UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD 5

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Título:

Sistema de control para el direccionamiento horizontal y vertical de una cámara de vigilancia Pant Tilt basada en Motores de paso y Microcontroladores AVR.

Autor: Yassiel Gómez Díaz

Tutores: Ing. Julio Alberto Leyva Durán.
Ing. Yordan Carlos Pérez Attanaf

Ciudad de la Habana, junio de 2016.
"Año 58 de la Revolución."

DECLARACIÓN DE AUTORÍA.

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____
Yassiel Gómez Díaz

_____ _____

Ing. Julio Alberto Leyva Durán. Ing. Yordan Carlos Pérez Attanaf.

Datos de contacto.

Tutor: Ing. Julio Alberto Leyva Durán.
Edad: 33 años
Ciudadanía: cubana
Institución: Universidad de las Ciencias Informáticas (UCI)
Título: Ingeniero en Ciencias Informática
Categoría Docente: Instructor
E-mail: jaleyva@uci.cu

Tutor: Ing. Yordan Carlos Pérez Attanaf.
Edad: 24 años
Ciudadanía: cubana
Institución: Universidad de las Ciencias Informáticas (UCI)
Título: Ingeniero en Ciencias Informática
Categoría Docente: Instructor
E-mail: yattanasov@uci.cu

Dedicatoria

A mis padres y familiares por ser mis guías y ejemplo en todo momento.
A mis abuelos Pablo y Maria Dolores, mi tía Magalys, que Dios los tenga en su gloria, que siempre se preocuparon por mis estudios y me hubiese gustado muchísimo que hoy me vieran ya graduado.

Agradecimientos

Especialmente a mis padres estos que han sabido guiarme en todo el transcurso de mi vida y continuamente me han ofrecido lo mejor.

Agradezco a mi familia, no solo por todo el trabajo que han asumido a mi lado, sino también por el apoyo emocional y la seguridad que me han dado durante todo este proceso.

Agradezco además a mis tutores por el apoyo brindado.

A mis compañeros, tanto los que iniciaron conmigo en aquel grupo 5105 como los que terminaron, los del grupo 5502 , en especial a Juan Gabriel, Rogelio, Luis Alberto , Lachi, Isasi, Javier Rodriguez, la Xiuny a todos muchas gracias y feliz de haber compartido buenos y malos momentos con ustedes.

Agradezco también a todos mis amigos de mi barrio los palseros Lagnier, Etian, Alejandro, Carlos, Yaneydi, El yayo, Yobal que siempre me apoyaron.

Resumen

El desarrollo de la automatización de las cámaras de vigilancia en el mundo, ha aumentado gracias a las ventajas de las Tecnologías de la Información y las Comunicaciones (TIC). Su aseguramiento es un aspecto esencial para lograr la monitorización ante cualquier escenario de visualización de objetivos; las funciones que intervienen en ellas se llevan a cabo gracias al empleo de dispositivos que están compuestos por motores de pasos y sistemas embebidos.

El presente trabajo persigue como objetivo el desarrollo de una aplicación para *Smartphone* que permita controlar remotamente el posicionamiento horizontal y vertical de una cámara de vigilancia Pant Tilt (PT) mediante el uso de motores de paso y microcontroladores AVR. Para el desarrollo del sistema se decidió emplear: la metodología de desarrollo AUP-UCI, los lenguajes de programación Arduino para el sistema embebido y Java para la aplicación del dispositivo móvil. El estudio del estado del arte permitió desarrollar una aplicación basada en las buenas prácticas empleadas por los sistemas estudiados y enfocada hacia la solución del problema identificado. Como resultado del trabajo realizado se logró el control mediante un *Smartphone* de una cámara de seguridad PT permitiendo un mejor empleo del sistema de vigilancia de las áreas en las entidades, instituciones, hogares y establecimientos.

Palabras clave: Arduino, cámaras de vigilancia, microcontroladores, motores de paso, sistema de control.

Índice

Contenido

Índice de figuras

Índice de tablas

Introducción

El desarrollo de la automatización de las cámaras de vigilancia en el mundo, ha aumentado gracias a las ventajas de las Tecnologías de la Información y las Comunicaciones (TIC). Su continuo avance ha llevado a que la gran mayoría de hogares, negocios e instituciones públicas y privadas tengan la necesidad de poseer equipos que le faciliten el resguardo de sus establecimientos. La calidad del servicio que brindan estos dispositivos es muy importante para mantener el control en la ciudad y la seguridad a los ciudadanos.

El aseguramiento de las plataformas de las cámaras de vigilancia es un aspecto esencial para lograr la monitorización ante cualquier escenario de visualización de objetivos; las funciones que intervienen en ellas se llevan a cabo gracias al empleo de dispositivos que están compuestos por motores de pasos y sistemas embebidos.

Cuba es un país que poco a poco ha ido creando varias instituciones con el objetivo de desarrollar y elevar el nivel científico de la sociedad, entre las que se encuentra La Universidad de las Ciencias Informáticas (UCI). Este es un centro con características especiales determinada por la combinación Docencia–Producción, tiene entre sus objetivos lograr una competitividad nacional e internacional de la industria cubana del *software*, así como insertarse en el mercado mediante sus líneas de productos y servicios.

En esta institución existe la línea de proyecto Sistemas Embebidos (LSE) del Centro de Informática Industrial (CEDIN), la cual tiene como principal objetivo el desarrollo de productos de *software* para sistemas con pequeña capacidad de cómputo. Para este tipo de trabajo se cuenta con la tarjeta Arduino, basada en microcontroladores AVR, y taxonomía de *hardware* y *software* libre, la misma puede ser utilizada para el desarrollo de Tarjetas de Adquisición de Datos (DAQT, por sus siglas en inglés) de procesos de mediana y baja complejidad.

En la actualidad existen varias entidades y ministerios radicados en Cuba como: Guarda Bosques, Defensa Civil, Ministerio de las Fuerzas Armadas (MINFAR), Ministerio del Interior (MININT), Guarda Frontera, Policía Nacional Revolucionaria (PNR), empresas, hoteles que realizan el sistema de vigilancia de sus áreas utilizando cámaras de visión diurna analógicas no robotizadas. Por la buena óptica que poseen estos artefactos, son cotizados a altos precios en el mercado internacional, incrementando así el valor del servicio de vigilancia que se realiza en cada entidad, al tener en cuenta el valor agregado

que surge por la necesidad de adquirir gran cantidad de equipos para la supervisión de todas las áreas.

Actualmente estas cámaras no son muy efectivas en áreas de gran amplitud debido a que al mantenerse visualizando en una posición fija, no ofrecen supervisión del campo necesario dejando puntos ciegos potenciales que pueden propiciar el desarrollo de actividades delictivas. Además de esta deficiencia, se afecta al presupuesto de las entidades al tener en cuenta que, para obtener un buen sistema de vigilancia al alcance de todas sus áreas, se necesitan más cámaras para cubrir dicho terreno, por lo que aumenta el costo total de adquisición de dicha tecnología.

Teniendo en cuenta la situación antes descrita, se define como **problema de investigación**: ¿Cómo controlar el direccionamiento de una cámara de vigilancia PT basada en motores de paso y microcontroladores AVR para el monitoreo de las áreas?
Lo cual enmarca como **objeto de estudio** el control del direccionamiento de cámaras de vigilancia PT y como **campo de acción**: control del direccionamiento de una cámara de vigilancia basada en motores de paso y microcontroladores AVR, mediante un *Smartphone*.
Para dar solución al problema planteado se ha formulado como **objetivo general**: Desarrollar un sistema de control que permita, mediante un *Smartphone,* el direccionamiento horizontal y vertical de una cámara de vigilancia PT.
Con el fin de darle cumplimiento a dicho objetivo se trazaron las siguientes tareas de investigación:

- Elaboración del marco teórico de la investigación a partir del estudio del estado del arte existente sobre el tema.
- Selección de tecnologías necesarias para el desarrollo de la investigación.
- Diseño del protocolo de comunicación entre el Arduino y el dispositivo móvil.
- Implementación de la persistencia de eventos de rotación.
- Implementación de la programación automática y del planificador de rotación horizontal/vertical.
- Integración del protocolo de comunicación entre el Arduino y el *Smartphone*.
- Realización de pruebas de aceptación para validar el correcto desarrollo de los requerimientos implementados en la solución.

Para el desarrollo de la investigación se emplean los siguientes **métodos científicos**:

De nivel teórico:

- **Histórico–Lógico**: En la presente investigación se utilizó este método para realizar el estudio del arte, es decir, para realizar el estudio acerca de los sistemas o soluciones similares, además de los lenguajes, herramientas y metodologías para el desarrollo del sistema que se propone.
- **Modelación**: Se empleó para realizar los diagramas necesarios para darle cumplimiento a los requisitos funcionales y no funcionales asociados al sistema.
- **Análisis y Síntesis**: Permitió realizar el análisis y estudio de las bibliografías existentes sobre el tema en cuestión y lograr obtener de manera sintetizada el contenido necesario y suficiente para la realización del presente trabajo.

De nivel empírico:

- **Consulta de toda fuente de información**: Se consulta la información referente al funcionamiento de las cámaras PT, así como otras fuentes que tributen al adecuado desarrollo de la investigación.
- **Consulta a especialistas**: Se realizaron encuentros con personal de gran experiencia en el tema con el objetivo de recopilar información, opiniones que permitan un mejor entendimiento y desarrollo de la investigación.
- **La observación**: Se empleó con el objetivo de observar el funcionamiento de algunos sistemas orientadas a la video-vigilancia; así como mecanismos para mantener un control automático y manual de estas cámaras. Este método proporciona la vía para realizar un registro visual de las características comunes en este tipo de sistemas e identificar las que puedan formar parte de la solución.
- **Pruebas de validación**: Se realizaron las pruebas pertinentes a la aplicación para así tener constancia del cumplimiento de los objetivos.

El documento contará para una mejor comprensión del mismo, con la siguiente estructura capitular:

CAPÍTULO 1. Fundamentación Teórica: En este capítulo se presentan los principales conceptos que se deben tener en cuenta para un mejor entendimiento del trabajo, además del estudio y selección de las herramientas, tecnologías y metodología para el desarrollo de la propuesta de solución.

CAPÍTULO 2. Análisis y Diseño: En este capítulo se establecerá la arquitectura y los patrones de diseño a utilizar, los requerimientos funcionales y no funcionales que debe cumplir el sistema, además de describir el funcionamiento general de la solución.

CAPÍTULO 3. Implementación y Pruebas: Se describe la implementación de la aplicación desarrollada, así como las pruebas realizadas a las funcionalidades trazadas.

Capítulo1: Fundamentación Teórica

1.1 Introducción

En el presente capítulo se abordan diversos conceptos relacionados con los sistemas de control, las cámaras de seguridad, los microcontroladores y motores dados por distintos autores. Se realiza además un estudio del estado del arte referente al tema de investigación a fines de obtener un mayor entendimiento y las tecnologías necesarias para su desarrollo. Además, se identificará la metodología y las tecnologías a usar para el desarrollo de la propuesta de solución.

1.2 Sistemas de control

En la actualidad los sistemas de control se han convertido en factor importante en el sector de la industria, proporcionando un mejor control de calidad en los productos manufacturados, líneas de ensamble automático, en el control de máquina-herramienta, en los sistemas de transporte y de potencia, además de la robótica. Estos sistemas han sido de gran impacto para el desarrollo de nuestra sociedad ya que han permitido una mejora en la seguridad de operación hacia las máquinas y procesos, además de una disminución en los costos de producción.

