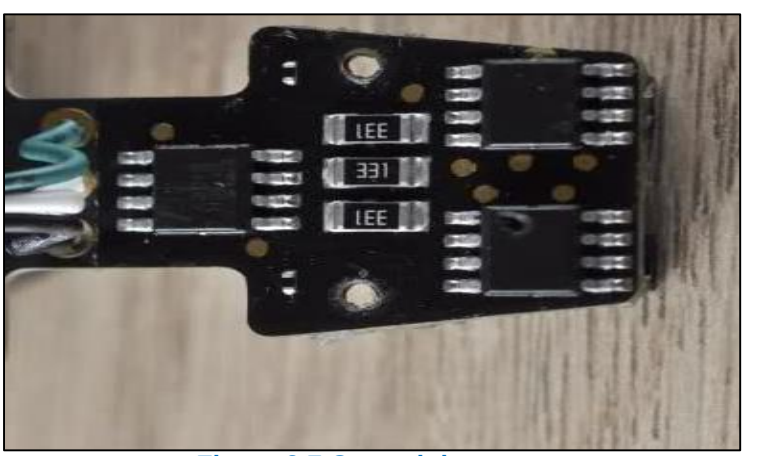

Figura 3.7 Control de motor

3.1.4 Estructura del AR Drone

Cuenta con un diseño de vanguardia enfocado en la duración larga de tiempo en uso, esto se debe a que el AR Drone 2.0 Elite es elaborado con poliestireno. Los anillos de la carcasa para interiores son diseñados para la protección de las hélices y motores contra el contacto de algún objeto o choque accidental. También aísla el centro de inercia de vibraciones de los motores. Las piezas del cuadricóptero son hechas de una mezcla de nylon en 70% y fibra de vidrio en 30% [6].

El cuerpo del dron se presenta a continuación:

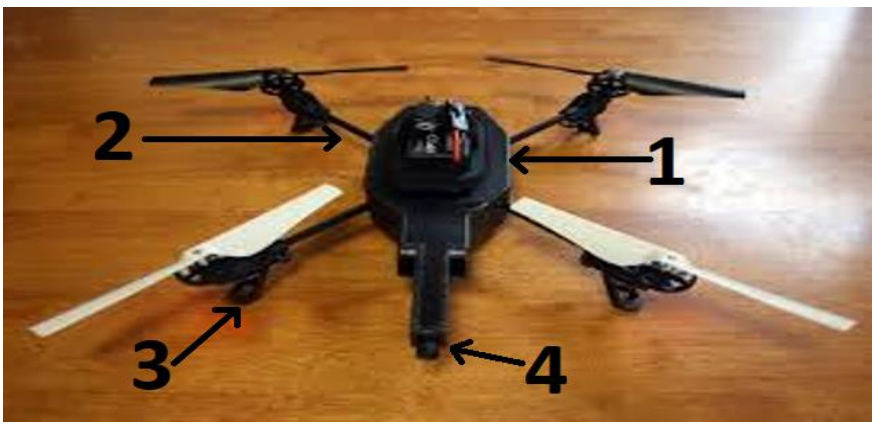

Figura 3.8 Estructura del AR Dron

Donde según la rotulación se localiza:

Cuerpo principal donde se localiza la batería y procesador cuatro tubos de fibra de carbono que son la base para los motores eléctricos patas para el soporte del cuadricóptero y soporte de cámara frontal.

El uso de la carcasa para exteriores deja un peso de 380 g y a la de interiores 420 g. El casco de protección está fabricado con PPE que es polipropeno expandido que es inyectado en un molde metálico [6].

Figura 3.9 Carcasas del AR Drone 2

3.1.5 Alimentación del Drone.

El dron dispone de una batería recargable compuesta por tres celdas de Litio-Polímero con capacidad de 1000mAh a 11.1V y de descarga de 10Coulomb.Cumple con la normativa de seguridad UL2054 que brinda seguridad para vuelos de mayor longitud y duración [6].

Figura 3.10 Cargador y Batería del AR Dron 2

3.2 SDK AR Drone 2.0.1

El kit de desarrollo de software es utilizado para el control y creación de aplicaciones los *UAV* AR Drone, fue el primer *SDK* lanzado para drones comerciales convencionales. Al adquirir un dispositivo, este incluye documentación completa y código de ejemplo para iOS, Android, Linux y Windows. La última versión estable es SDK2.0.1, que proporciona las herramientas necesarias para el host e incorpora algunas características, como son:

- Controles intuitivos y de inclinación de vuelo.
- Transmisión de video en vivo.
- Grabación de vídeo y fotografía.
- Actualización de ángulos de Euler.
- Detección de etiquetas embebidas para juegos de realidad aumentada.

Además, permite a los desarrolladores de terceros crear y distribuir nuevos juegos, basado en el producto AR Drone 2.0 para Wifi, sensores de movimiento de dispositivos móviles como el iPhone de Apple, IPad, iPod, computadoras personales o dispositivos Android.

Este SDK incluye:

- **ARDroneLIB:** Proporciona las API necesarias para comunicarse fácilmente y configurar un AR Drone 2.0
- **ARDroneTool:** Brinda un dron completamente funcional al cliente donde los desarrolladores solo tienen que insertar su código específico de aplicación personalizada
- **Control Engine:** Permite el control intuitivo desarrollado por Parrot para controlar remotamente el dron desde un dispositivo iOS
- **Ejemplos de código:** Muestran cómo controlar el dron desde una computadora personal con Linux.
- **ARFreeFlight 2.0:** Es el código fuente para aplicaciones en iOS y Android

3.3 Tarjeta de desarrollo BeagleBoneBlack

Es la última adición a la familia BeagleBone y al igual que sus predecesores, está diseñado para hacer frente a la comunidad de código abierto interesados en un procesador de bajo costo basado en un *ARM* Cortex-A8. Ha sido equipado con un conjunto mínimo de funciones para permitir al usuario experimentar la potencia del procesador sin pretender ser una plataforma de desarrollo completo.

Esta sección cubre las especificaciones y características de la tarjeta y proporciona una descripción de alto nivel de los principales componentes e interfaces que componen la tarjeta [7].

3.3.1 Especificaciones de la tarjeta BBB

Al ser una tarjeta de desarrollo para sistemas embebidos de código abierto, incluye procesamiento y periféricos necesarios para diversas tareas, como lo son:

- Cuatro puertos *UART* (Universal Asynchronous Receiver Transmitter)
- Dos módulos *SPI* (Serial Peripheral Interface)
- Ocho módulos *PWM* (Pulse Width Modulation)
- Dos periféricos *CAN* (Controller Area Network)
- Un convertidor *ADC* (Analog toDigital Converter) de 12 bits

Estos están distribuidos entre los 92 pines disponibles en dos tiras de pines.

La BeagleBoneBlack no intenta remplazar a una computadora de escritorio, sino que busca ser implementada exclusivamente como computadora embebida para proyectos específicos y brindar flexibilidad para moverse en conjunto con el proyecto desarrollado cuando sea necesario cambiar el área de trabajo [12].

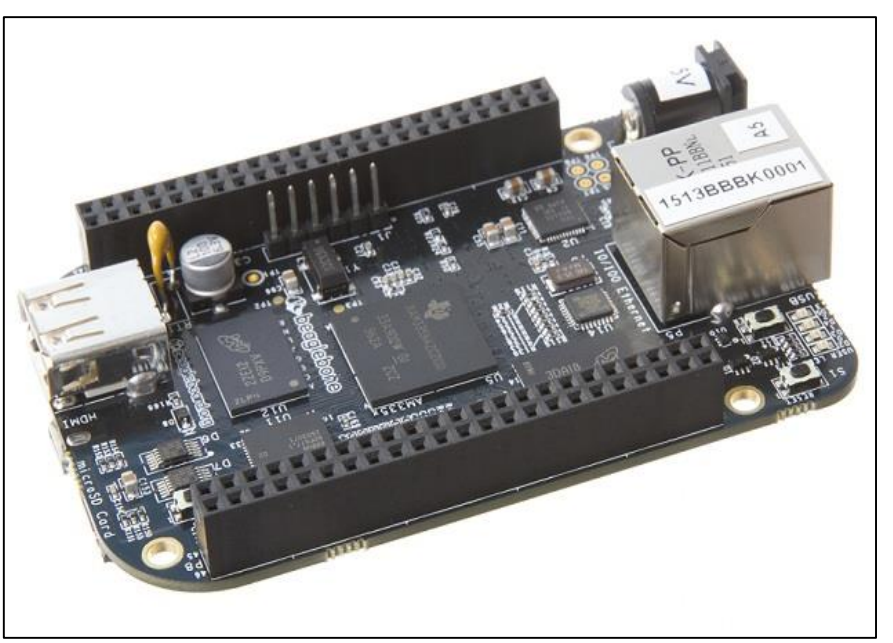

Figura 3.11 Tarjeta BeagleBoneBlack

A continuación, se presentan algunas características técnicas de la tarjeta BeagleBoneBlack:

Processor	Sitara AM3359AZCZ100 1GHz, 2000 MIPS	
Graphics Engine	SGX530 3D, 20M Polygons/S	
SDRAM Memory	512MB DDR3L 606MHZ	
Onboard Flash	2GB, 8bit Embedded MMC	
Power Source	miniUSB USB or DC Jack	5VDC External Via Expansion Header
PCB	3.4" x 2.1"	6 layers
Indicators	1-Power, 2-Ethernet, 4-User Controllable LEDs	
HS USB 2.0 Client Port	Access to USB0, Client mode via miniUSB	
HS USB 2.0 Host Port	Access to USB1, Type A Socket, 500mA LS/FS/HS	
Serial Port	UART0 access via 6 pin 3.3V TTL Header. Header is populated	
Ethernet	10/100, RJ45	
SD/MMC Connector	microSD , 3.3V	
User Input	Reset Button Boot Button Power Button	
Video Out	16b HDMI, 1280x1024 (MAX) 1024x768,1280x720,1440x900 w/EDID Support	
Audio	Via HDMI Interface, Stereo	
Expansion Connectors	Power 5V, 3.3V , VDD_ADC(1.8V) 3.3V I/O on all signals McASP0, SPI1, I2C, GPIO(65), LCD, GPMC, MMC1, MMC2, 7 4 Timers, 3 Serial Ports, CAN0 EHRPWM(0,2),XDMA Interrupt, Power button, Expansion Board ID	

Figura 3.12 Características técnicas

3.3.2 Procesador ARM.

Es un *SITARAAM335x* de 1GHz y 2000 *MIPS* basado en la arquitectura *ARM Cortex-A8*. Es fabricado por Texas Instruments como microprocesador de alto rendimiento incluyendo combinaciones exclusivas de periféricos y aceleradores para reducir costos de los sistemas y expandir las opciones de conectividad.

Con este dispositivo se pueden realizar tareas de alto desempeño en aplicaciones gráficas multimedia, procesamiento de señales, control de robots, entre otras. Básicamente es un microcontrolador de 32 bits capaz de ejecutar un sistema operativo [9].

La arquitectura del procesador puede ser representada con la siguiente figura:

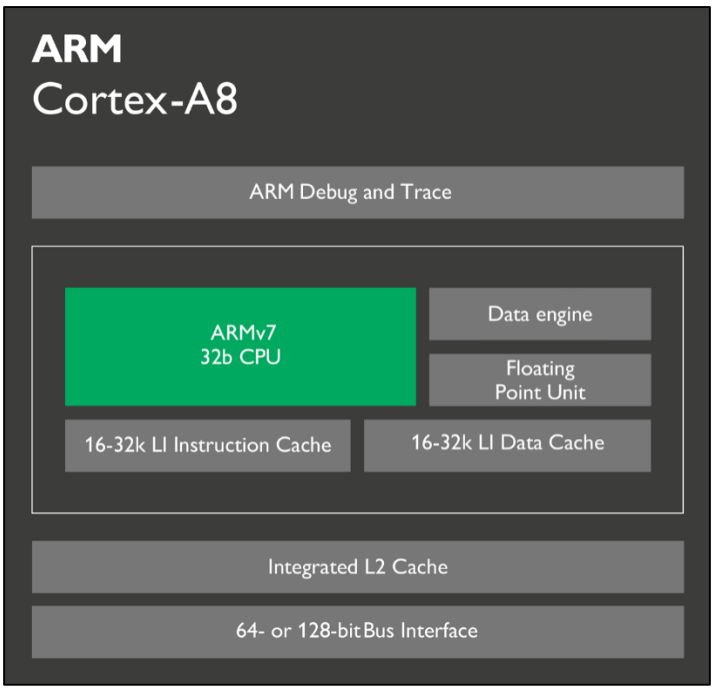

Figura 3.13 Arquitectura del procesador cortex-A8

3.4 Arduino Leonardo

Es una placa electrónica que implementa el microcontrolador*ATmega32u4*. Difiere de otras placas Arduino al incorporar comunicación USB en el microcontrolador principal, por lo que se elimina la necesidad de un procesador secundario [13].

El entorno de desarrollo de Arduino Leonardo se divide en tres bloques esenciales:

- Microcontrolador
- Tarjeta
- Programación

Los cuáles serán explicados en los siguientes subtemas.

3.4.1 Microcontrolador AVR

Es un *AVR* de 8 bits basado en la arquitectura *RISC*. Cuenta con 135 instrucciones de ejecución simple por ciclo de reloj, 32x8 registros de propósito general. Alcanza las 16 *MIPS* con un oscilador externo de 16MHz.

Respecto a las memorias, cuenta con una *Flash* para almacenar programas de 32KB, *SRAM* () de 2.5KB, *EEPROM* (Electrical Ereasable PROM) de 1KB. Retención de datos por 100 años a temperatura de 25°C

Incorpora un módulo *USB2.0 HID* (Human Interface Device) con interrupción de transferencia entre 1.5Mb y 12Mb y bus independiente de comunicación. El microcontrolador puede ser reseteado a través de este módulo.

En cuanto a periféricos, cuenta con cuatro canales *PWM* de resolución de 8 bits, cuatro de 2 a 16 bits, seis de alta velocidad de 2 a 11 bits. Además, incorpora doce canales *ADC* de 10 bits, USART, interfaz *SPI*, WatchDog y un sensor de temperatura dentro del chip [14].

Cabe mencionar que solo está disponible en versión de montaje superficial de 44 pines.

3.4.2 Especificaciones de la tarjeta de desarrollo Arduino Leonardo.

El Arduino Leonardo puede ser alimentado a través de la conexión USB o con suministro de energía externo. La alimentación externa puede provenir desde un adaptador AC/DC o una batería. La placa puede operar con un suministro externo de 6 a 20 voltios. Si es suministrada con menos de 7 V, sin embargo, el pin de 5 V puede suministrar menos de cinco voltios y la placa podría ser inestable. Si usa más de 12 V, el regulador de tensión puede sobrecalentarse y dañar la placa. El rango recomendado es de 7 a 12 voltios. Los pines de alimentación son los siguientes:

- **V_{IN}:** La entrada de tensión a la placa Arduino cuando está usando una fuente de alimentación. Puedes suministrar tensión a través de este pin, o, si suministra tensión a través del conector de alimentación
- **5V:** El suministro regulado de energía usado para alimentar al microcontrolador y otros componentes de la placa.
- **3.3V:** Un suministro de 3.3 V generado por el chip FTDI de la placa. La corriente máxima es de 50 mAmp.
- **GND:** Pines de Tierra.
- **~X:** Pines conectaos a los módulos *PWM* del microcontrolador.
- **1-13:** Pines de entrada y salida digital de 0-5V.
- **Ax:** Pines conectados a los módulos *ADC* del microcontrolador.
- **AREF:** Voltaje de referencia para las entradas analógicas.
- **RESET:** Resetea el microcontrolador al recibir un pulso bajo.
- **SDA, SCL:** Comunicación TWI usando la biblioteca Wire.
- **Rx, Tx:** Puerto USART [13].

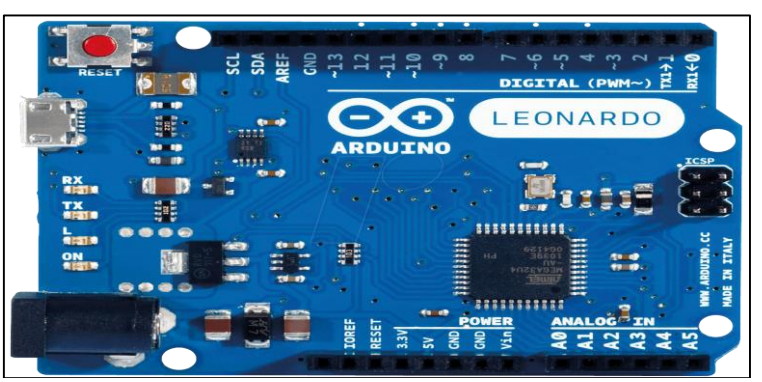

Figura 3.14 Tarjeta de desarrollo Arduino Leonardo

A continuación, se presenta una tabla que muestra algunos datos técnicos referentes a la tarjeta:

Características	***Descripción***
Microcontrolador	ATmega32u4

Voltaje de operación	5 V
Tensión de entrada (recomendada)	7 - 12 V
Tensión de entrada (límite)	6 - 20 V
Pines digitales de E/S	20 (7 de estos proveen salidas PWM)
Pines de entrada analógicos	6
Corriente DC por pin E/S	40 mA
Corriente DC para pin 3.3 V	50 mA
Memoria Flash	16/32 KB
SRAM	1.25/2.5KB
EEPROM	512Bytes/1KB
Frecuencia de reloj	16 MHz

Figura 3.15 Datos técnicos de Arduino Leonardo

3.4.3 Programación del Arduino

Como se mencionó en el capítulo anterior, Arduino Leonardo puede ser programado con el software Arduino, ya que el microcontrolador *ATmega32u4*tiene pregrabado un bootloader que permite el grabado in-circuit. Si se desea grabar con otro entorno de programación, hay que hacer las modificaciones dentro del código para no estropear el bootloader y la tarjeta ejecute el programa sin algún inconveniente. La comunica de la tarjeta y el software original usa el protocolo STK500 [13].

3.5 Sensor CNY70

Es un sensor de infrarrojos de corto alcance basado en un emisor y receptor de luz, ambos apuntando en la misma dirección, y cuyo funcionamiento se basa en la capacidad de reflexión del

objeto reflectado en el receptor. El *CNY70* tiene cuatro pines de conexión, dos de ellos se corresponden con el ánodo y cátodo del emisor, y las otras dos se corresponden con el colector y el emisor del receptor. Los valores de las resistencias son típicamente 10K ohmios para el receptor y 220 ohmios para el emisor [15]. En la Figura 3.17 se presenta este sensor.

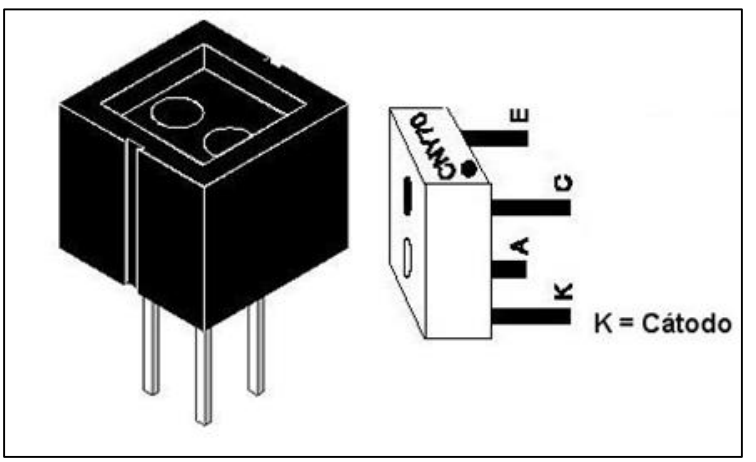

Figura 3.16 CNY70

El *CNY70* devuelve por la pata de salida correspondiente, según el montaje, un voltaje relacionado con la cantidad de rayo reflectado por el objeto. Para el montaje A, se leerá del emisor un '1' cuando se refleje luz y un '0' cuando no se refleje, para el montaje B los valores se leen del colector, y son los contrarios al montaje A. Si conectamos la salida a una entrada digital del microcontrolador, entonces se obtiene un '1' o un '0' en función del nivel al que el microcontrolador establece la distinción entre ambos niveles lógicos [16]. Ambos montajes se muestran a continuación:

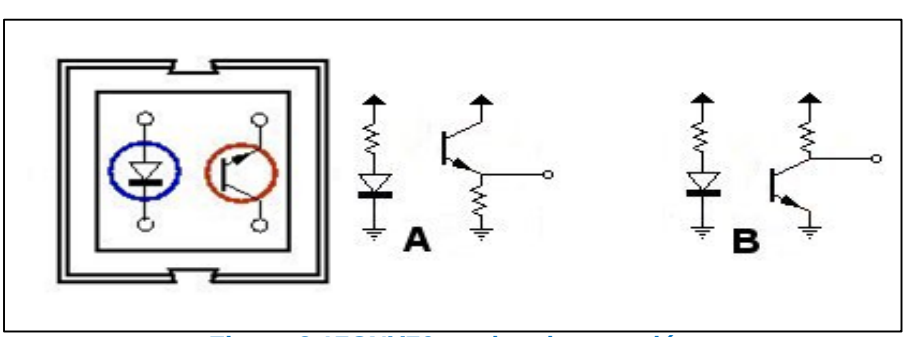

Figura 3.17CNY70 modos de conexión

Los niveles se pueden controlar introduciendo un buffer Trigger-Schmitt (como el 74HC14 que es un inversor) entre la salida del CNY70 y la entrada del microcontrolador. Este sistema es el que se emplea para distinguir entre blanco y negro en la conocida aplicación del robot seguidor de línea. Otra posibilidad es conectar la salida a una entrada analógica, de este modo, mediante un conversor A/D para obtener distintos valores para detección dinámica de blanco y negro cuando se presentan alteraciones en la iluminación [16] a continuación muestro el circuito eléctrico con su ecuación de voltaje de salida y su grafica de corriente de colector contra distancia de alcance.