En algunas bibliografías consultadas se define como sistema de control:

- Arreglo de componentes físicos interconectados de forma que se puedan comandar dinámicamente (Ramirez, et al., 2007).
- Un sistema de control es una interconexión de componentes que forman un sistema de configuración que proporcionará una respuesta deseada del sistema (Dorf, y otros, 2001).

Teniendo en cuenta las definiciones anteriores se puede definir un sistema de control, como un conjunto de dispositivos que actúan como un sistema, para así lograr un objetivo de control.

Los sistemas de control se clasifican en sistemas de lazo abierto y de lazo cerrado. La distinción la determina la acción de control, que es la que activa al sistema para producir la salida. Un sistema de **control de lazo abierto** es aquel en el cual la acción de control es independiente de la salida ver ilustración 1 (Díaz , 2009). Los sistemas de control a lazo abierto tienen dos rasgos sobresalientes:

- La habilidad que éstos tienen para ejecutar una acción con exactitud está determinada por su calibración. Calibrar significa establecer o restablecer una relación entre la entrada y la salida con el fin de obtener del sistema la exactitud deseada.
- Estos sistemas no tienen el problema de la inestabilidad, que presentan los de lazo cerrado.

La principal desventaja es que las perturbaciones no pueden ser corregidas, por tanto, al no haber medición hay incertidumbre y no hay posibilidad de corrección. A continuación, una demostración del funcionamiento de un sistema de lazo abierto.

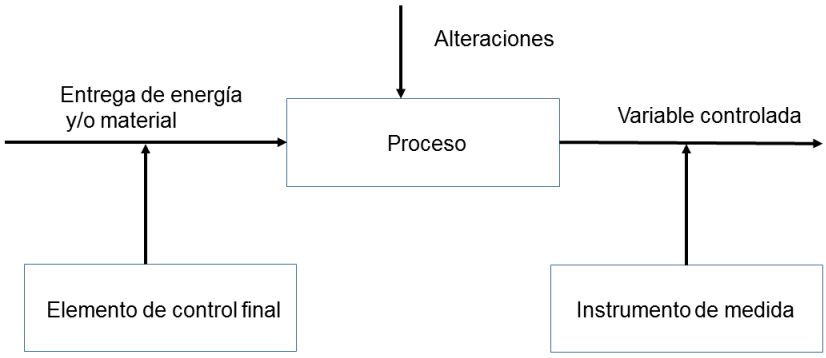

Ilustración 1. Sistema de control de lazo abierto

Un sistema de **control de lazo cerrado** es aquel en el que la acción de control es en cierto modo dependiente de la salida. Son llamados comúnmente sistemas de control por realimentación (Díaz , 2009).

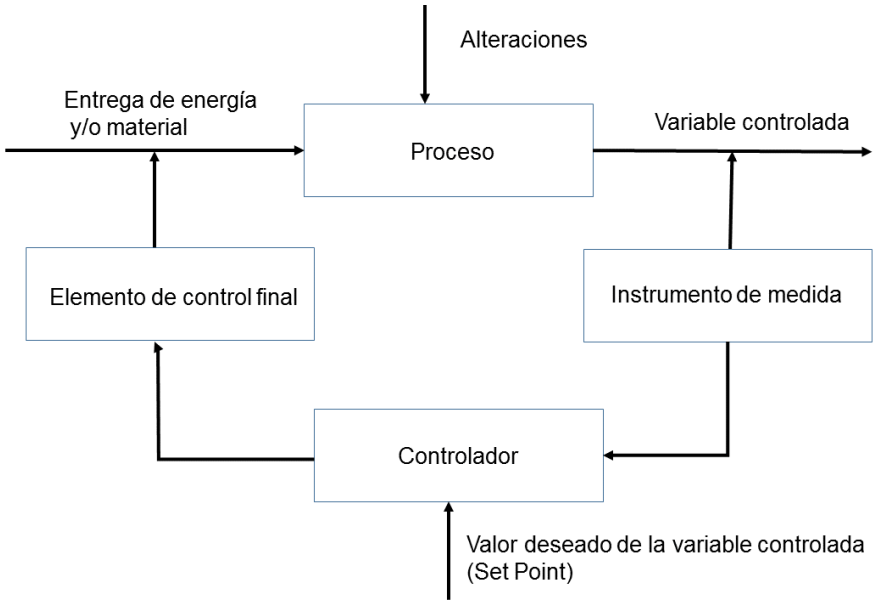

Ilustración 2. Sistema de control de lazo cerrado

1.3 Cámara de Vigilancia

Una cámara es un aparato o una máquina que a través de un lente permite registrar imágenes estáticas o en movimiento (Lopez Rodriguez, 2007). Actualmente las cámaras de vigilancia, representan un factor importante para el avance tecnológico en cuanto a la seguridad tanto en hogares como en sectores comerciales, empresas e instituciones. En casi la mayoría de todos los sistemas de video-vigilancia podemos encontrarnos con una gran variedad de tipos y modelos de cámaras de seguridad donde básicamente existen 2 tipos, estas son cámaras de seguridad analógica o de Protocolo de Internet (IP, por sus siglas en inglés).

Las cámaras de seguridad analógicas son las que se usaron toda la vida en CCTV (Circuito Cerrado de Televisión) y aún mantiene su uso vigente. La imagen sale en estos equipos de manera analógica, debido a una señal de corriente alterna que varía en el tiempo con diferente amplitud. Poseen una salida de 75 ohm, es por eso que se requiere el uso de cable Coaxial o Par Trenzado. Cuando se diagrama un sistema de CCTV con cámaras de seguridad analógicas, hoy en día, lo más recomendable es utilizar un Grabador de Video Digital (DVR) como se muestra en la ilustración 3, pero existen muchos otros dispositivos para interconectarlas, como podrían ser multiplexores, secuenciadores, etc.

Ilustración 3. Cámara analógica

Las cámaras IP son pensadas para ser visualizadas mediante internet o desde una red local. Por ejemplo, si se tiene una red de computadoras interconectadas entre sí mediante un *switch* o *router*, estas cámaras se conectarían como si fuera una computadora más. Las más clásicas y profesionales poseen un puerto *ethernet* con terminal Rj45 y se conectan al *switch* o *router* mediante cable UTP. También hay cámaras IP inalámbricas, las cuales se conectan a la red utilizando una antena que permite esta infraestructura. Estas últimas cámaras si bien pueden ser efectivas para algunos casos, no entran dentro de la categoría de cámaras de seguridad profesionales debido a que la gran mayoría son de escasa calidad y los enlaces inalámbricos son muy fáciles de bloquear. Por tanto, una cámara IP es una cámara analógica que en su salida tiene un conversor que hace digital la imagen (Nicolas Sosio, 2013).

Ilustración 4. Esquema del funcionamiento de las cámaras IP

Existe un gran número de cámaras de red que de acuerdo a su diseño se pueden clasificar en cámaras fijas o móviles. Las cámaras fijas disponen de un campo de vista fijo normal,

telefoto, de gran angular una vez montada, pueden instalarse en armazones diseñadas para su uso en instalaciones interiores o exteriores, este tipo de cámaras es la mejor opción en aplicaciones en las que resulta útil que la cámara esté bien visible (Nicolas Sosio, 2013). Existen también las cámaras domo fijas que consiste básicamente en una cámara fija preinstalada en una pequeña carcasa domo, esta cámara puede enfocar el punto que seleccionemos en cualquier dirección, la ventaja principal que tiene radica en su disimulado y discreto diseño, así como también en la dificultad de ver hacia qué dirección apunta nuestra cámara; así mismo es muy resistente a cualquier tipo de manipulaciones (Axis, 2016).

Las cámaras móviles se encuentran basadas en el funcionamiento Pant Tilt, lo que permite, por lo general, un movimiento de 360° horizontalmente y un giro 180° verticalmente.

1.3.1 Tipos de cámaras

En la actualidad existen un gran número de cámaras que garantizan la seguridad de nuestros establecimientos, algunas de estas son (CAMSEG, 2016):

Cámara Bullet: Es una cámara con montaje en la pared o en el techo, que está diseñada típicamente para uso en interiores, pero también se puede aplicar en exteriores. La cámara toma su nombre de su elegante forma cilíndrica delgada. Muchas cámaras de este tipo también son resistentes al agua. La cámara no está normalmente diseñada para tener el control de hacer una panorámica, inclinarse o hacer zoom, pero en lugar de eso se encarga de capturar imágenes de un área determinada. La unidad se monta apuntando a un área en particular.

Cámaras Domo: Las cámaras domo deben su nombre a su carcasa en forma de cúpula en la que se colocan. Estas cámaras están diseñadas para que sean discretas, no encubiertas u ocultas, es decir, la cámara está diseñada para ser discreta pero visible. Estas unidades tienen dos propósitos, los ladrones saben que están siendo vigilados y los propietarios sienten la seguridad de que la instalación está siendo protegida. Muchos tipos de cámaras domo pueden girar 360° a una rápida velocidad. Pueden hacer una panorámica, inclinarse y hacer zoom, y además de permitirle a los operadores moverlas hacia la izquierda o la derecha, hacia arriba o abajo, acercar y alejar, es decir, ejercer un control sobre ellas. Sin embargo, hay cámaras que están automatizadas para que se muevan en base a algo ya sincronizado. Estos domos muchas veces son utilizados para cubrir un área amplia con una sola cámara.

Cámaras infrarrojas o para visión nocturna: Estas cámaras de visión nocturna tienen la capacidad de ver imágenes en condiciones oscuras mediante el uso de LEDs.

Cámaras inalámbricas: No todas las cámaras inalámbricas están basadas en IP. Algunas cámaras inalámbricas pueden usar modos alternativos de transmisión basadas en este medio. Aunque el método de transmisión no es significativo para su comparación con otros tipos de cámara, sí se valora su principal ventaja que resulta ser: una gran flexibilidad en la instalación del dispositivo.

Cámaras de alta definición: Estas dan a los operadores la posibilidad de ampliar la imagen con una claridad extrema (mirar los jugadores de póker, por ejemplo, a alguien que podría tener algo bajo la manga).

Una de las más usadas en la actualidad son las cámaras domo y las IP debido a que cubren bastante terreno y son muy útiles por las ventajas que estas ofrecen. Estas cámaras pueden ser fijas o móviles, ejemplo de estas se encuentran las llamadas PTZ.

1.3.2 Cámaras de vigilancia Pant Tilt Zoom(P/T/Z)

Las cámaras PTZ, o de paneo, inclinación y ampliación ("Pan, Tilt and Zoom", en inglés), esto se refiere a las capacidades de las cámaras automatizadas y grabadoras de vídeo. Pueden hacer un barrido de 360° del ambiente, cambiar ángulos para mirar objetos por encima y por debajo de la cámara, y ampliar para darle más detalles a un objeto. Estas cámaras son usadas más a menudo en situaciones de vigilancia donde los espectadores remotos puedan rastrear individuos (eHow, 1999-2016).

Ilustración 5. Cámaras PTZ

Hoy en día existen las cámaras PTZ y las cámaras domo PTZ. Todos los comandos se envían a través del mismo cable de red que la transmisión de vídeo. Aunque ambas

poseen características y funcionalidades similares, existen algunas diferencias entre ellas como se exponen en lo adelante.

Las cámaras PTZ no disponen de un movimiento horizontal continuo de 360° debido a la existencia de un tope mecánico. Esto significa que la cámara no puede seguir a una persona que esté andando de forma continua en un círculo completo alrededor del dispositivo. Además, no están diseñadas para la operación automática continua o las llamadas rondas de vigilancia, en las que la cámara se mueve automáticamente de una posición predefinida a la siguiente (Axis, 2016).

Las cámaras de red domo PTZ pueden cubrir una amplia área al permitir una mayor flexibilidad en las funciones de movimiento horizontal, vertical y zoom. Asimismo, permiten un movimiento horizontal continuo de 360° y un movimiento vertical de normalmente 180°. Debido a su diseño, montaje y dificultad de identificación del ángulo de visión de la cámara (el cristal de las cubiertas de la cúpula puede ser transparente o ahumado), las cámaras de red domo PTZ resultan idóneas para su uso en instalaciones discretas. También proporcionan solidez mecánica para operación continua en el modo ronda de vigilancia, en el que la cámara se mueve automáticamente de una posición predefinida a la siguiente de forma predeterminada o aleatoriamente. Normalmente, pueden configurarse y activarse hasta 20 rondas de vigilancia durante distintas horas del día (Axis, 2016).

Las cámaras móviles en general están controladas a través de sistemas de control por computadora, donde los usuarios generalmente programan la cámara para que se mueva en un patrón establecido, o la controlan manualmente con una interfaz que usualmente es manipulada por un teclado. Un factor muy importante en su construcción son los sistemas embebidos y motores de paso que son utilizados para el correcto funcionamiento de estos equipos.

1.5 Sistemas embebidos

Con el avance tecnológico en el sector de la electrónica y microelectrónica, el diseño de sistemas embebidos o empotrados (SE en lo adelante) se han convertido en un motor clave para el desarrollo tecnológico. Estos son sistemas de propósito especial, donde el equipo está completamente encapsulado por el dispositivo que controla. A diferencia de un ordenador de uso general, tal como un ordenador personal, un sistema embebido realiza tareas predefinidas, por lo general con requisitos muy específicos (Electricaland, 2016). Dado que el sistema está dedicado a una tarea específica, los ingenieros de diseño pueden optimizarlo, reduciendo el tamaño y el coste del producto. Es identificado también como una combinación de hardware y software de computadora, sumado tal vez a algunas

piezas mecánicas o de otro tipo, diseñado para tener una función específica (Galiana Llinare, 2005). Son usados en muchos dispositivos como un taxímetro, sistemas de control de acceso (como en las universidades), copiadoras, impresoras, sistemas de cámaras de seguridad, un reloj, un reproductor de MP3, un teléfono celular, un *router*, el sistema de control de un automóvil, de un satélite o de una planta nuclear (Barrios Frometa, y otros, 2015). Las principales características que poseen son las siguientes:

- Compatibles con una amplia gama de procesadores y arquitecturas de procesador.
- Sensibles económicamente.
- Tienen restricciones de tiempo real.
- Las implicaciones de un fallo de software son mucho más graves que en los sistemas de escritorio.
- Suelen tener limitaciones en el consumo de energía.
- Tienden a tener menos recursos del hardware que los sistemas de escritorio.
- Suelen almacenar todo su código objeto en la memoria ROM.
- Requieren para su desarrollo de herramientas y técnicas especializadas junto a métodos de diseños simples y eficientes.
- A menudo deben operar bajo condiciones ambientales extremas.

Ilustración 6. Sistema Embebido

Estos sistemas basan su funcionamiento en la implementación de un código desarrollado por programadores con el objetivo que cumpla una tarea específica. Este código insertado en el microcontrolador que usan estos sistemas embebidos es comúnmente llamado *firmware*. Un *firmware* no es más que un pequeño programa fijo, que controla el hardware en un sistema, es generalmente responsable de las operaciones básicas de bajo nivel, sin la cual un dispositivo no sería completamente funcional **(Anton, y otros, 2012)**.

1.5.1 Componentes de los sistemas embebidos

Los principales componentes que podemos encontrar en estos sistemas son (Úbeda

Miñarro, 2009):

- **Microprocesador**: Es el encargado de realizar las operaciones de cálculo principales del sistema. Ejecuta código para realizar una determinada tarea y dirige el funcionamiento de los demás elementos que le rodean, a modo de director de una orquesta.
- **Memoria**: En ella se encuentra almacenado el código de los programas que el sistema puede ejecutar, así como los datos. Su característica principal es que debe tener un acceso de lectura y escritura lo más rápido posible para que el microprocesador no pierda tiempo en tareas que sean de cálculo. Al ser volátil el sistema requiere de un soporte donde se almacenen los datos incluso sin disponer de alimentación o energía.
- **Caché**: Memoria más rápida que la principal en la que se almacenan los datos y el código accedido últimamente. Dado que el sistema realiza microtareas, muchas veces repetitivas, la caché hace ahorrar tiempo ya que no hará falta ir a memoria principal si el dato o la instrucción ya se encuentra en la caché. Dado su alto precio tiene un tamaño muy inferior (8 – 512 KB) con respecto a la principal (8 – 256 MB).
- **Disco duro**: En él la información no es volátil y además puede conseguir capacidades muy elevadas. A diferencia de la memoria que es de estado sólido éste suele ser magnético. Pero su excesivo tamaño a veces lo hace inviable para los sistemas embebidos, con lo que se requieren soluciones como discos de estado sólido. Existen en el mercado varias soluciones de esta clase (*DiskOnChip*, *CompactFlash*, IDE *Flash Drive*, etc.) con capacidades suficientes para la mayoría de sistemas embebidos (desde 2 hasta más de 1 GB).
- **Disco flexible**: Su función es la de un disco duro, pero con discos con capacidades mucho más pequeñas. Siempre se encuentra en un ordenador personal estándar pero no así en un sistema embebido.
- **BIOS-ROM**: BIOS (*Basic Input & Output System*, sistema básico de entrada y salida) es código que es necesario para inicializar el ordenador y para poner en comunicación los distintos elementos de la placa madre. La ROM (*Read Only Memory*, memoria de sólo lectura no volátil) es un chip donde se encuentra el código del BIOS.
- **CMOS-RAM**: Es un chip de memoria de lectura y escritura alimentado con una pila donde se almacena el tipo y ubicación de los dispositivos conectados a la placa

madre (disco duro, puertos de entrada y salida, etc.). Además, contiene un reloj en permanente funcionamiento que ofrece al sistema la fecha y la hora.

- **Chip Set**: Chip que se encarga de controlar las interrupciones dirigidas al microprocesador, el acceso directo a memoria (DMA) y al bus ISA, además de ofrecer temporizadores, etc. Es frecuente encontrar la CMOS-RAM y el reloj de tiempo real en el interior del Chip.
- **Entradas y salidas** al sistema necesarias para comunicarse con el mundo exterior.

Uno de los avances que actualmente ha influido en el mundo de los sistemas embebidos ha sido el empleo de estos componentes descritos anteriormente en un solo chip, estos chips son llamados microcontroladores.

1.6 Microcontroladores

Un microcontrolador es un circuito integrado que en su interior contiene una unidad central de procesamiento (CPU), unidades (de memoria RAM y ROM), pines de entrada y salida. Estas partes están interconectadas dentro del microcontrolador, y en conjunto forman lo que se le conoce como microcomputadora. Se puede decir con toda propiedad que un microcontrolador es una microcomputadora completa encapsulada en un circuito integrado (Estudio, 2006).

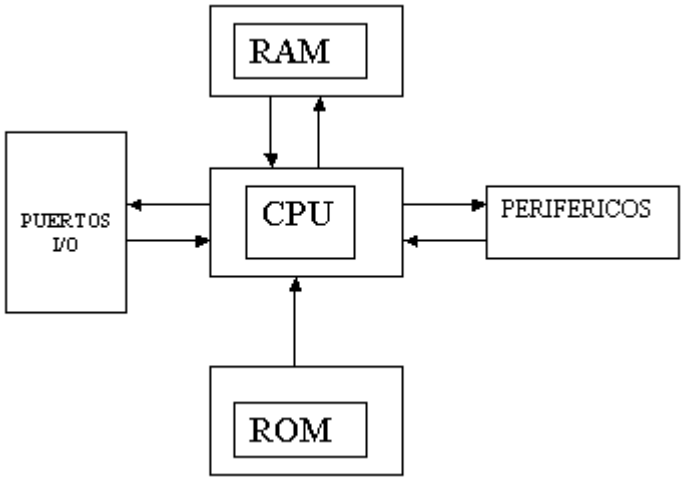

Ilustración 7. Componentes de un microcontrolador

El propósito fundamental de los microcontroladores es el de leer y ejecutar programas implementados por desarrolladores por lo que tienen la característica de ser programables, es por esto que la programación es una actividad básica e indispensable cuando se diseñan circuitos y sistemas que los incluyan. El carácter programable de los

microcontroladores simplifica el diseño de circuitos electrónicos. Permiten modularidad y flexibilidad, ya que un mismo circuito se puede utilizar para que realice diferentes funciones con solo cambiar el programa del microcontrolador.

1.6.1 Diferencia entre microcontrolador y microprocesador

Microprocesador: Procesador implementado en un solo circuito integrado (registros, unidad de control, ALU, unidad de interfaz de bus). Se le conoce también como unidad central de procesamiento (CPU) (Ibarra Esquer, 2012).

Microcontrolador: Microprocesador, memoria y dispositivos de entrada/salida (E/S) incluidos en un solo circuito integrado. Por lo general se utiliza en sistemas empotrados (Ibarra Esquer, 2012).

Existen muchas familias fabricantes de microcontroladores, entre las más comunes están: Atmel (AVR), Hitachi (H8), Intel de 8 bits (8XC42, MCS51, 8xC251) o Intel de 16 bits (MCS96, MXS296), *National Semiconductor* (COP8), Microchip (PIC), Motorola de 8 bits (68HC05, 68HC08, 68HC11) o de 16 bits (68HC12, 68HC16) o de 32 bits (683xx). De todos estos fabricantes de microcontroladores se destacan dos familias de microcontroladores: la familia AVR producida por ATmel y la familia PIC por Microchip, cuya popularidad es alta entre diseñadores de sistemas embebidos que requieren un rendimiento alto y bajo costo, y eligen uno u otro ya sea por su nivel de integración, por su arquitectura, la disponibilidad de recursos o su lenguaje de programación.

1.6.2 Comparación de microcontroladores AVR y PIC

Los microcontroladores PIC (*Peripheral Interface Controller*) está diseñado con la capacidad de procesar palabras de 8 bits de datos, poseen una arquitectura Computador con Conjunto de Instrucciones Reducidas (RISC, por sus siglas en ingles). Entre las características más importantes que estos poseen, se encuentran (MORALES BEJARANO , et al., 2012):

- Temporizadores de 8 y 16 bits.
- Conversores analógico-digital de 8, 10 y12 bits.
- Módulos que permiten el manejo de protocolos ethernet.
- Capacidad para utilizar la interfaz USB.
- Periféricos para el control de motores.
- Memorias flash con capacidad desde 256 bytes hasta 256 kbytes.