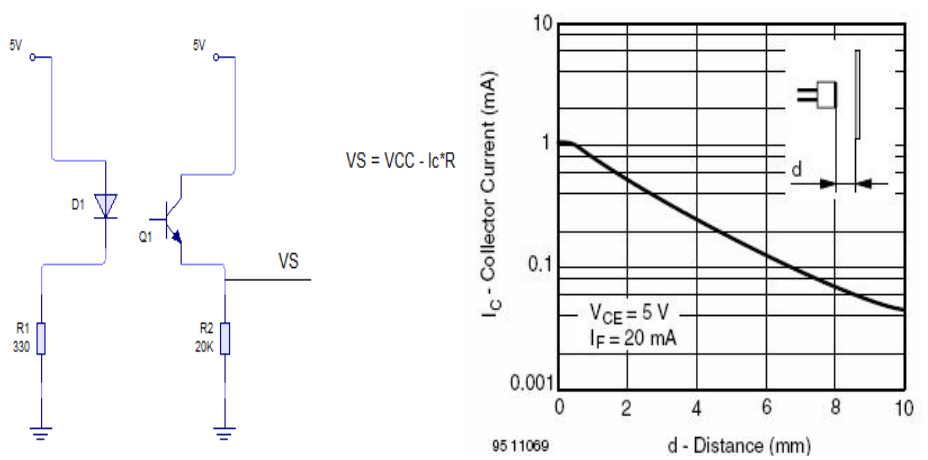

Figura 3.18 Ecuación de voltaje y grafica de Ic contra distancia

3.6 Sistema operativo Ubuntu

Se trata de un sistema operativo distribuido por Linux y está basado en Debian. Contiene una amplia gama de aplicaciones preinstaladas y muchas más disponibles. Como sistema operativo es sencillo de usar. El sistema operativo se comunica con el Hardware y las aplicaciones que utiliza para realizar su trabajo, el sistema operativo controla todos los aspectos de una computadora.

Ubuntu es parte de la familia más grande de distribuciones de Linux con kernel (Núcleo de sistema operativo que realiza funciones básicas, como control de memoria y gestión de procesos) abierto y libre, fuertemente basado en conceptos esbozados para UNIX, el antepasado de Linux [17].

Debido a que Linux es sólo un kernel, necesita otros programas para ejecutarse como un sistema operativo completo. Diferentes distribuciones de Linux, necesitan distintos paquetes de software. Ubuntu es una de esas distribuciones.

3.7 Distribución Debian

Fue fundado en 1993 por Ian Murdock, es conocido por su adhesión al software libre y de código abierto, que es incorporado en el Contrato Social de Debian y las Directrices de Software Libre de Debian.

El Proyecto Debian es una asociación de personas que han hecho causa común para crear un sistema operativo completo y libre. Los sistemas Debian actualmente usan el núcleo de Linux o FreeBSD, sin embargo, actualmente están trabajando para ofrecer Debian con otros núcleos. Debian incorpora más de 43,000 paquetes que es software pre compilado y comprimido para instalar en una PC de manera gratuita [18].

3.8 Entorno LXDE

LXDE (Lightweight X11 Desktop Environment) es un entorno de escritorio ligero libre para Unix y *BSD*(). Este proyecto apunta hacia la ligereza y rapidez para la gestión de recursos y energía

A diferencia de otros ambientes de escritorio, los componentes no se integran firmemente, sino que son independientes y cada uno de ellos puede ser utilizado independientemente con muy pocas dependencias. Esto es logrado al incorporar *OpenBox* como gestor de ventanas predeterminado.

Para este proyecto se incorporará en la BeagleBoneBlack el sistema operativo Debian 7.11 2015-06-15 4GB SD LXDE que es cargado desde una tarjeta microSD para ser instalado. Además, se instala un escritorio ligero para contar con una interfaz gráfica. Esta última ya viene pregrabada en una memoria al adquirir la tarjeta.

3.9 Bibliotecas OpenCV

Open Source Computer Visión fue desarrollada por Intel en 1999 como una herramienta de software libre para el uso académico y

comercial para la visión computarizada y el aprendizaje de máquinas. Cuenta con interfaces C ++, C, Python y Java; soporta Windows, Linux, Mac OS, iOS y Android. Fue diseñado para la eficiencia computacional y aplicaciones en tiempo real [20].

Cada biblioteca aprovecha el procesamiento MultiCore, al habilitar OpenCL aprovecha la aceleración de hardware de la plataforma de computación heterogénea subyacente [11].

Para el proyecto se incorporan las siguientes bibliotecas:

- **NumPy:** Proporciona funcionalidad de cálculo numérico incluyendo matrices eficientes bajo código en Python.
- **SciPy:** Esta es una biblioteca de computación científica que está estrechamente relacionada con NumPy. Es útil para manipular los datos en Imágenes en Open CV.
- **OpenNI:** Añade soporte para cámaras de profundidad
- **Sensor Kinect:** Se trata de un complemento OpenNI y añade soporte para la cámara de profundidad

3.10 Lenguaje de programación Python

Es un lenguaje de programación robusto, cuenta con estructuras de datos eficientes, de alto nivel y un enfoque efectivo a la programación orientada a objetos.

La elegante sintaxis de Python y su tipa do dinámico, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas. El intérprete de Python y la extensa biblioteca estándar están a libre disposición en forma binaria y de código fuente para las principales plataformas [19]

El sitio oficial provee distribuciones y enlaces de muchos módulos libres de Python, programas, herramientas, y documentación adicional. El intérprete de Python puede extenderse fácilmente con nuevas funcionalidades y tipos de datos implementados en C o C++ (u otros lenguajes accesibles desde C). Python también puede usarse como un lenguaje de extensiones para aplicaciones personalizables [20].

3.11 Arquitectura del software implementado

La incorporación de las herramientas de software mencionadas anteriormente dentro de la BeagleBoneBlack e instaladas en una PC, provee el entorno necesario para llevar a cabo la segmentación de las piezas de manufactura y la transmisión de las instrucciones necesarias al dron.

En la Figura 3.19 se muestra un diagrama a bloques donde se representa la arquitectura del software implementado en la BeagleBoneBlack y su función dentro del proyecto.

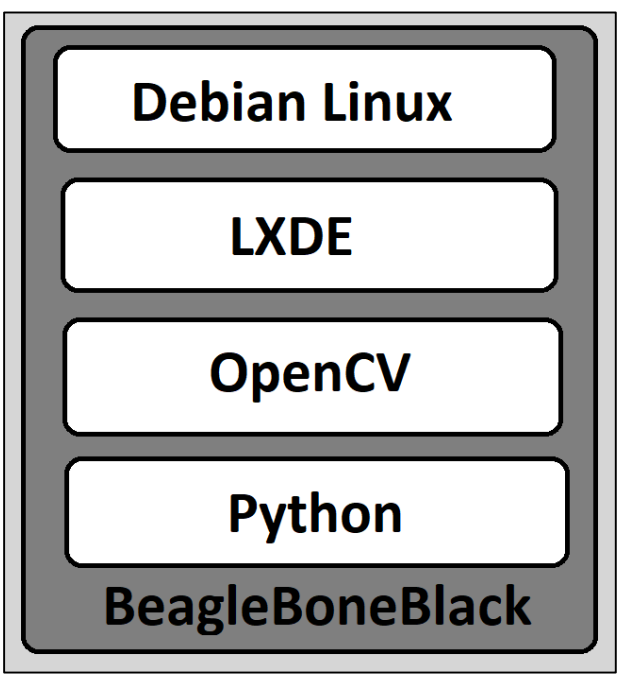

Figura 3.19 Arquitectura de software en BBB

Donde:

- **Debian:** Es el núcleo sistema operativo y gestiona las tareas de las demás herramientas
- **LXDE:** Es el escritorio ligero que brinda la interfaz gráfica del OS.

- **OpenCV:** Provee las herramientas para el procesamiento de imágenes y reconocimiento de objetos
- **Python:** Despliega los algoritmos a OpenCV y trabajando en conjunto, procese y obtenga la información necesaria para transmitir al resto del sistema autónomo de este proyecto

Respecto a la PC, se representa de la siguiente manera:

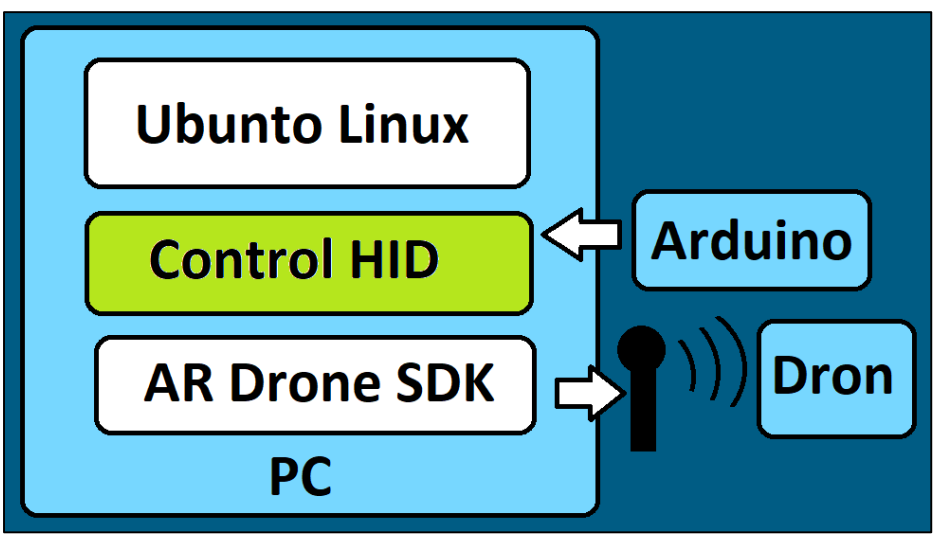

Figura 3.20 Arquitectura de software en PC

Donde:

- **Ubuntu:** Es el sistema operativo en el que se trabaja
- **Control *HID*:** Es la tarjeta Arduino es reconocida como *HID* y transmite la información de movimiento que realizará el dron
- **AR Drone SDK:** Es el entorno de comunicación entre la PC y el dron, se encarga de transmitir las instrucciones recibidas por Arduino a través del módulo Wifi del ordenador

Capítulo 4
Diseño del sistema autónomo

4.1 Descripción del sistema

El sistema está compuesto por una tarjeta de desarrollo BeagleBoneBlack, un Arduino Leonardo, la interface SDK instalada en una computadora personal, el sensor CNY70 y un cuadricóptero eléctrico. En la Figura 4.1 se muestra el diagrama esquemático del sistema y a lo largo de este tema se detallará el funcionamiento del mismo.