La tecnología de los microcontroladores AVR es CMOS (*Complementary Metal Oxide Semiconductor*) de 8 bits con bajo consumo, basados en arquitectura RISC. Esto implica que poseen un reducido juego de instrucciones, la mayor parte de las cuales se ejecutan en un único ciclo de reloj, consiguiendo una capacidad de procesamiento cercana a 1MIPS por MHz, permitiendo al diseñador del sistema optimizar el consumo gracias a la gran velocidad de procesamiento. La familia AVR utiliza el concepto de arquitectura Harvard con buses y memorias, separados para los datos y el programa, permitiendo que las instrucciones sencillas sean ejecutadas en un ciclo de reloj. En la actualidad existe una gran familia de microcontroladores AVR entre los que se encuentran ATMega8, ATMega16, ATMega328. Las características más sobresalientes de los microcontroladores AVR son:

- Temporizadores de 8 y 16 bits.
- Conversores A/D de 16 bits.
- Poseen un reloj interno capaz de generar señales de 1MHz, 4MHz y 8MHz sin hardware adicional
- Manejo de interfaz USB.

Para el dar cumplimiento a los objetivos trazados se hará uso de microcontroladores pertenecientes a la familia de Atmel (AVR), debido a que su diseño difiere de los demás microcontroladores, ya que estos poseen mayor cantidad de registros (32) y un conjunto ortogonal de instrucciones. Esto hace que la arquitectura AVR sea más fácil de programar a nivel de lenguaje ensamblador y que sea fácil de optimizar con un compilador. Además, presentan una mayor inmunidad al ruido y es por lo general más económico que un microcontrolador PIC. Además, son muy usados en la actualidad en plataformas electrónicas destinadas a la realización de sistemas embebidos.

1.7 Plataformas electrónicas usadas en sistemas embebidos

Como principales plataformas empleadas en la actualidad para el desarrollo de sistemas embebidos se identifican las Raspberry Pi y la placa arduino. Aunque están destinadas a resolver problemas similares, estas son muy diferentes debido a que una Raspberry Pi es identificada como una computadora completamente funcional, mientras que arduino es identificado como un microcontrolador, el cual es sólo un componente de una computadora.

1.7.1 Comparación de las principales plataformas

A continuación, una comparación entre hardware y software de dichas plataformas (Hacedores, 2016).

	Arduino	**Raspberry Pi**
Precio en dólares	$30	$35
Tamaño	7.6 x 1.9 x 6.4 cm	8.6cm x 5.4cm x 1.7cm
Memoria	0.002MB	512MB
Velocidad de reloj	16 MHz	700 MHz
On Board Network	Ninguna	10/100 wired Ethernet RJ45
Multitarea	No	Si
Voltaje de entrada	7 a 12 V	5 V
Memoria Flash	32KB	Tarjeta SD (2 a 16G)
Puertos USB	Uno	Dos
Sistema operativo	Ninguno	Distribuciones de Linux
Entorno de desarrollo integrado (IDE)	Arduino	Scratch, IDLE, cualquiera con soporte Linux

Tabla 1. Comparación entre Arduino y Raspberry PI

Como resumen de esta comparación podemos expresar que ambas son placas pequeñas y baratas, pero la Raspberry Pi es muy superior en todos los demás aspectos, pero eso pasa sólo cuando se trata de aplicaciones de software. La simplicidad de Arduino hace que éste sea una apuesta mucho mejor para proyectos de hardware ya que este proporciona algunas ventajas para estos proyectos como son (Hacedores, 2016) :

- Arduino es más flexible que la Pi debido a que ofrece una capacidad analógica y en tiempo real. Esta flexibilidad le permite trabajar con casi cualquier tipo de sensor o actuador, mientras que la Pi para la lectura de los sensores analógicos requiere la asistencia de un hardware adicional.
- El IDE Arduino es mucho más fácil de usar que el entorno de desarrollo que emplea la Pi. Por ejemplo, si se desea escribir un programa para hacer parpadear un LED con Raspberry Pi, se necesita instalar un sistema operativo y algunas librerías de código, mientras que en Arduino se logra obtener una luz LED parpadeando con tan sólo ocho líneas de código.

Por tanto, se ha decidido usar la plataforma Arduino para el cumplimiento de los objetivos propuestos debido a que se ajusta más para el desarrollo de solución que se propone.

1.7.2 Plataforma Arduino

La placa Arduino es una plataforma de prototipos electrónica de código abierto (open-source), basada en *hardware* y *software* flexibles y fáciles de usar. Está pensado para artistas, diseñadores, como medio para aprendizaje y para cualquier interesado en crear objetos o entornos interactivos. (TOME, 2011)

El *hardware* de dicha plataforma consiste en una placa con un microcontrolador reprogramable y varios puertos de E/S, tanto digitales como analógicos, así como las salidas de modulación por ancho de pulso (PWM, por sus siglas en ingles) y de comunicaciones. Puede tomar información del entorno a través de sus entradas analógicas y digitales, y permite el control sobre luces, motores y otros actuadores. Existen muchos otros microcontroladores y plataformas micro-controladoras disponibles para la computación física, así como: Parallax, Basic, Stamp, Netmedia's, BX-24, Phidgets, MIT's Handyboard, y muchas otras ofertas de funcionalidad similar. Todas estas herramientas toman los desordenados detalles de la programación del microcontrolador y la encierran en un paquete fácil de usar. Arduino también simplifica el proceso de trabajo con microcontroladores, pero ofrece algunas ventajas para profesores, estudiantes y aficionados interesados sobre otros sistemas:

- Barato: Las placas Arduino son relativamente baratas comparadas con otras plataformas micro-controladoras.
- Multiplataforma: El software de Arduino se puede instalar y ejecutar en varios sistemas operativos, ya sea, Windows, Macintosh OSX y GNU/Linux. La mayoría de los sistemas microcontroladores están limitados a Windows.
- Entorno de programación simple y clara: El entorno de programación de Arduino es fácil de usar para principiantes, pero suficientemente flexible para que usuarios avanzados puedan aprovecharlo también.
- Código abierto y *software* extensible: El IDE Arduino está publicado como herramientas de código abierto, disponible para el desarrollo de extensiones por programadores experimentados. El lenguaje puede ser expandido mediante librerías C++, y la gente que quiera entender los detalles técnicos pueden hacer el salto desde Arduino a la programación en lenguaje AVR, C en el cual está basado.

De forma similar, puedes añadir código AVR-C directamente en tus programas Arduino si desea.

- Código abierto y *hardware* extensible: El Arduino está basado en microcontroladores ATMEGA8 y ATMEGA168 de Atmel. Los planos para los módulos están publicados bajo licencia *Creative Commons*, por lo que diseñadores experimentados de circuitos pueden hacer su propia versión del módulo, extendiéndolo y mejorándolo. Incluso usuarios relativamente inexpertos pueden construir la versión de la placa del módulo para entender cómo funciona y ahorrar dinero.

1.7.3 Tipos de placas arduino

Algunas de placas más utilizadas de la familia de arduino se describen a continuación (Torrente Artero, 2013):

Arduino UNO: El microcontrolador que lleva esta placa es el modelo ATmega328P de la marca Atmel. La "P" del final significa que este chip incorpora la tecnología "*Picopower*" (propietaria de Atmel), la cual permite un consumo eléctrico sensiblemente menor comparándolo con el modelo equivalente sin "*Picopower*", el Atmega328 (sin la "P"). El voltaje de funcionamiento de la placa Arduino (incluyendo el microcontrolador y el resto de componentes) es de 5 V. Dispone de 14 pines de E/S digitales. Es aquí donde conectaremos nuestros sensores para que la placa pueda recibir datos del entorno, y también donde conectaremos los actuadores para que la placa pueda enviarles las órdenes pertinentes, además de poder conectar cualquier otro componente que necesite comunicarse con la placa de alguna manera. Además, dispone de 6 entradas analógicas.

Arduino Mega2560: Placa basada en el microcontrolador ATmega2560. Esta placa posee 54 pines de E/S digitales (de los cuales 14 pueden ser usados como salidas analógicas PWM), 16 entradas analógicas y 4 receptores/transmisores serie TTL-UART. Consta de una memoria Flash de 256 Kilobytes (de los cuales 8 están reservados para el *bootloader* (gestor de arranque)), una memoria SRAM de 8 KB y una EEPROM de 4 KB. Su voltaje de trabajo es igual al del modelo UNO (5V).

Arduino Ethernet: Al igual que el modelo UNO, la placa Ethernet está basada en el microcontrolador ATmega328P, posee la misma cantidad de memoria Flash, SRAM y EPPROM, también el mismo número de pines de E/S digitales y de entradas analógicas. El resto de características también es muy similar al modelo UNO. La

mayor diferencia que existe con la placa Arduino UNO es que incorpora un zócalo de tipo RJ-45 para poder conectarse mediante el cable adecuado (cable de par trenzado de categoría 5 o 6) a una red de tipo Ethernet y un zócalo para insertar una tarjeta micro-SD, la cual puede ser usada para guardar diferentes tipos de ficheros y ofrecerlos a través de la red. La placa Arduino Ethernet permite transferir datos entre ella misma (los cuales pueden ser obtenidos de algún sensor, por ejemplo) y cualquier otro dispositivo conectado a su misma red LAN (normalmente, un computador que los recopila y guarda), o viceversa: transferir datos entre un dispositivo conectado a la LAN (normalmente, un computador ejecutando algún tipo de software de control) y ella misma (la cual puede estar conectada a algún actuador controlado remotamente por ese computador). También se puede lograr, gracias al establecimiento del enrutamiento de paquetes adecuado, comunicar nuestra placa Ethernet con cualquier dispositivo conectado a cualquier red del mundo fuera de nuestra LAN privada (incluyendo “Internet”), con lo que sus posibilidades de uso se disparan.

Arduino Pro: Esta placa viene en dos “versiones”: ambas contienen un microcontrolador Atmega328P SMD, pero una funciona con 3,3 V y a 8 MHz y la otra funciona con 5 V y a 16 MHz. Dispone de 14 agujeros pensados para funcionar como pines de E/S digital (6 de los cuales pueden ser usados como salida PWM), 6 agujeros para entradas analógicas, agujeros para montar un conector de alimentación de 2,1 mm, un conector para una batería externa, un interruptor de corriente, un botón de reinicio y los pines necesarios para conectar un adaptador o cable USB-Serial y así poder programarla (y también alimentarla) directamente vía USB.

Arduino Lilypad: La placa Arduino LilyPad está diseñada para ser cosida a material textil. Permite además conectarle (mediante hilos conductores) fuentes de alimentación, sensores y actuadores de forma que se puedan “llevar encima”, haciendo posible la creación de vestidos y ropa “inteligente”. Además, se puede lavar. Esta placa incorpora el microcontrolador ATmega328V (una versión de bajo consumo del Atmega328P), el cual se programa acoplando a la placa un adaptador o cable USB-Serie.

Arduino Nano: La característica más destacable de esta placa es que a pesar de su tamaño (0,73 pulgadas de anchura por 1,70 de longitud), sigue ofreciendo el mismo número de E/S digitales y analógicas que la Arduino UNO, como también la misma funcionalidad que esta. La consecuencia más evidente de su reducido tamaño es que carece del conector de alimentación de 2,1 mm (aunque puede seguir siendo

alimentada por una fuente externa mediante el pin “Vin” o “5 V”) e incorpora un conector USB mini-B en vez del conector USB tipo B.