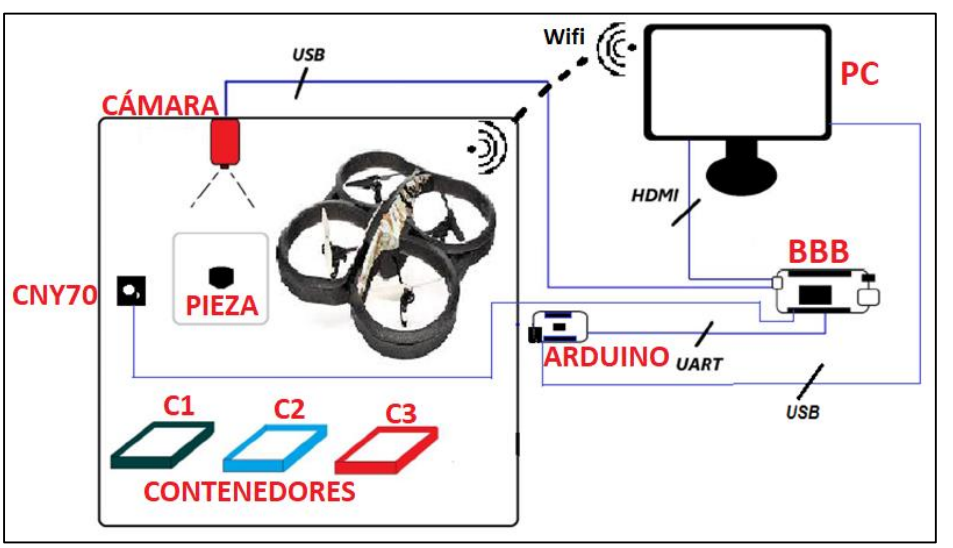

Figura 4.1Diagrama esquemático del sistema

El sensor CNY70 monitorea el área de trabajo en espera de una pieza de manufactura, una vez detectada la pieza, el sensor envía una señal que activa una de las *GPIO* (General Purpose Input/Output) del BeagleBoneBlack; que tiene instalado el sistema operativo Debian donde están instaladas las paqueterías de OpenCV usadas para el procesamiento de la imagen captada por la cámara. Esta última es conectada por medio de un cable *USB* y al adquirir la imagen, es almacenada en la memoria de la BeagleBoneBlack.

El procesamiento de la imagen capturada consta de:

1. Convertirla en escala de grises para obtener un mapa de bits
2. Binarizarla para reducir la cantidad de información y agilizar el procesamiento
3. Obtener el contorno, centroide y coordenadas en pixeles
4. Desarrollar la función frontera y se caracteriza al objeto a través de un vector que contiene la información del área y perímetro
5. Representar gráficamente la captura de cada elemento, visualizar cada etapa del procesamiento y monitorear cualquier error que se presente.
6. Cada pieza es clasificada según la información obtenida por la función de compacidad, está brinda información que, si es encontrada dentro de un rango de valores, determina la pieza de manufactura que se está visualizando
7. Con la información obtenida**,** la BeagleBoneBlack transmite por el puerto serial un número decimal (1,2 o 3) en formato *ASCII* hacia la tarjeta Arduino Leonardo, según haya sido la figura reconocida y procesada.
8. La tarjeta Arduino incluye la librería *joystick.h,* la cual permite que Arduino emule un control HID conectado vía USB con una computadora personal, la cual cuenta con el sistema operativo Ubuntu Linux 12.04 LTS, en este se encuentra instalado el entorno AR Drone SDK, que es el software usado para construir la interface para pilotar el AR Drone 2.0 mediante conectividad wifi.

4.2 Procesamiento de la imagen

La distribución precargada con la que cuenta el procesador Sitara Cortex A8 es *Debian*, esta consume aproximadamente 600MB de los 4GB disponibles en la memoria de almacenamiento de la tarjeta BBB, dejando suficiente espacio para las bibliotecas.

El lenguaje *Python* es utilizado para definir el funcionamiento del procesador. Se optó por este lenguaje de programación de alto nivel ya que es software libre, robusto y de fácil interpretación. Permite escribir programas compactos, ya que no es necesario declarar variables ni argumentos, debido a que la agrupación de instrucciones se realiza mediante sangrías.

Para el reconocimiento de objetos se utiliza OpenCV, que es una biblioteca de código abierto empleada en aplicaciones de visión artificial ya que incluye gran número de algoritmos de visión por computadora, además es multiplataforma por lo que posibilita la instalación y funcionamiento en cualquier sistema operativo [20].

En cuanto al entrenamiento y clasificación de los objetos caracterizados, se utiliza una biblioteca *LIBSVM* interpretada en *Python*, la cual cuenta con algoritmos para el diseño y entrenamiento de clasificadores SVM.

4.2.1 Adquisición de la imagen

El proceso comienza con la detección de una pieza de manufactura colocada en el área de trabajo, a través del sensor infrarrojo CNY70. Este se encuentra conectado a una GPIO de la BeagleBoneBlack y este a su vez vía USB, tiene conectada una cámara de video Logitech C170

Cuando el sensor indica que un objeto se encuentra en el área de trabajo, envía una señal a la tarjeta para iniciarla captura de imágenes utilizando la función **cap.read()**de OpenCV. Por último, la almacena en la memoria flash de la BeagleBoneBlack.

Este proceso es representado de la siguiente manera:

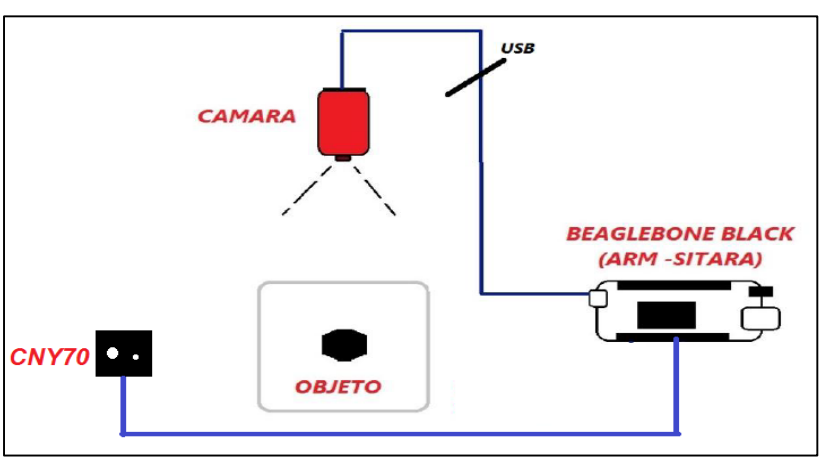

Figura 4.2 Adquisición y almacenamiento de la imagen

4.2.2 Escala de grises y binarización de la imagen

Al capturar la imagen se encuentra en formato RGB (Red Green Blue) que está compuesto de 24 bits, por lo que es demasiada información para procesar. El convertirla a escala de grises, se reduce la información a 8 bits que representan la luminancia o brillo de cada pixel. Esto es realizado con la siguiente función.

```
cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

Con la imagen almacenada en escala de grises, se procede a la binarización para hacer más eficiente el estudio de la imagen. Además, reduce aún más la información de 8 a 1 bit. Esto quiere decir que la imagen solo cuenta con dos colores: blanco (1) y negro (0).

Para indicar a cada pixel que valor debe tomar, es aplicado un parámetro conocido como valor de umbral. Este método sirve para separar los objetos que queremos analizar en una imagen. La separación o segmentación, se basa en la variación de intensidad entre un objeto y el fondo. El valor de umbral conveniente para este proyecto es de 200, la binarización se lleva a cabo con la función:

```
cv2.THRESH_BINARY
```

4.2.3 Obtención del contorno de una imagen

El primer paso es detectar el contorno de un objeto, esto se realiza con la función ***findContours***, la cual devuelve un conjunto de puntos que representan la frontera de cada uno de los elementos que se encuentren en la imagen. Estos contornos son enumerados de acuerdo a su tamaño, por lo que el propio perímetro de la imagen siempre será el contorno más grande y al cual se va a descartar en cada captura.

En esta aplicación solo es posible detectar una pieza, ya que si fueran capturadas dos o más, el programa detectaría a los dos elementos en una sola imagen unificada y cambiaria los valores obtenidos con la función frontera y el centroide sería diferente.

La obtención de la función frontera se lleva a cabo con el siguiente fragmento de código:

```
1 contours, hierarchy = cv2.findContours(thresh,cv2.RETR_TREE,
  cv2.CHAIN_APPROX_NONE)

2 for i in range( ):
    lista.append(len(contours[i]))
3 lista.remove(max(lista))
  print "elemento_mayor2:", lista.index(max(lista))
4 cnt_index=lista.index(max(lista))
  best_cnt= cnt_index +1
  print "best_cnt:", best_cnt
  print "# de contornos:",  len(contours)
5 cntx = contours[best_cnt]
```

Figura 4.3 Obtención de la función frontera

Primeramente se encuentra los contornos de una imagen binaria y guarda cada una de las coordenadas de los puntos que lo conforman, para después formar un vector llamado “lista” con la cantidad de contornos que detecta en la imagen.

Después, se remueve el perímetro de la propia imagen ya que no es de utilidad, con esto el contorno más grande es agrupado en el vector “lista” y es representado por la variable *cntx*. Ya que se tiene la información del contorno del objeto, se determina el área, perímetro y centroide con las siguientes funciones:

```
1 punto = puntos_cntx/32.0
2 area = cv2.contourArea(cntx)
  print "Area pixeles:  ",area

3 perimeter = cv2.arcLength(cntx,True)
  print "Perimetro: ", perimeter
  cx = int(M['m10']/M['m00'])
4 cy = int(M['m01']/M['m00'])
  print "Centroide Cx", cx
  print "Centroide Cy", cy
```

Figura 4.4 Funciones para la obtención de parámetros

Con estas funciones el contorno del objeto se divide en 32 partes, se obtienen las coordenadas del centroide, el área y perímetro de la figura en pixeles.

4.2.4 Valor de compacidad

Una vez obtenidos estos valores debemos ocupar la ecuación de la compacidad donde el concepto se relaciona con la geometría de los objetos y tiene múltiples aplicaciones en diferentes campos de la ciencia. En su forma clásica relaciona el perímetro con el área de cualquier objeto, y es válido para dos y tres dimensiones, Al desarrollar la ecuación de la compacidad discreta para el análisis del mundo digital, donde todos los objetos están compuestos por elementos discretos y finitos llamados pixeles, Ernesto Bribiesca explicó que “la compacidad clásica es la relación entre el perímetro y el área de un objeto. En su fase discreta se basa directamente en el perímetro de contacto de cualquier figura compuesta por celdas regulares (triángulos, rectángulos o hexágonos)”.

El perímetro de contacto es la vecindad de los pixeles que se tocan. Ahora bien, en esa vecindad cuenta el número de lados de estos últimos, que componen una forma. Así, entre más lados se toquen, la forma será más compacta (un círculo), y entre menos, será menos compacta (una amiba o una araña), El hecho de que dependa más de cómo se tocan los pixeles vecinos que del perímetro hace que la compacidad discreta sea una herramienta de medición más sencilla, apta y robusta para la clasificación de formas [21].

Ecuación de la compacidad: $C = \frac{A}{P^2}$

4.3 Control HID

Después de hacer el análisis de la imagen y usando la ecuación de compacidad, el valor entregado por el programa es procesado y se transmite un número (1, 2 o 3) a través del módulo UART dela BeagleBoneBlack hacia Arduino Leonardo. Este último es configurado como control HID, el cual interactúa con la interface AR DroneNavigation para controlar el dron. La conexión de estos dispositivos se presenta en la Figura 4.5.

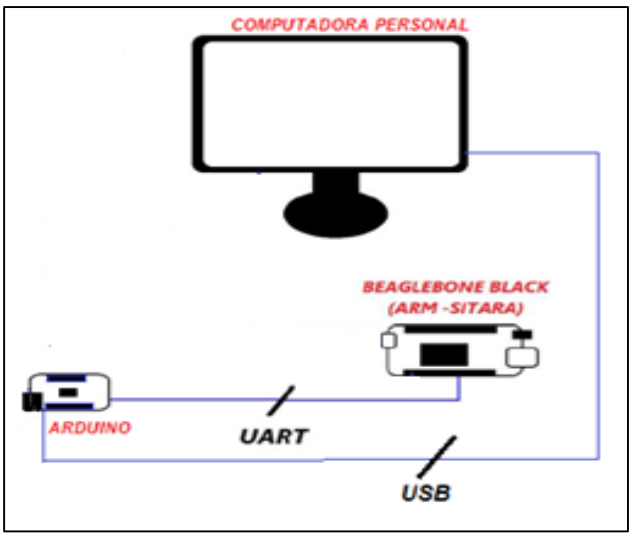

Figura 4.5 Conexión de tarjetas y PC

4.3.1 Configuración de Arduino

Se optó por la conectividad *HID* entre el Arduino y la PC ya que la interfaz AR DroneNavigation permite conectividad con diferentes controles remotos como: PlayStation 3, Microsoft XBOX, Nintendo Wii, Joystick *HID*, entre otros.

Arduino Leonardo ofrece la posibilidad de emular un Joystick *HID* al implementar la biblioteca **<joystick.h>**. Esta solo es posible operarla en tarjetas que cuenten con el microcontrolador ATmega32u4, como Arduino Micro o Leonardo.

La biblioteca antes mencionada cuenta con una serie características de implementación fácil como:

- Botones
- Mandos analógicos de 3 ejes
- Movimientos rotatorios
- Subir y bajar, acelerador y freno

La configuración se lleva a cabo siguiendo la siguiente metodología:

1. Incorporar la biblioteca**<joystick.h>**
2. Definir variables para cada característica que se quiera implementar
3. Inicializar el Joystick con la instrucción **Joystick.begin();**

4. Inicializar la comunicación serial
5. Cargar datos de navegación
6. Iniciar interrupciones externas y crear las rutinas

Para este proyecto se hace uso de:

- Tarjeta arduino Leonardo
- Periférico USB para la comunicación con el pc
- Software de calibración de movimientos

4.3.2 Calibración de Arduino como control HID

Ya que se tiene el programa, se pasa a calibrar el dispositivo reconocido como *HID* en el ordenador. La calibración se lleva a cabo con la herramienta “Dispositivos de juego” que se localiza en el panel de control de *Windows*.

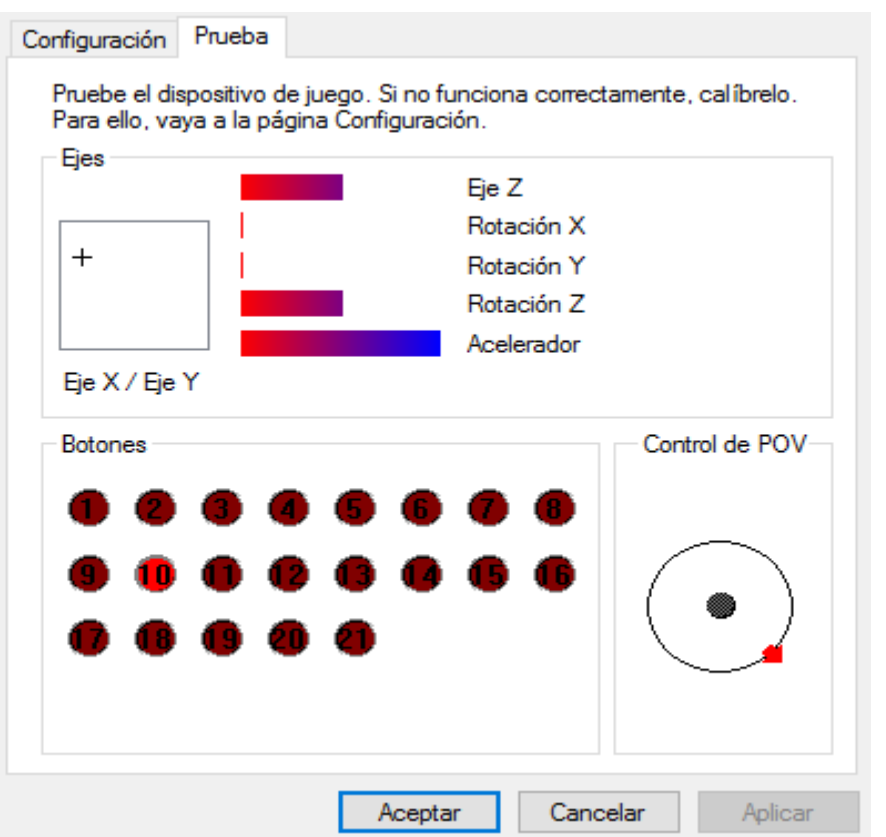

Figura 4.6 Ventana para calibración de control HID

Como se aprecia en la Figura 4.6, se cuenta con un espacio de trabajo con un cursor ubicado en el centro de un cuadro blanco, que simula un plano cartesiano donde es posible visualizar los movimientos en el eje “x” y “y” .Además, cuenta con barras para visualizar el eje z y movimientos rotativos. También cuenta con un grupo de botones que indican el estado de cada uno de manera lumínica.

La calibración dio como resultado los siguientes valores:

- **Eje X:** Centro a derecha de 0 a 127, centro a izquierda de 255 a 128
- **Eje Y:**Centro a frontal de 0 a 127, centro a trasero de 255 a 128
- **Eje Z:** Centro a alto de 0 a 255, centro a bajo de 255 a 0

4.4 Rutinas de pilotaje automático

La programación de las rutinas ocupa una sola variable para el control de los tres ejes (x,y,z), ya que el dron realiza un movimiento a la vez y no combinaciones de estos tres.