Arduino Mini: Esta placa es muy parecida a la placa Arduino Nano: está basada igualmente en el microcontrolador ATmega328P SMD funcionando a 16MHz, tiene 14 pines de E/S digitales (6 de los cuales pueden funcionar como salidas PWM) y 8 entradas analógicas. La diferencia más importante con la placa Arduino Nano está en que la Arduino Mini (para ahorrar aún más espacio físico y así conseguir un tamaño realmente mínimo de 0,7 pulgadas de ancho por 1,3 de largo) no incorpora ningún chip conversor USB-Serie. Debido a ello, para su programación se necesita utilizar un adaptador USB-Serial externo.

Arduino Leonardo: La gran novedad de esta placa es que el microcontrolador que incorpora es el ATmega32U4, el cual tiene todas las funcionalidades que ofrece el Atmega328P, pero incorpora además 0,5 kilobytes más de memoria SRAM y sobretodo, soporta comunicaciones USB directamente (por tanto, no necesita de ningún chip suplementario como el ATmega16U2 o el FTDI). Otras diferencias con la placa UNO es que la placa Leonardo incorpora un pin digital más para ser usado como salida PWM y 6 entradas analógicas extras, las cuales están situadas físicamente en los pines digitales marcados con un puntito en el exterior de la placa. El hecho de que la placa Leonardo solo incluya un microcontrolador tanto para ejecutar los programas como para comunicarse directamente a través de USB con el computador permite que esta placa pueda simular fácilmente (si se programa convenientemente) ser un teclado o un ratón USB conectados a dicho computador. Técnicamente, cuando la placa Leonardo se conecte con un cable USB al computador, este detectará dos “puertos de conexión diferentes: un puerto USB estándar listo para usar la placa Leonardo como un periférico USB más (lo típico sería un teclado o un ratón, tal como hemos dicho) y otro puerto diferente, similar al generado cuando se conecta la Arduino UNO, usable de la forma “tradicional”, para la programación y comunicación con la placa a través del entorno de programación.

Arduino Micro: Esta placa ofrece las mismas funcionalidades que la Arduino Leonardo (tiene por ejemplo el mismo microcontrolador ATmega32U4 a 16MHz, los mismos 32KB de memoria Flash y 2,5 KB de memoria SRAM, el mismo *bootloader*, el mismo voltaje de trabajo 5V) pero con tamaño realmente mínimo: 48 x 18 mm, ideal para ser ubicado sobre una *breadboard* sin ocupar apenas espacio. Al igual que el modelo Leonardo, se puede programar a través de una conexión USB (dispone de un

zócalo mini-B para ello), pudiendo funcionar además como teclado o ratón simulado. Las características del "*auto-reset*" de la placa Leonardo también son aplicables a la Micro. Puede ser alimentada a través del cable USB o bien mediante una fuente de alimentación externa.

El modelo concreto que se ha utilizado es el Arduino UNO debido a que esta plataforma es la disponible para las condiciones de desarrollo actual en el centro CEDIN de la UCI. Las especificaciones de este *hardware* son:

Microcontroller	**ATmega328**
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) using 0.5 for the boot sector
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Tabla 2. Especificaciones del Arduino UNO

Con el empleo de estas plataformas en la actualidad se ha permitido ejercer el control sobre diversos componentes como sensores, luces (LED), actuadores y motores eléctricos (servomotores, motores DC, motores de paso) a fin de crear proyectos que tributen al desarrollo de la sociedad.

1.8 Motores paso a paso

Los motores paso a paso o también conocidos como motores de paso, han estado con nosotros desde los años 60. Estos son actuadores electromagnéticos rotatorios que convierten mecánicamente entradas de pulsos digitales a movimiento rotatorio incremental (Canto, 2013). Algunas de las principales características que poseen estos motores son las siguientes:

- La rotación no solo tiene una relación directa al número de pulsos de entrada, sino que la velocidad está relacionada a la frecuencia de los pulsos.
- Motores sin escobillas que digitalmente controlados, rotan un número específico de grados paso a paso.
- El número de grados (resolución) puede ser entre 0.72 a 90 grados. Los “steppers” (o motores paso a paso) de propósito general tiene una resolución de entre 15 y 30 grados por paso.
- Son de alta precisión y bajo torque.

En la actualidad existen dos tipos básicos de motores paso a paso, los bipolares que se componen de dos bobinas y los unipolares que tienen cuatro bobinas. Externamente se diferencian entre sí por el número de cables. Los bipolares solo tienen cuatro conexiones dos para cada bobina y los unipolares que normalmente presentan seis cables, uno para cada bobina y otro para alimentación de cada par de éstas, aunque en algunos casos podemos encontrar motores unipolares con cinco cables, básicamente es lo mismo, solo que el cable de alimentación es común para los dos pares de bobinas.

Motores Unipolares: En este tipo de motores, todas las bobinas del estator están conectadas en serie formando cuatro grupos. Esta a su vez, se conectan dos a dos, también en serie, y se montan sobre dos estatores diferentes, del motor paso a paso salen dos grupos de tres cables, uno de los cuales es común a dos bobinados. Los seis terminales que parten del motor, deben ser conectados al circuito de control, el cual, se comporta como cuatro conmutadores electrónicos que, al ser activados o desactivados, producen la alimentación de los cuatro grupos de bobinas con que está formado el estator. Si generamos una secuencia adecuada de funcionamiento de estos interruptores, se pueden producir saltos de un paso en el número y sentido que se desee (Apitz, Javier Colmenares, 2007).

Motores Bipolares: En este tipo de motores las bobinas del estator se conectan en serie formando solamente dos grupos, que se montan sobre dos estatores (Apitz, Javier Colmenares, 2007). De este motor salen cuatro hilos que se conectan, al circuito de control, que realiza la función de cuatro interruptores electrónicos dobles, que nos permiten variar la polaridad de la alimentación de las bobinas. Con la activación y

desactivación adecuada de dichos interruptores dobles, podemos obtener las secuencias adecuadas para que el motor pueda girar en un sentido o en otro.

Para dar cumplimiento al objetivo de dicha investigación se emplearán motores de paso unipolares debido a que son los que trae ya incorporado la cámara de vigilancia, proporcionando mayor control y precisión sobre la cámara.

1.9 Dispositivo móvil inteligente

Las tecnologías móviles tienen mucho tiempo entre nosotros simplificando nuestras actividades cotidianas facilitando nuestros trabajos, estudios o vida normal con sus innumerables aplicaciones disponibles que se han ido incrementando con el tiempo. Actualmente, las tecnologías móviles han cubierto la mayoría de las áreas de servicio de comunicaciones y entretenimiento enfocando sus aplicaciones a generar un mercado cautivo de estas, a los miles de usuarios que día a día adoptan el uso de servicios, tal como lo es, telefonía móvil, envío de mensajes de texto y multimedia; y en los últimos años la actualización de la información de las redes sociales.

"(…) Un Smartphone (teléfono inteligente en español) es un dispositivo electrónico que funciona como un teléfono móvil con características similares a las de un ordenador personal. Casi todos los teléfonos inteligentes son móviles que soportan completamente un cliente de correo electrónico con la funcionalidad completa de un organizador personal. Una característica importante de casi todos los teléfonos inteligentes es que permiten la instalación de programas para incrementar el procesamiento de datos y la conectividad. Estas aplicaciones pueden ser desarrolladas por el fabricante del dispositivo, por el operador o por un tercero. El término "Inteligente" hace referencia a cualquier interfaz, como un teclado QWERTY en miniatura, una pantalla táctil, o simplemente el acceso a Internet y al correo electrónico de una compañía, pagando o personal, gratuito" (39A, 2012).

Múltiples son las empresas productoras que desarrollan estos dispositivos. Entre las principales se encuentran Samsung, Apple, Sony, LG, Motorola, Huawei ya que estas se mantienen entre los primeros en el ranking de dispositivos móviles emitidos en los últimos años.

1.9.1 Sistemas Operativos usados por los dispositivos inteligentes

Un sistema operativo (SO) es un grupo de programas de proceso con las rutinas de control necesarias para mantener continuamente operativos dichos programas. (la Red Martínez,

2001). Para un mejor entendimiento un SO es un conjunto de programas especialmente hechos para la ejecución de varias tareas, en las que sirve de intermediario entre el usuario y la computadora.

En breve una caracterización de los sistemas operativos más usados en la actualidad.

Android:

- Es de código abierto (se puede modificar), es gratis y está basado en Linux.
- Se adapta a las diferentes resoluciones de pantalla.
- Soporte de HTML, HTML5, Adobe Flash Player, etc.
- Un gran catálogo de aplicaciones para descargar, gratuitas y de pago.
- Multitarea real de aplicaciones.
- Gran cantidad de formas diferentes de personalizar el escritorio de nuestro Smartphone.
- Se pueden buscar aplicaciones que se necesiten e instalarlas directamente con el PC puesto que todo se sincroniza automáticamente en el teléfono sin necesidad de conexión de cables.
- Permite el control del teléfono móvil desde el ordenador de forma muy sencilla.

iOS:

- Es un SO cerrado, es decir no se puede modificar.
- Tiene un sistema de monitorización del consumo de batería que podría ayudar a gestionarla de forma mucho más eficiente.
- Permite que podamos instalar un teclado de terceros.
- Presenta funciones que incluyen atajos para mandar fotos, videos, notas de voz, compartir tu ubicación, mejor gestión de conversiones en grupo y una opción para silencio.
- Otra característica divertida de iOS es la posibilidad que Apple le ha dado a Siri de identificar canciones.
- Sensación de velocidad durante su uso, mediante algunos trucos de programación.
- Reciben de manera constante actualizaciones.
- Apple no da licencia del *software* iOS a terceros, por lo que tan solo los iPhone disponen de este SO.

Symbian:

- Posee un eficiente uso de todos los recursos de la máquina (especialmente de la batería, la memoria RAM y la ROM).

- La paginación bajo demanda permite un mejor aprovechamiento de la memoria RAM de los dispositivos ya que solo se carga en memoria la "pagina" que se va a ejecutar.
- El sistema posee componentes que permiten el diseño de aplicaciones multiplataforma, o sea, diferentes tamaños de pantalla, color, resolución, teclados, etc.

Windows Phone:

- Está diseñado para ser similar a las versiones de escritorio de Windows estéticamente y tiene la posibilidad de utilizar importantes herramientas pertenecientes a las suites Office Mobile, Outlook Mobile e Internet Explorer.
- Skype estará completamente integrado en Windows Phone 8, música en *streaming* Pandora, podremos crear "habitaciones" de usuarios en el PeopleHub, en las que podremos crear grupos para chats privados, compartir calendarios y notas públicas.
- Lo malo de este sistema operativo es que hay pocas aplicaciones (apps) para este SO.

BlackBerry OS:

- El sistema permite multitarea.
- Uso profesional, gracias a sus herramientas para correo electrónico y agenda, teclado QWERTY.

En la presente investigación se hará uso del sistema operativo Android debido a su flexibilidad y facilidad para utilizar los servicios integrados de *Google*, además de ser libre, gratuito y multiplataforma.

1.10 Tecnologías, metodologías y herramientas usadas para el desarrollo de la propuesta de solución

En el presente apartado se realiza un estudio y selección de la metodología, tecnologías y herramientas que contribuyan al cumplimiento del objetivo general de la investigación.

1.10.1 Tecnología de comunicación

Hoy en día, la conexión entre dispositivos para el intercambio de datos se puede establecer mediante cables o de manera inalámbrica. La comunicación entre el *Smartphone* y la cámara de seguridad se establecerá de manera inalámbrica, ya que ambos dispositivos disponen de las tecnologías *Wifi* y *Bluetooth*. En el caso de la

plataforma Arduino es necesario de un módulo para disponer de dichas tecnologías mientras que los teléfonos inteligentes ya los traen consigo. La tecnología seleccionada fue la de *Bluetooth* debido a petición del cliente. A continuación, se describe en que consiste esta tecnología.

Bluetooth es un estándar global de comunicación inalámbrica establecido por la IEEE 802.15.1, donde se pueden realizar conexiones de red inalámbricas teniendo la posibilidad de transmitir voz, datos, imagen, multimedia entre diferentes dispositivos utilizando la tecnología de radio frecuencia de corto alcance (Mata Ramírez, 2013). Permite la comunicación, incluso a través de obstáculos y a distancias de hasta unos 10 metros. Esto significa que, por ejemplo, puedes oír tus mp3 desde tu comedor, cocina, cuarto de baño, etc. También sirve para crear una conexión a internet inalámbrica desde tu portátil usando tu teléfono móvil. Un caso aún más práctico es el poder sincronizar libretas de direcciones, calendarios, en tu Asistente Digital Personal (PDA, por sus siglas en ingles), teléfono móvil, ordenador de sobremesa, todo al mismo tiempo.

Se utilizan protocolos de alto nivel en esta tecnología, como el Protocolo de Descubrimiento de Servicios (SDP) que es un protocolo que permite detectar otros dispositivos en el rango de comunicación permitido. También se encuentra el Protocolo de Comunicación por Radio Frecuencia (RFCOMN) que permite emular la conexión de un puerto serial y Protocolo de Control de Telefonía (TCS) que es un protocolo de control de telefonía, todos interactúan entre sí para tener comunicación con el controlador de banda base, mediante el Protocolo de Adaptación y de Control de enlace a nivel Lógico (L2CAP). Este último es el encargado de la segmentación y re-ensamble de los paquetes y a su vez los envía a través de la conexión *Bluetooth*. Los objetivos principales que persigue esta tecnología son los siguientes:

- Permitir la comunicación sencilla entre dispositivos fijos y móviles.
- Evitar la dependencia de cables que permitan la comunicación.
- Permitir la creación de pequeñas redes de forma inalámbrica.

Existen varios módulos de *Bluetooth* que pueden ser conectados al Arduino para que este permita la comunicación con otros dispositivos, algunos de estos son:

El módulo **Bluetooth HC-06** es un dispositivo muy fácil de obtener, económico y sencillo de utilizar. Una de las principales ventajas, además de su pequeño tamaño y sus buenas características de transmisión y recepción, es el bajo consumo de corriente que posee

tanto en funcionamiento, como en modo de espera. Otra característica interesante de este módulo es que una vez que ha realizado un enlace con otro dispositivo es capaz de recordarlo en su memoria y no solicita validación alguna ("*1234*" por defecto), pero si se activa el pin 26 (*KEY*) hacia la tensión de alimentación, esta información se elimina y el módulo HC-06 solicitará nuevamente la validación del enlace. Otro detalle particular es que su tensión de alimentación de 3,3Volts y su bajo consumo (8mA en transmisión/recepción activa) lo transforman en un dispositivo ideal para trabajar con microcontroladores de la misma tensión de alimentación (Mario, 2015-2016).

El módulo de **Bluetooth HC-05** ofrece una mejora sobre el HC-06, con respecto al precio y características, ya que es un módulo Maestro-Esclavo, esto quiere decir que además de recibir conexiones desde una PC o Tablet, también es capaz de generar conexiones hacia otros dispositivos *Bluetooth*. Esto nos permite conectar dos módulos de *Bluetooth* y formar una conexión punto a punto, para transmitir datos entre dos microcontroladores o dispositivos (Durán Rocha, 2015).

Aunque el HC-05 ofrece algunas mejoras sobre el HC-06 en cuanto a costo y características. Se empleará el HC-06 debido a que su uso se ajusta mejor para el cumplimiento de los objetivos de la investigación, tomando en cuenta las características antes mencionadas.

1.10.2 Lenguaje de modelado

UML: es un lenguaje para visualizar, especificar, construir y documentar los elementos que componen un sistema basado en la programación orientada a objetos. Se emplea para modelar algunos artefactos como: el diagrama de estados y las tarjetas CRC (Barrios Frometa, y otros, 2015). También es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad.

1.10.3 Herramientas

Visual Paradigm v8.0: Es una herramienta CASE que utiliza para el modelado el lenguaje Unified Model Language (UML, por sus siglas en inglés). Es libre y multiplataforma, además se caracteriza por su robustez, usabilidad y portabilidad. Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Se integra fácilmente con varios entornos de desarrollo y permite la generación semiautomática de código a partir de los diagramas construidos. Esta

herramienta permite realizar ingeniería tanto directa como inversa, a partir de un modelo relacional en SQL Server, MySQL, PostgreSQL, es capaz de desplegar todas las clases asociadas a las tablas (siguiendo el patrón de diseño Una Clase-Una Tabla). Soporta múltiples usuarios trabajando sobre el mismo proyecto, permite control de versiones y genera de forma automática documentación en formatos web y PDF (VisualParadigm.com, 2016).

1.10.4 Entornos de desarrollo

IDE Android Studio: Es un entorno de desarrollo integrado (IDE), basado en IntelliJ IDEA de la compañía *JetBrains*, es el IDE recomendado por *Google* hasta la fecha ya que proporciona varias mejoras con respecto al *plugin Android Developer Tools* (ADT) para Eclipse y utiliza una licencia de *software* libre Apache 2.0. Persigue como objetivo de crear un entorno dedicado en exclusiva a la programación de aplicaciones para dispositivos Android, proporcionando a *Google* un mayor control sobre el proceso de producción. Está programado en Java y es multiplataforma (Android, 2005).

Ventajas de usar Android Studio:

- Es entorno recomendado para el desarrollo de aplicaciones en Android, al tratarse de un IDE oficial de *Google* en colaboración con JetBrains (compañía de desarrollo software especializada en diseño de IDEs).
- Permite la creación de nuevos módulos dentro de un mismo proyecto, sin necesidad de estar cambiando de espacio de trabajo para el manejo de proyectos, algo habitual en Eclipse.
- Con la simple descarga de Android Studio se disponen de todas las herramientas necesarias para el desarrollo de aplicaciones para la plataforma Android.
- Su nueva forma de construir los paquetes apk (formato de archivo utilizado para la instalación de software en el sistema operativo Android), mediante el uso de Gradle, proporciona una serie de ventajas más acorde a un proyecto Java:
    - Facilita la distribución de código, y por lo tanto el trabajo en equipo.
    - Reutilización de código y recursos.
    - Permite compilar desde línea de comandos, para aquellas situaciones en las que no esté disponible un entorno de desarrollo.
    - Mayor facilidad para la creación de diferentes versiones de la misma aplicación, que proporciona numerosas ventajas como puede ser la

creación de una versión de pago y otra gratuita, o por ejemplo diferentes dispositivos o almacén de datos.

Para la conformación de aplicaciones Android existen diversos entornos de desarrollo, entre los más utilizados por los programadores se encuentran Eclipse, NetBeans, IntelliJ IDEA y Android Studio. Se ha seleccionado este último debido a todas las ventajas que ofrece ya antes mencionadas y además por ser el IDE recomendado por *Google* para la creación de aplicaciones Android.

IDE Arduino es una aplicación multiplataforma escrita en Java, basado en un entorno de desarrollo que utiliza varios lenguajes de programación. El entorno de desarrollo Arduino está basado en C por lo cual soporta todas las funciones de este estándar y algunas de C++. Es el IDE seleccionado debido a que es un entorno desarrollado para trabajar específicamente con la plataforma Arduino, permitiendo la conformación del firmware a usar en la propuesta de solución y además de su incorporación en el microcontrolador (Arduino, 2016).

1.10.5 Lenguajes de programación

La programación es un gran recurso que nos permite crear diversas secuencias de pasos lógicos que van a satisfacer nuestras necesidades y las de nuestros sistemas, su propósito es crear programas que exhiban un comportamiento deseado. Programar es todo un arte que requiere de una gran habilidad lógica y concentración por parte del programador. Es el proceso de diseñar, escribir, probar, depurar y mantener el código fuente de programas computacionales. El código fuente es escrito en un lenguaje de programación (robotica, 2014).

Un lenguaje de programación es un idioma artificial diseñado para expresar operaciones que pueden ser llevadas a cabo por máquinas como los computadores. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones (robotica, 2014).

Luego de un análisis de los principales conceptos, se han seleccionado los lenguajes de programación Arduino basado en C/C++ para el desarrollo del firmware de control y java para la aplicación de control debido a que son los lenguajes que utiliza los entornos de desarrollo seleccionados. A continuación, características de los lenguajes seleccionados:

Características del lenguaje de C/C++

- Es un lenguaje imperativo y estructurado
    - Permite el uso de subrutinas y estructuras de control
- Es un lenguaje amigable, flexible y muy potente para el programador
    - Combina la flexibilidad de los lenguajes de alto nivel con el control y la funcionalidad que ofrecen lenguajes ensambladores (manipulación de bits, bytes y direcciones)
- Es un lenguaje eficiente
- Es un lenguaje portable
    - Un programa escrito en ANSI C puede ejecutarse en cualquier ordenador con prácticamente ninguna modificación
- Es un lenguaje compilado

Características del lenguaje Java

- La principal característica de Java es la de ser un lenguaje compilado e interpretado.
- Java es un lenguaje orientado a objetos de propósito general.
- Su sintaxis es muy parecida a la de C y C++, pero hasta ahí llega el parecido. Java no es una evolución ni de C ni un C++ mejorado.
- Java define procedimientos para tratar errores.
- Está preparado para la programación concurrente sin necesidad de utilizar ningún tipo de biblioteca.

1.10.6 Metodología de desarrollo

Una metodología de desarrollo de software tiene entre sus objetivos aumentar la calidad del software que se produce (Sánchez, 2015). Entre algunas de las metodologías más usadas en la actividad productiva de la Universidad de Ciencias Informáticas, se encuentran: eXtreme Programming (XP), Proceso Unificado de Rational (RUP), Proceso Unificado Abierto (OPEN UP), Agile Unified Process (AUP), y Agile Unified Process ajustada a la UCI (AUP-UCI), cuyas características veremos a continuación.

eXtreme Programming (XP): es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los

desarrolladores, y propiciando un buen clima de trabajo. Se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Es destinada para proyectos con requisitos imprecisos y muy cambiantes, y además donde existe un alto riesgo técnico (Letelier, et al., 2006).

Proceso Unificado de Rational (RUP): Es una metodología de desarrollo de *software* que intenta integrar los aspectos a tener en cuenta durante todo el ciclo de vida del sistema, con el propósito de abarcar grandes o pequeños proyectos de *software*. Proporciona un acercamiento a la asignación de tareas y responsabilidades en una organización de desarrollo. Su objetivo es asegurar la producción de *software* de alta calidad, que se ajuste a las necesidades de sus usuarios finales. Provee herramientas para los pasos del desarrollo y una vasta documentación de ayuda para sus clientes (Jacobson, y otros, 2000).

RUP cuenta con tres características importantes que la hacen única: está dirigida por casos de uso, centrada en la arquitectura, iterativa e incremental (Martínez, 2003). Otras de sus características más importantes son:

- Propone un flujo de trabajo específicamente para el análisis y diseño del sistema, donde se especifican los requisitos identificados y se describen en términos de diseño sobre cómo se va a implementar el sistema.
- Provee vasta documentación al equipo de trabajo.
- Describe como obtener, organizar y documentar todas las funcionalidades y restricciones del sistema.
- En cada una de sus fases obtiene una serie de artefactos que sirven para comprender mejor tanto el análisis como el diseño del sistema.