Para evitar que el código de ejecución de las rutinas fuera interpretado como una sola función se crearon 3 **void** para cada una de las rutinas y en ellas se ejecutan las instrucciones correspondientes.

En el código de programación de Arduino se puede visualizar en el texto las anotaciones para explicar su función, Sin embargo, al usar Arduino como control *HID* ya no es posible utilizarlo como tarjeta de desarrollo con conectividad *COM*, ya que no se cuenta con instrucción alguna que termine la comunicación como dispositivo *HID.* Esto evita ejecutar una nueva rutina y transmitir valores por el puerto *COM* hacia la PC.

Esta problemática fue abordada con interrupciones externas, donde al recibir un flanco de subida, el programa sale de la función que está ejecutando para regresar a la condición inicial y recibir el dato de una nueva rutina.

El direccionamiento a una nueva rutina es llevado a cabo por otra interrupción externa y dependiendo del valor recibido, pone en

alto un pin de salida que a la vez se encuentra conectado a una interrupción externa. Cada interrupción direcciona a la función **void** de la rutina correspondiente a la imagen visualizada por la cámara conectada a la tarjeta BeagleBoneBlack.

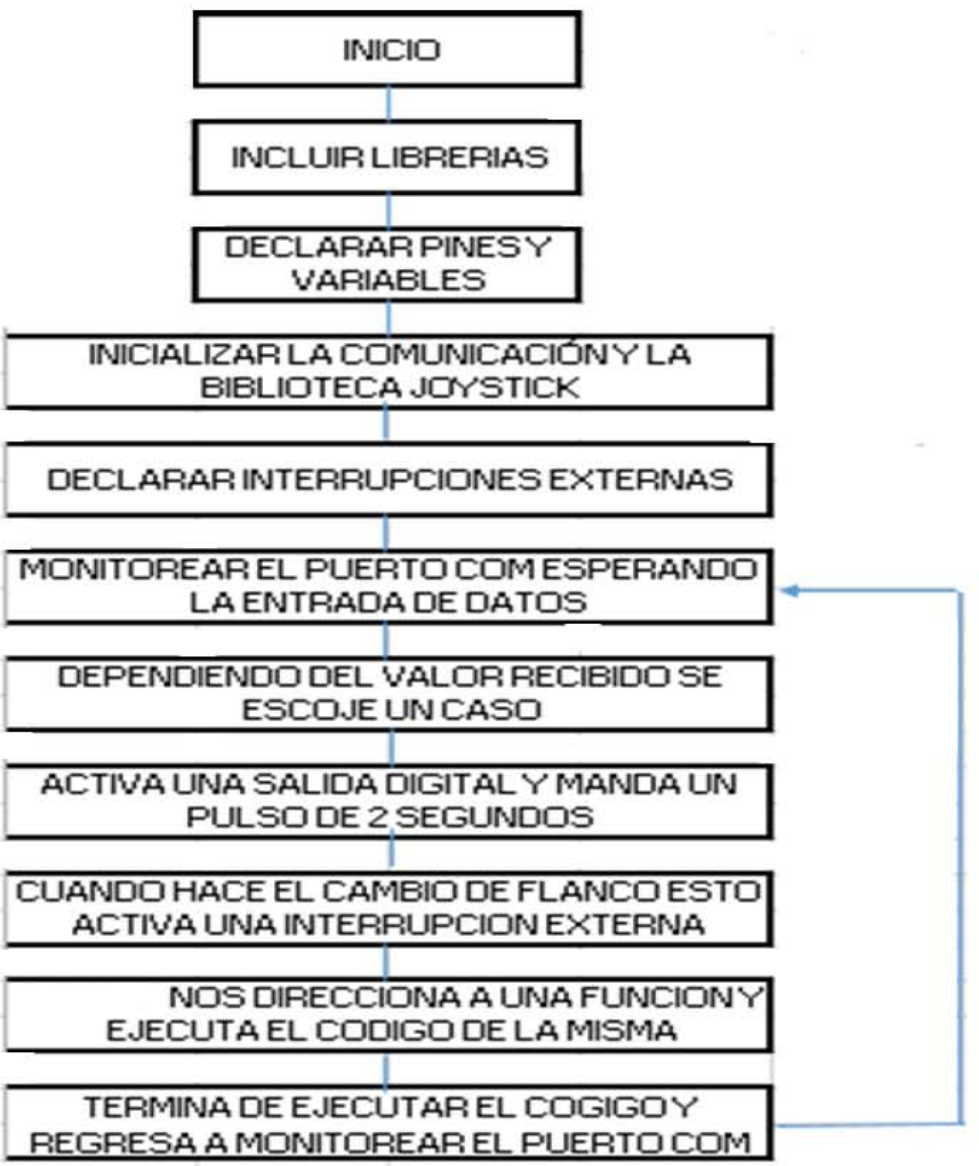

Figura 4.7 Diagrama de flujo

4.5 Interface de navegación y ejecución de rutina

Al recibir los comandos de navegación transmitidos por Arduino por el puerto *USB*, el SDK transmite estas instrucciones al dron para dirigirlo al contenedor correspondiente, como observa en la Figura 4.8.

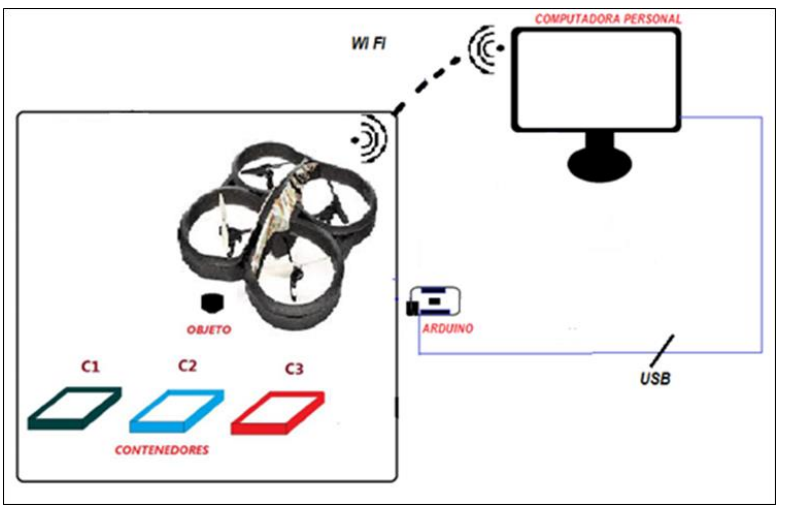

Figura 4.8 Transferencia de comandos y selección de contenedor

Al visualizar actividad de la cámara frontal y sensores de telemetría en la interfaz, se conecta el Arduino Leonardo a un puerto *USB* del ordenador.

Una vez reconocido Arduino como control *HID* en el *SDK*, se posiciona el *UAV* en la zona de trabajo correspondiente.

El sensor espera una pieza de manufactura para indicar a la BeagleBoneBlack que comience con la captura de imagen, realice el procesamiento de la misma y transmita según el procesamiento un numero al monitor serial de Arduino Leonardo el cual enviara los comandos para generar la rutina en el AR DRONE 2.0 y este realizara el viaje al contenedor según sea la clasificación del objeto y lo realizara una y otra vez cada que nuestro sensor inicie el ciclo.

Capítulo 5
Resultados experimentales

5.1 Resultados del de reconocimiento de piezas

Cuando la pieza es colocada dentro del rango de sensado del CNY70, indica a la BeagleBoneBlack que active la cámara y comience a capturar imágenes del área de trabajo.

En la Figura 5.1 se muestra la captura de las tres piezas con las que se trabajó en este proyecto:

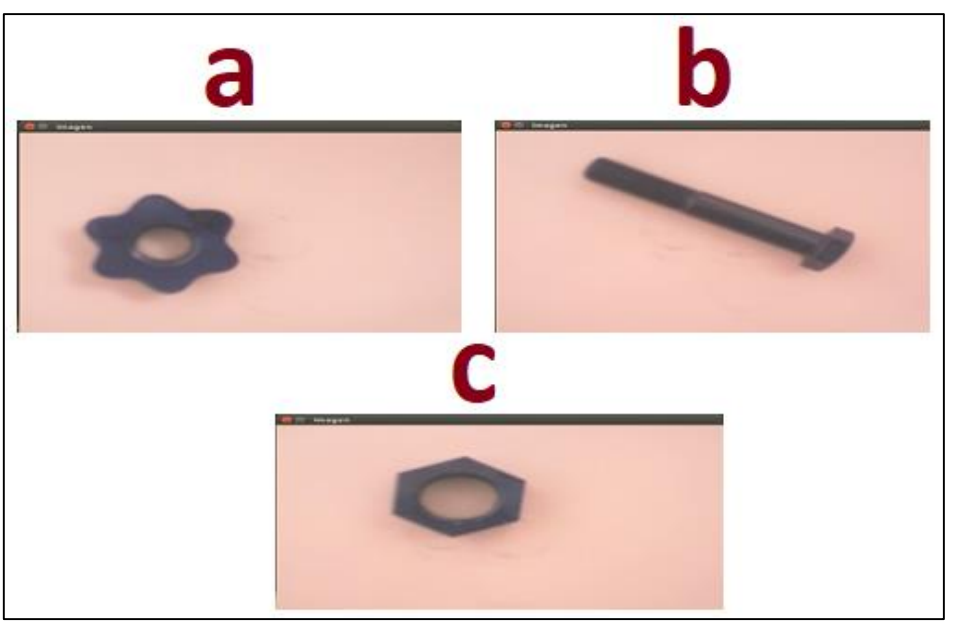

Figura 5.1 Captura de piezas (a, b y c)

En la siguiente figura se muestra el resultado de las imágenes convertidas a escala de grises:

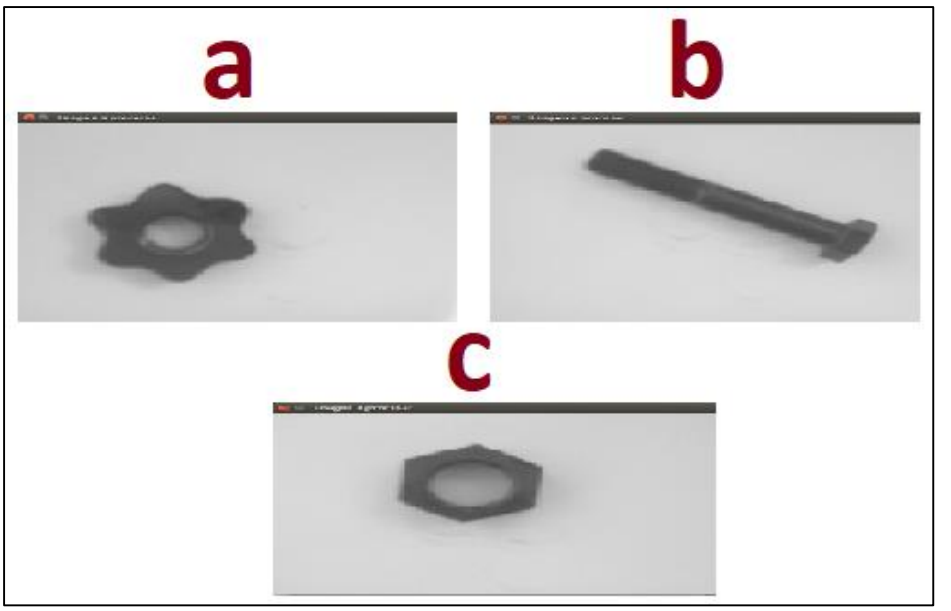

Figura 5.2 Imágenes a escala de grises (a, b y c)

A continuación, se muestra gráficamente la obtención del contorno de cada una de las piezas:

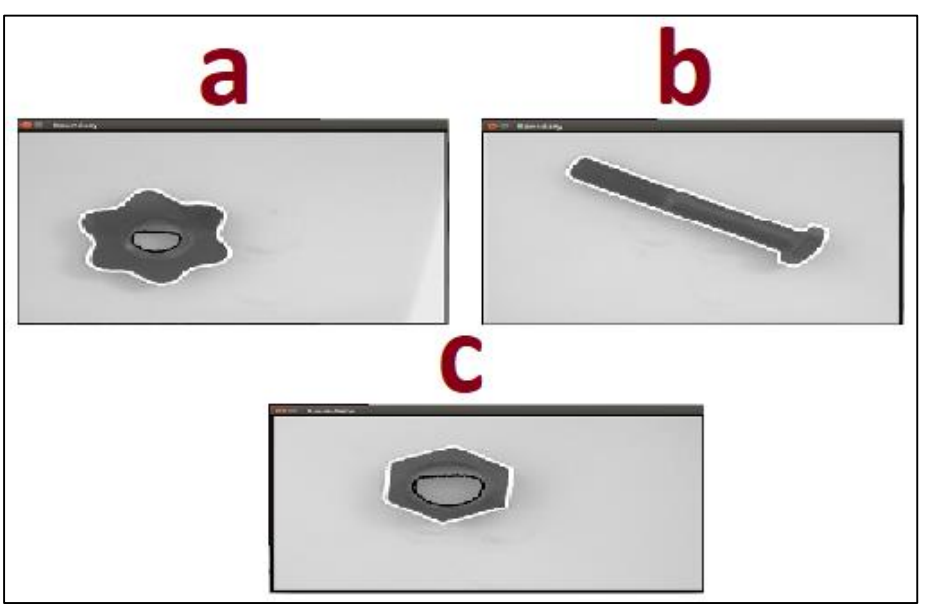

Figura 5.3 Obtención de contorno (a, b y c)

En la Figura 5.4 se muestran las tres piezas binarizádas con umbral de 200.

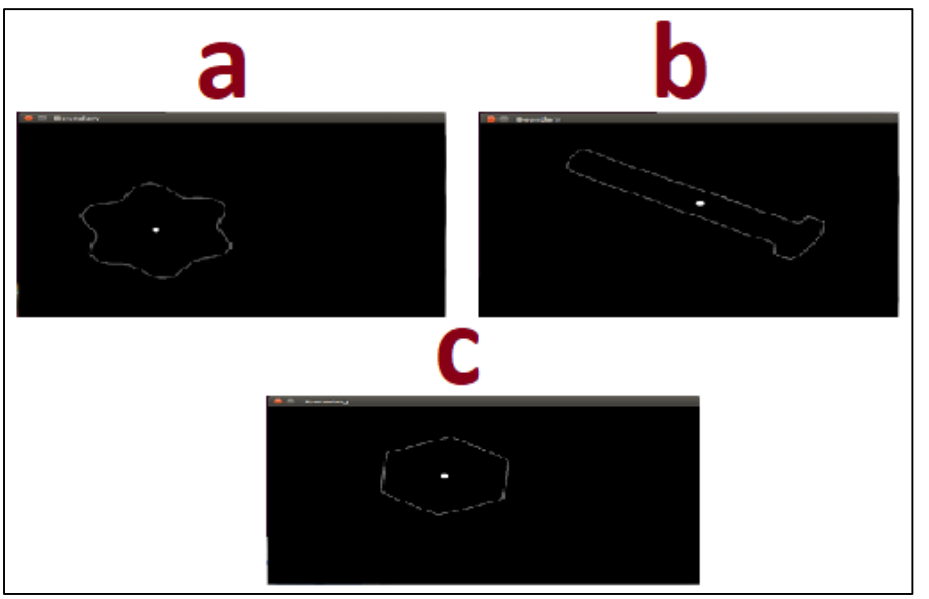

Figura 5.4 Binarización con umbral de 200 (a, b y c)

5.2 Recopilación de datos procesados

Tras haber realizado el procesamiento de imágenes, el programa brinda informacion sobre cada proceso. Esta información fue recopilada y registrada.

a)Tabla de informacion 1 (estrella).

TABLA DE DATOS A						
ITEM	ÁREA	PERIMETRO	CENTROIDE X	CENTROIDE Y	COMPACIDAD	
1	35540	820.264061	224	177	18.931714	
2	33289	854.832602	351	240	21.951358	
3	35333	787.636644	206	263	17.557849	
4	35790	798.849847	205	264	17.83046	
	34988	815.3957885	246.5	236	19.06784525	promedios

Tabla 5.1 Tuerca estrella.

b) Tabla de información 2 (tornillo).

TABLA DE DATOS B						
ITEM	ÁREA	PERIMETRO	CENTROIDE X	CENTROIDE Y	COMPACIDAD	
1	26871	1105.994071	336	197	45.522045	
2	26424	1114.101716	320	201	46.973505	
3	26560	1097.81327	309	207	45.376279	
4	28049	1142.170696	332	183	46.509818	
	26976	1115.019938	324.25	197	46.09541175	promedios

Tabla 5.2 Tornillo.

c) Tabla de información 3 (tuerca).

TABLA DE DATOS B						
ITEM	ÁREA	PERIMETRO	CENTROIDE X	CENTROIDE Y	COMPACIDAD	
1	28828	672.582822	261	187	15.691954	
2	27552	672.724957	345	271	16.425626	
3	28118.5	666.541191	332	200	15.800172	
4	29110	698.582822	278	175	16.764615	
	28402.125	677.607948	304	208.25	16.17059175	promedios

Tabla 5.3 Tuerca.

5.3 Ejecución de rutinas

Con los valores obtenidos se hace un promedio y el valor obtenido en el promedio después de analizar y procesar las imágenes nos indica de que figura se está hablando y esto envía un numero por la uart al arduino Leonardo dependiendo el valor, si se trata de la estrella envía un “1” al monitor serial del arduino Leonardo y esto activa la rutina que deberá de hacer el AR DRONE para clasificar y acomodar al objeto, lo mismo sucederá con las otras figuras cuando sea el tornillo enviara un “2” por el monitor serial y si se tratase de la tuerca enviara un “3” y estos números al igual que el uno activaran una rutina que el AR DRONE ejecutara automáticamente.

RUTINA “1”

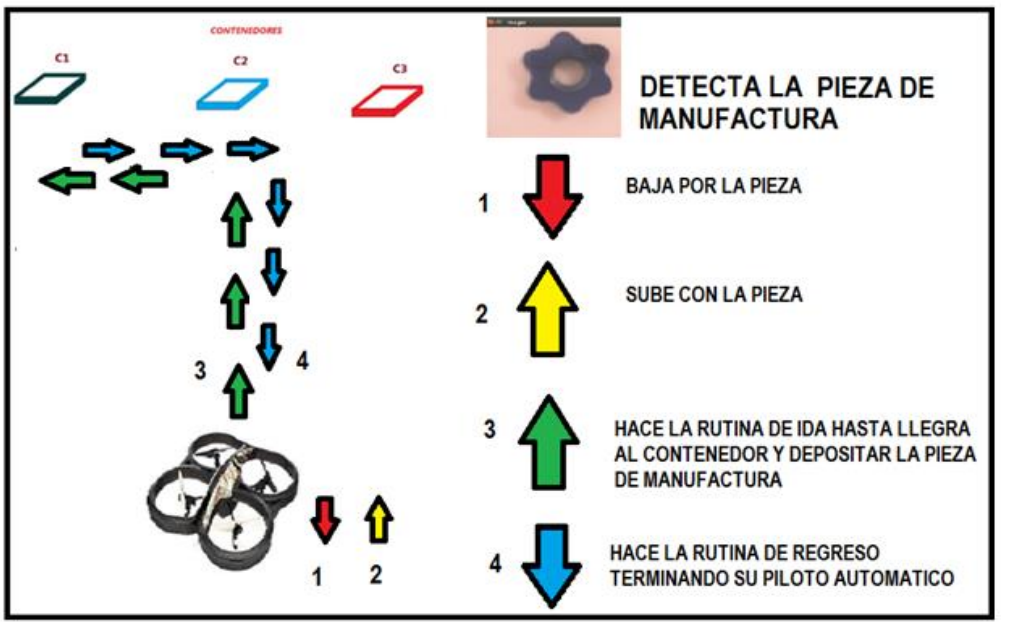

Figura 5.5 Rutina 1

RUTINA“2”

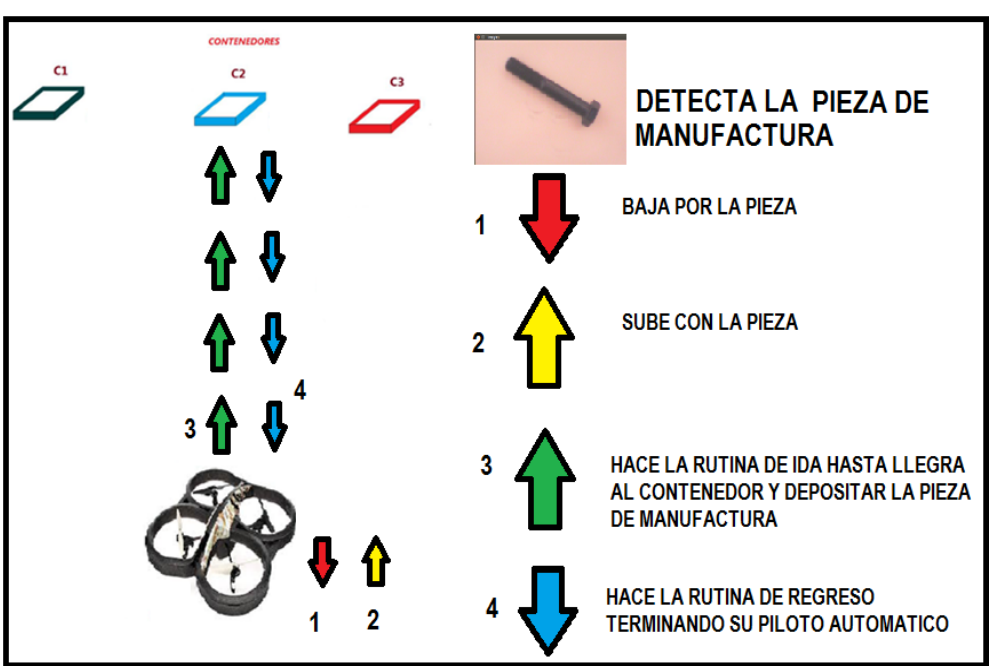

Figura 5.6 Rutina 2

RUTINA “3”

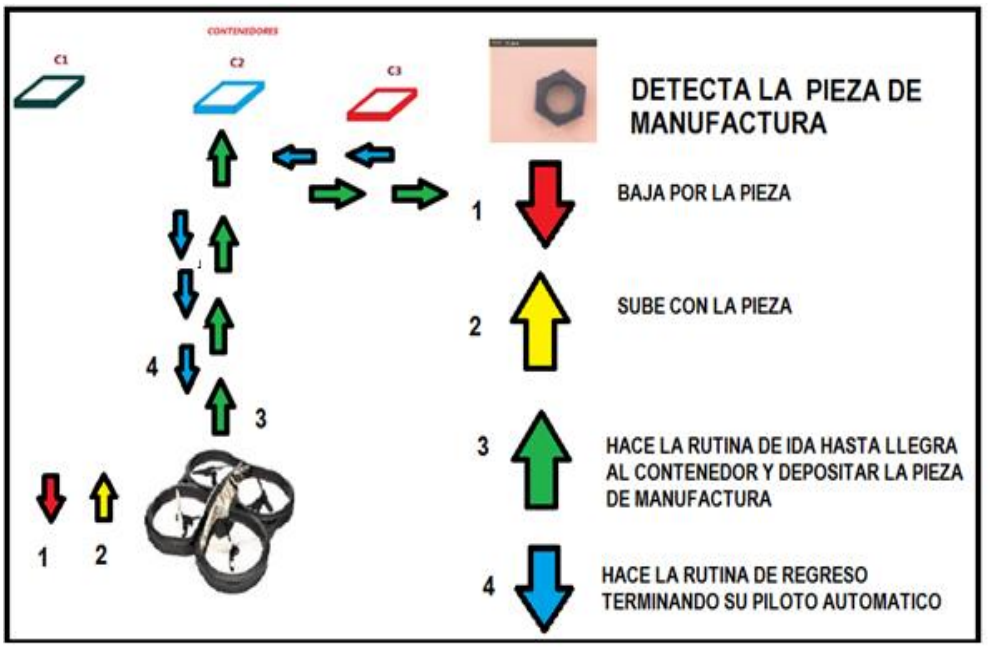

Figura 5.7 Rutina 3

5.4 Montaje del sistema autónomo

En primer lugar se monta el sistema completo en el lugar destinado para el experimento el cual consiste en: la pieza de manufactura según la rutina elegida, para este caso es la tuerca estrella de una pulgada sobre un fondo color rosa el cual me dio una mejor respuesta, el proto board con el sensor cny70 en la zona de sensado, la web cam, las tarjetas arduino Leonardo y la BeagleBoneBlack, una computadora y el AR DRONE.

Como se muestra en la imagen 5.8 podemos observar la tuerca estrella sobre la zona de trabajo la cual es un fondo color rosa que para este experimento ocupe una hoja de color rosa.

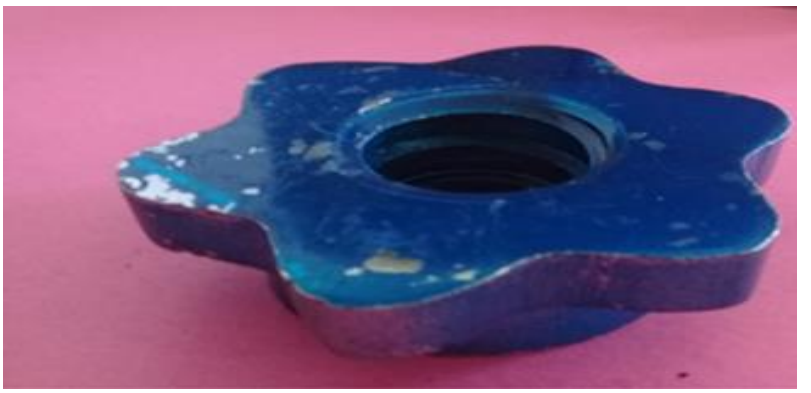

Figura 5.8 Tuerca estrella de 1 pulgada

En la siguiente imagen podemos observar la zona de sensado en la cual está el proto board con el sensor cny70 activo detectando la pieza de manufactura sobre el área de trabajo.

Figura 5.9 Zona de sensado con CNY70

Como podemos observar en la figura 5.10 se encuentra la tarjeta BeagleBoneBlack conectada al proto board y a la tarjeta arduino Leonardo y esta a su vez conectada a la pc.

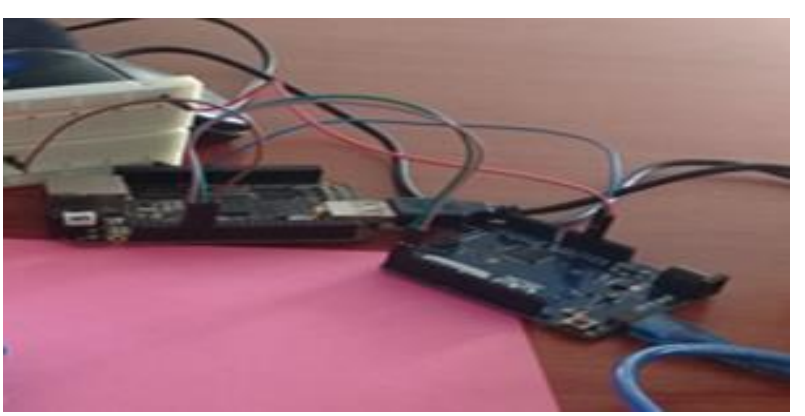

Figura 5.10 Tarjeta BBB y arduino Leonardo

En la imagen que mostrare a continuación se observa la pc y el AR DRONE ubicado en el área delimitada para el experimento dentro del Centro de Investigaciones Multidisciplinarias Aragón.

Figura 5.11 PC con interfaz y AR DRONE

Por ultimo en la figura 5.12 muestro el sistema completo con el cual logre realizar mi experimento.

Figura 5.12 Sistema autónomo en un cuadricóptero

5.5 Descripción de la primera rutina del AR DRONE

1.-Inicio del sistema

Al principio, después de haber montado el sistema autónomo y estar en una zona despejada para no causar algún accidente ni a las personas ni al equipo, se conecta la alimentación al circuito de sensado, se corre la interfaz y estamos listos para conectarnos y recibir la primer pieza de manufactura que en este caso será la tuerca estrella de 1 pulgada que es la de la primer rutina como se muestra en la figura 5.13

Figura 5.13 Inicio del sistema

2.-Posicion estable esperando código de reconocimiento

Como se menciona en el paso anterior al ejecutar la interfaz en Ubuntu debemos conectarla a la red que genera el AR DRONE y esta comienza a enviar los datos de telemetría, con esto podemos saber que ya están enlazados el AR DRONE y la interfaz de vuelo la cual con el botón de *take off* hace el levantamiento del cuadricóptero sobre su mismo lugar es decir en coordenadas espaciales solo realiza un movimiento en dirección del eje Z positivo conservando su ubicación en los ejes X y Y , este se levanta aproximadamente a 1 metro con respecto al suelo y permanece estable esperando las siguientes instrucciones que determina el código según la rutina correspondiente a la pieza de manufactura.

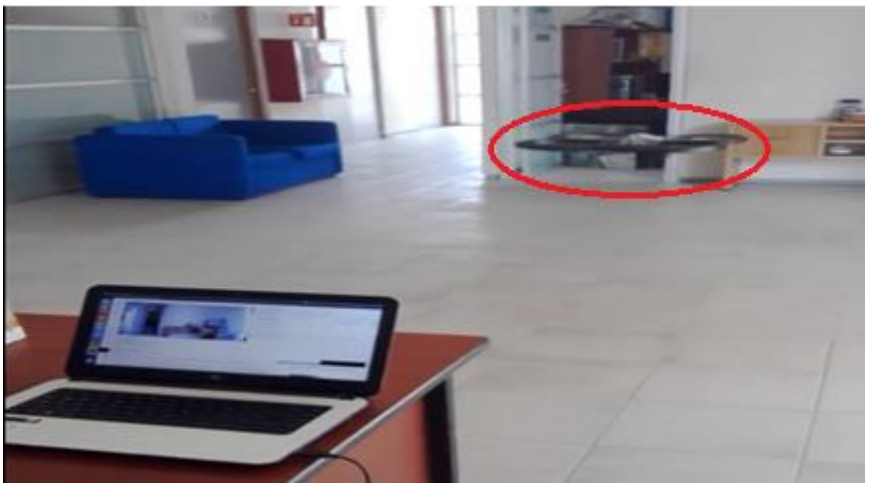

Figura 5.14 Esperando código de reconocimiento

3.- Baja por el objeto reconocido

Una vez colocada la pieza el sensor cny70 manda la señal para realizar el reconocimiento de la pieza y poder enviar el número que resulta del procesamiento digital de la imagen al arduino Leonardo para activar la primer rutina en el AR DRONE la cual comienza con el descenso del cuadricóptero por la pieza de manufactura como se ve en la figura 5.15.

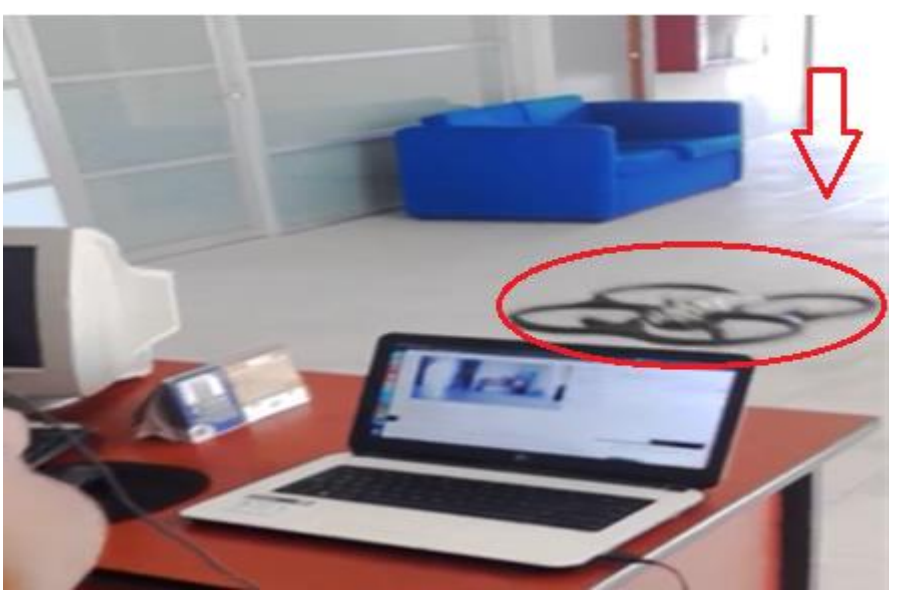

Figura 5.15 Baja por pieza de manufactura

4.-Sube con el objeto y avanza

Una vez tomado el objeto, el AR DRONE sube a la altura de trabajo y comienza su recorrido hacia adelante al área de trabajo com se muestra en la figura

Figura 5.16 Avanzando con la pieza

5.-Se desplaza a su izquierda

Hace su recorrido a la izquierda según la distancia que uno le programe esta la podemos variar según nuestras necesidades como se ve en la siguiente figura.

Figura 5.17 Desplazamiento a la izquierda y deposita la pieza

6.-Regresa al centro y se desplaza hacia atrás