Ventajas:

- Coste del riesgo a un solo incremento.
- Reduce el riesgo de no sacar el producto en el calendario previsto.
- Se adapta mejor a las necesidades del cliente.

Es una metodología de desarrollo de *software* que está basada en componentes e interfaces bien definidas y junto con el UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

Proceso Unificado Abierto (OPEN UP): es un proceso unificado ágil y liviano, que aplica un enfoque iterativo e incremental dentro de un ciclo de vida estructurado y contiene un conjunto mínimo de prácticas que ayuda al equipo a ser más efectivo desarrollando *software*. Open Up abraza una filosofía pragmática y ágil de desarrollo, que se enfoca en la naturaleza colaborativa del desarrollo de *software*. Se trata de un proceso con herramientas neutrales y de baja ceremonia que puede extenderse para alcanzar una amplia variedad de tipos de proyectos. Es parte del *framework* de modelo de proceso de Eclipse (Eclipse Process Framework), desarrollado por la fundación Eclipse (Gimson , 2012). Mantiene las características esenciales de RUP, en el cual se incluyen las siguientes características:

- Desarrollo incremental.
- Uso de casos de uso y escenarios.
- Manejo de riesgos.
- Diseño basado en la arquitectura.

Agile Unified Process (AUP): es una metodología fácil de entender para el desarrollo de aplicaciones *software* de negocio, utilizando técnicas ágiles y conceptos aun fieles a las de RUP, por lo tanto, es una versión simplificada RUP. Posee muchas de las técnicas ágiles de la que dispone metodología XP y de las formalidades de RUP, teniendo como filosofía adaptarse a las necesidades del proyecto. También, plantea un ciclo de vida iterativo, que se basa en la ampliación y refinamiento sucesivo del sistema, mediante múltiples iteraciones con retroalimentación cíclica y adaptación como elementos principales que dirigen para converger hacia un sistema adecuado (NÚÑEZ MORI, 2010).

Agile Unified Process ajustada a la Universidad (AUP-UCI): Es una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI. Persigue como objetivo aumentar la calidad del *software* que se produce, de ahí la importancia de aplicar buenas prácticas, para ello nos apoyaremos en el modelo CMMI-DEV v1.3. El cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora. Estas prácticas se centran en el desarrollo de productos y servicios de calidad. (Sánchez, 2015)

La metodología a usar como guía para alcanzar el cumplimento del objetivo de la investigación es AUP-UCI en su escenario número 4, por las siguientes razones:

- Recomendable para proyectos de mediano alcance, en los que el negocio a informatizar esté bien definido y el equipo de desarrollo esté siempre acompañado por el cliente.
- Simplicidad: Todo se describe concisamente utilizando poca documentación.

1.11 Conclusiones parciales

- A lo largo de la investigación realizada se pudo adquirir el conocimiento que permitió identificar los conceptos principales para fundamentar teóricamente la propuesta de solución.
- Se realizó un estudio de las metodologías y las herramientas más usadas para el desarrollo de proyectos relacionados con sistemas embebidos. El objetivo principal de este estudio fue seleccionar las metodología y herramientas que permitan darle cumplimiento a los objetivos planteados.

Capítulo 2: Análisis y diseño de la propuesta de solución

2.1 Introducción

En este apartado se precisan un conjunto de elementos para conformar la propuesta de solución de la presente investigación. Se realiza un análisis de las características del hardware y sus requerimientos, además de describir el funcionamiento general del sistema que se desea implementar. Se realiza una descripción de la arquitectura propuesta para la solución, así como el uso de patrones presentes en el mismo.

2.2 Propuesta del sistema

Luego de haber analizado la información necesaria, identificar las necesidades a desarrollar y tener seleccionadas las herramientas que se utilizarán para el desarrollo de la solución al problema planteado; se decide implementar un sistema de control para una cámara Pant Tilt basada en motores de paso. Este sistema está diseñado para controlar, desde un *Smartphone*, el direccionamiento de una cámara PT utilizando para ello la plataforma Arduino integrada. Con el objetivo de garantizar este control, este sistema está conformado por dos partes que consisten en una aplicación de control ejecutada desde el *Smartphone,* donde se recogen las peticiones de direccionamiento por parte de los usuarios. Mientras que, por otro lado, se desarrolla un firmware que es embebido en el microcontrolador AVR de la placa Arduino integrada en la cámara de seguridad. Ambas partes se comunican a través de tecnología *Bluetooth,* conformando así el sistema de control antes mencionado.

La aplicación permitirá al usuario escoger el tipo de control sobre la cámara, los diferentes tipos son:

- **Manual**: Permite el movimiento paso a paso de la cámara, donde un paso equivale a un giro de 1° hasta llegar a 360° en el eje horizontal y 180° en el eje vertical.
- **Automática**: Permite el movimiento automático de la cámara, para ello la aplicación ofrece al usuario la posibilidad de seleccionar el grado y el eje que utilizará el dispositivo para mantener su movimiento.

El sistema embebido es el encargado de controlar los motores de paso que activan el movimiento de la cámara. Para ello utiliza los pines de entrada/salida que proporciona el microcontrolador AVR, lo que permite transformar las peticiones recibidas de la aplicación de control en pulsos digitales que se envían a los motores para realizar un movimiento rotatorio incremental en la cámara.

2.3 Requerimientos del sistema

Los requerimientos de software son especificaciones que debe cumplir el sistema que se va a desarrollar. Estos requerimientos se clasifican en funcionales y no funcionales. Los funcionales son capacidades o condiciones que el sistema debe cumplir, mientras que los no funcionales son propiedades o cualidades que hacen al producto atractivo, usable, rápido y confiable.

2.3.1 Requisitos funcionales del sistema

Los requisitos básicamente son aquellas funcionalidades que deben cumplir los sistemas basados en las necesidades del cliente, estos pueden ser funcionales y no funcionales (STD, 1998). A continuación, se describen los requisitos asociados a la propuesta de solución.

- Establecer conexión del *Smartphone* con el Arduino.
- Direccionar de forma manual la cámara hacia la derecha.
- Direccionar de forma manual la cámara hacia la izquierda.
- Direccionar de forma manual la cámara hacia arriba.
- Direccionar de forma manual la cámara hacia abajo.
- Direccionar de forma automática la cámara en el eje horizontal.
- Direccionar de forma automática la cámara en el eje vertical.
- Repetir movimiento automático de la cámara de vigilancia.
- Direccionar la cámara hacia su posición inicial de forma automática.

2.3.2 Requisitos no funcionales

Atributo de Calidad	Usabilidad
Sub-atributos/Sub-características	
Objetivo	La aplicación Android debe contar con una interfaz de fácil entendimiento para que los usuarios inexpertos puedan interactuar fácilmente.
Origen	Externo
Artefacto	
Entorno	Ejecución de la aplicación de control.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**

1.a Interacción con el sistema	
Medida de respuesta	

Tabla 3 Requisito no funcional #1

Atributo de Calidad	Software
Sub-atributos/Sub-características	
Objetivo	La aplicación Android debe poder instalarse en Android 4.0 IceCream o superior.
Origen	Externo
Artefacto	
Entorno	Ejecución de la aplicación de control.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1.a Interacción con el sistema	
Medida de respuesta	

Tabla 4 Requisito no funcional #2

Atributo de Calidad	Seguridad
Sub-atributos/Sub-características	
Objetivo	Se debe garantizar la seguridad de la conexión vía Bluetooth con el sistema de alarma.
Origen	Externo
Artefacto	
Entorno	Ejecución de la aplicación de control.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1.a Interacción con el sistema	
Medida de respuesta	

Tabla 5 Requisito no funcional #3

2.4 Descripción de las Historias de Usuarios

Una historia de usuario (HU) es una técnica que se utiliza para especificar los requisitos de las aplicaciones software en las metodologías ágiles (Sánchez, 2015), en ellas el interesado describe brevemente (con el fin de que sean dinámicas y flexibles) las características que el sistema debe poseer. Las HU son representadas por tablas que contienen los siguientes campos o secciones:

- Número: Las siglas de HU precedido de un número representa el identificador de la historia.
- Nombre del requisito: Nombre que identifica la HU.
- Referencia: Es el conjunto de identificadores de las HU a las cuales depende la historia actual que está en desarrollo.
- Prioridad: El valor de este campo es Baja, Media o Alta, definido por el cliente dependiendo de la importancia y el orden para el proceso de desarrollo.
- Iteración Asignada: Número de la iteración en la cual se desarrollará la HU.
- Puntos Estimados: Tiempo estimado en días que se le asignará.
- Descripción: Breve descripción del proceso que define la historia.
- Observaciones: Algunas aclaraciones que es importante señalar acerca de la historia.
- Prototipo de interfaz: Interfaz correspondiente a cada requisito que visualiza el usuario.

Se realizaron un total de 10 historias de usuario (todas con prioridad alta). Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores, utilizando como medida el punto. Un punto, equivale a una semana ideal de programación (5 días laborables). Para su confección se tuvieron en cuenta opiniones de diferentes especialistas, estas se especifican a continuación.

Número: 1	Nombre del requisito: Establecer conexión del *Smartphone* con el arduino.
Programador: Yassiel Gómez Díaz	Iteración Asignada: 1
Prioridad: Alta	Punto Estimado:1
Riesgo en Desarrollo: Alta	Referencia: -

Descripción: La comunicación entre el *Smartphone* y el arduino se establecerá por vía bluetooth.
Observaciones: -EL bluetooth del móvil y el de la placa arduino deben estar activados y visibles.
Prototipo de interfaz:

Tabla 6. Historia de Usuario #1

Número: 2	Nombre del requisito: Direccionar de forma manual la cámara hacia la derecha.
Programador: Yassiel Gómez Díaz	Iteración Asignada: 2
Prioridad: Alta	Punto Estimado:0.5
Riesgo en Desarrollo: Alta	Referencia: 1
Descripción: La aplicación permitirá al usuario mover la cámara hacia la derecha paso a paso.	
Observaciones: -EL bluetooth del *Smartphone* y el del arduino deben estar activados y visibles. -El movimiento será hasta 360°.	
Prototipo de interfaz:	

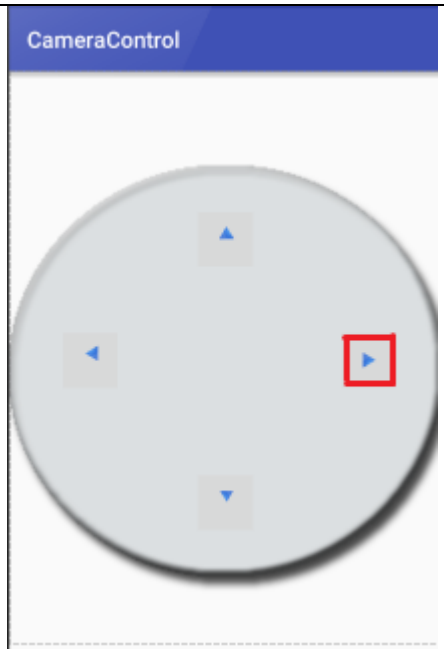

Tabla 7 Historia de usuario #2

Número: 3	Nombre del requisito: Direccionar de forma manual la cámara hacia la izquierda.
Programador: Yassiel Gómez Díaz	Iteración Asignada: 2
Prioridad: Alta	Punto Estimado: 0.5
Riesgo en Desarrollo: Alta	Referencia: 1
Descripción: La aplicación permitirá al usuario mover la cámara hacia la izquierda paso a paso.	
Observaciones: -EL bluetooth del cámara y el del arduino deben estar activados y visibles. -El movimiento será hasta 360°.	
Prototipo de interfaz:	