Una vez que llega a su recorrido de trabajo donde se encontraría el contenedor de la pieza clasificada este soltaría el objeto y regresaría al centro de la mesa de trabajo y luego hacia atrás al origen de salida en el área de trabajo como se ve en la figura 5.18.

Figura 5.18 Regresa al centro

7.-Regresa a el área de trabajo en espera de una nueva tarea

Pues como se ve en la figura 5.19 termina el recorrido en el área de trabajo de donde partimos y se mantiene esperando la siguiente rutina.

Figura 5.19 Regresa al área de trabajo

Capítulo 6
Conclusiones

Objetivo 1.-Desarrollar una serie de rutinas de auto navegación dentro de una tarjeta de desarrollo que sea capaz de emular un control HID (Human Interface Device) mediante USB (Universal Serial Bus) y cuente con un puerto serial.

EL primer objetivo fue logrado debido a que todo el control de vuelo fue realizado en el entorno de programación de Arduino desde la entrada de datos, el procesamiento de instrucciones de la biblioteca de joystick.h y él envió de comandos por el puerto USB a la interface de AR Drone navigation, ya que al inicializar la biblioteca antes mencionada se está emulando el control HID y para esto era indispensable que la tarjeta fuera un Arduino Leonardo por que la librería solo se puede correr con el procesador del ATmega 32u4.

Objetivo 2.-Implementar algoritmos de reconocimiento de objetos utilizando visión artificial dentro de una tarjeta que cuente con un procesador ARM y ejecute un sistema operativo.

Se cumple el objetivo número dos al ocupar la BeagleBoneBlack que cuenta con un procesador ARM cortex 8 a la cual le cargamos el sistema operativo Debian y para hacer el procesamiento digital de las imágenes lo realice con ayuda de las *bibliotecas de software libre de Open CV.*

Objetivo 3.-Entablar comunicación entre un UAV y PC para transmitir comandos de conducción aérea de manera remota a medio alcance con un control HID.

Al utilizar la interface de AR Drone navigation esta proporciona una comunicación wifi conectándonos a la red que genera el AR Drone desde nuestra pc y con nuestro arduino emulamos el control HID que enviara los comandos de conducción aérea via USB a la interface de AR Dron navigation.

6.1 Trabajo futuro

Para un trabajo futuro se pretende implementar todo el proyecto únicamente en la tarjeta de desarrollo BeagleBoneBlack y así reducir costos al ya no requerir la tarjeta ni el ordenador e implementar este proyecto con otros dispositivos que estén sincronizados y así trabajar en el campo de la robótica cooperativa.

Referencias bibliográficas

[1] BRIDGEs, Valery. HISTORIA DE LAS COMUNICACIONES TRANSPORTES AEREOS, Editorial: Salvat., Pamplona, (1971)

[2] Grégoire, Chamayou. TEORÍA DEL DRON NUEVOS PARADIGMAS DE LOS CONFLICTOS DEL SIGLO XXI, NED Ediciones Barcelona 2016.

[3] Francisco Rey Sacristán. *Mantenimiento Total de la Producción TPM Proceso de Implantación y Desarrollo.* FC Editorial, 2001 - 350 páginas

[4] Julio cesar Neffa. *Los riesgos psicosociales en el trabajo contribución a su estudio.* Primera edición ceil conicet

[5] Carmona Fernández, J. (2013). DISEÑO DE UN SISTEMA DE CONTROL PARA UN CUADRICÓPTERO. Licenciatura. Universidad Carlos III de Madrid.

[6] Parrot AR Drone 2.0 Especificaciones,

http://ardrone-2.es/especificaciones-ar-drone-2/

[7] Redactor: Juan Manzano Ulmeher, http://www.ibertronica.es/blog/tutoriales/que-son-los-procesadores-arm

[8] Joseph, Yiu.THE DEFINITIVE GUIDETO THE ARM CORTEX-M3 SECOND EDITION, Newnes is an imprint of Elsevier inc. 2011. PP 2

[9]Ontiveros Salgado, M. (2016). RECONOCIMIENTO DE OBJETOS Y MANEJO DE UN BRAZO ROBÓTICO MEDIANTE UN PROCESADOR SITARA. Maestría. Universidad Nacional Autónoma de México.

[10]Procesamiento Digital de Imágenes Apuntes del curso impartido por el Dr. Boris Escalante Ramírez Agosto, 2006 UNAM

[11]Adrian kaheler y Gary Bradski. *Learning Opencv 3 computer visión in c++ with the opencv library*. O´REILLY.

[12] Coley, Gerald,BeagleBoneBlack System Reference Manual, This work is licensed under the Creative Commons Attribution-Share Alike 3.0 UnportedLicense. All derivative works are to be attributed to Gerald Coley of BeagleBoard.org, TEXAS INSTRUMENTS.

[13]ARDUINOPRODUCTS>Arduino Leonardohttps://www.arduino.cc/en/Main/ArduinoBoardLeonardo

[14] Datasheet atmega32u4

[15] Datasheet CNY70

[16]Universidad de Castilla-La Mancha Sensor CNY70

http://www.infoab.uclm.es/labelec/solar/otros/infrarrojos/sensor_cny70.htm

[17] Raggi, Emilio. Thomas, Keir. Parsons, Trevor. Channelle, Andy, Van Vugt, Sander. BEGINNING UBUNTU LINUX, FIFTH EDITION, Apress.

[18]Debían The universal operating system. https://www.debian.org/intro/about#what

[19] Lentin Joseph. *Learning Robotics Using Python*. Packt, 2015

[20] Joseph Howse. *OpenCV Computer Vision with Python*.Packt, 2013

[21] http://www.dgcs.unam.mx/boletin/bdboletin/2012_490.html.