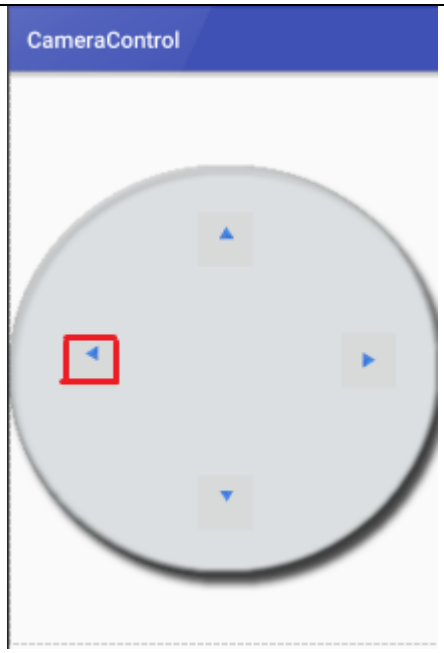

Tabla 8 Historia de usuario #3

Número: 4	Nombre del requisito: Direccionar de forma manual la cámara hacia arriba.
Programador: Yassiel Gómez Díaz	Iteración Asignada: 2
Prioridad: Alta	Punto Estimado: 0.5
Riesgo en Desarrollo: Alta	Referencia: 1
Descripción: La aplicación permitirá al usuario mover la cámara hacia arriba paso a paso.	
Observaciones: -EL bluetooth del *Smartphone* y el del arduino deben estar activados y visibles. -El movimiento será hasta 180°.	
Prototipo de interfaz:	

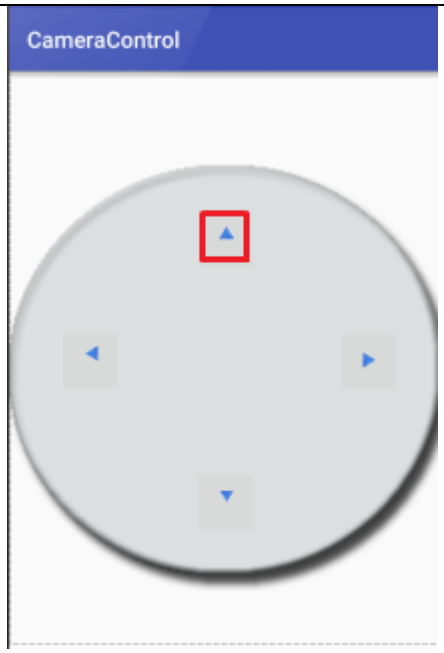

Tabla 9 Historia de usuario #4

Número: 5	Nombre del requisito: Direccionar de forma manual la cámara hacia abajo.
Programador: Yassiel Gómez Díaz	Iteración Asignada: 0.5
Prioridad: Alta	Punto Estimado: 0.2
Riesgo en Desarrollo: Alta	Referencia: 1
Descripción: La aplicación permitirá al usuario mover la cámara hacia abajo paso a paso.	
Observaciones: -EL bluetooth del *Smartphone* y el del arduino deben estar activados y visibles. -El movimiento será hasta 180°.	
Prototipo de interfaz:	

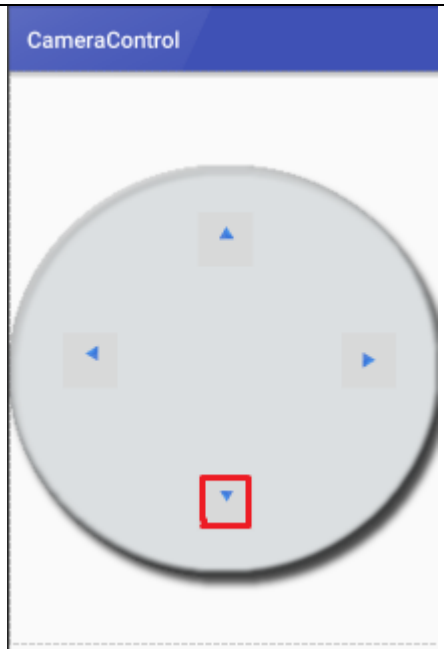

Tabla 10 Historia de usuario #5

Número: 6	Nombre del requisito: Direccionar de manera automática la cámara en el eje horizontal.
Programador: Yassiel Gómez Díaz	Iteración Asignada: 2
Prioridad: Alta	Punto Estimado:0.8
Riesgo en Desarrollo: Alta	Referencia: 1
Descripción: La aplicación permitirá al usuario que se mantenga moviendo la cámara (x grados) en el eje horizontal.	
Observaciones: -EL bluetooth del *Smartphone* y el del arduino deben estar activados y visibles. -El movimiento será desde 0 hasta 360°.	
Prototipo de interfaz:	

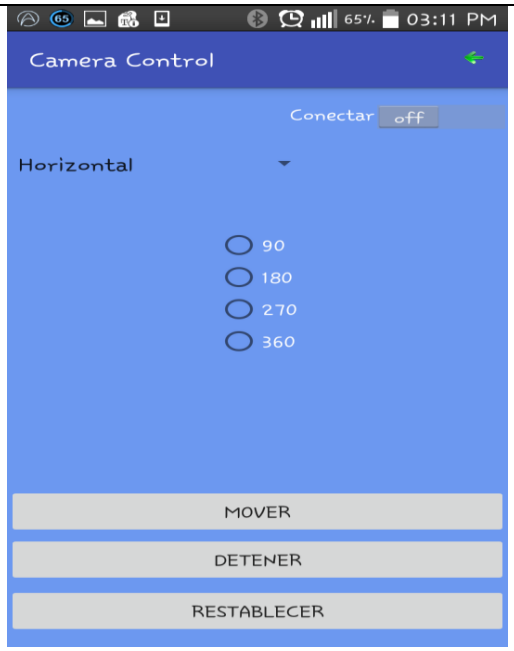

Tabla 11 Historia de usuario #6

Número: 7	Nombre del requisito: Direccionar de forma automática la cámara en el eje vertical.
Programador: Yassiel Gómez Díaz	Iteración Asignada: 2
Prioridad: Alta	Punto Estimado: 0.8
Riesgo en Desarrollo: Alta	Referencia: 1
Descripción: La aplicación permitirá al usuario que se mantenga moviendo la cámara (x grados) en el eje vertical.	
Observaciones: -EL bluetooth del *Smartphone* y el del arduino deben estar activados y visibles. -El movimiento será desde 0 hasta 180°.	
Prototipo de interfaz:	

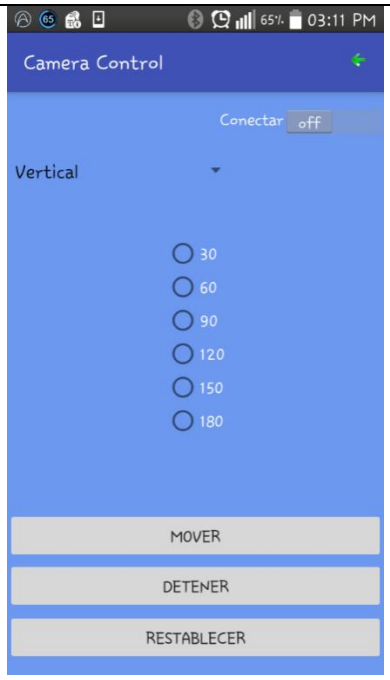

Tabla 12 Historia de usuario #7

Número: 8	Nombre del requisito: Repetir el movimiento automático de la cámara de vigilancia.
Programador: Yassiel Gómez Díaz	Iteración Asignada: 3
Prioridad: Alta	Punto Estimado: 0.5
Riesgo en Desarrollo: Alta	Referencia: 1
Descripción: La aplicación permitirá al usuario realizar nuevamente el último movimiento automático que la cámara realizó.	
Observaciones: -EL bluetooth del *Smartphone* y el del arduino deben estar activados y visibles.	
Prototipo de interfaz:	

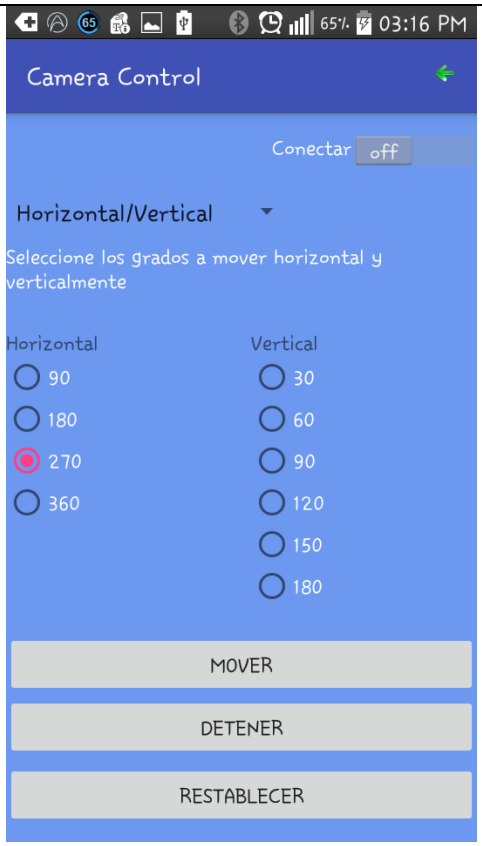

Tabla 13 Historia de usuario #8

Número:9	Nombre del requisito: Direccionar la cámara hacia su posición inicial de forma automática.
Programador: Yassiel Gómez Díaz	Iteración Asignada: 3
Prioridad: Alta	Punto Estimado:1
Riesgo en Desarrollo: Alta	Referencia: 1
Descripción: La aplicación permitirá al usuario mover la cámara hacia su estado inicial.	
Observaciones: -EL bluetooth del *Smartphone* y el del arduino deben estar activados y visibles.	
Prototipo de interfaz:	

Tabla 14 Historia de usuario #9

2.5 Descripción de la arquitectura del sistema de control

La propuesta de solución está conformada por una arquitectura en capas para un sistema distribuido. Esta arquitectura define cómo organizar el modelo de diseño en capas que pueden estar distribuidas de forma física, lo cual quiere decir que los componentes de una capa sólo pueden hacer referencia a componentes en capas inferiores inmediatas.
Principales estilos de este patrón arquitectónico:

- Arquitecturas de dos capas.
- Arquitecturas de tres capas.
- Arquitecturas de n capas.

Por las características del sistema de control, se aplica el estilo arquitectónico dos capas, debido a que, el modelo de diseño está organizado mediante dos capas físicas, identificadas como la aplicación de control ejecutada desde un *Smartphone* y la plataforma Arduino como sistema embebido controlador de la cámara de vigilancia.
Para representar la arquitectura se modela en diagramas de paquetes, estos permiten centrarse en ilustrar el acoplamiento paquete-paquete con la ayuda de las líneas de dependencia (Barrios Frometa, et al., 2015). En la ilustración 8 se muestra una representación del patrón arquitectónico aplicado a la solución.

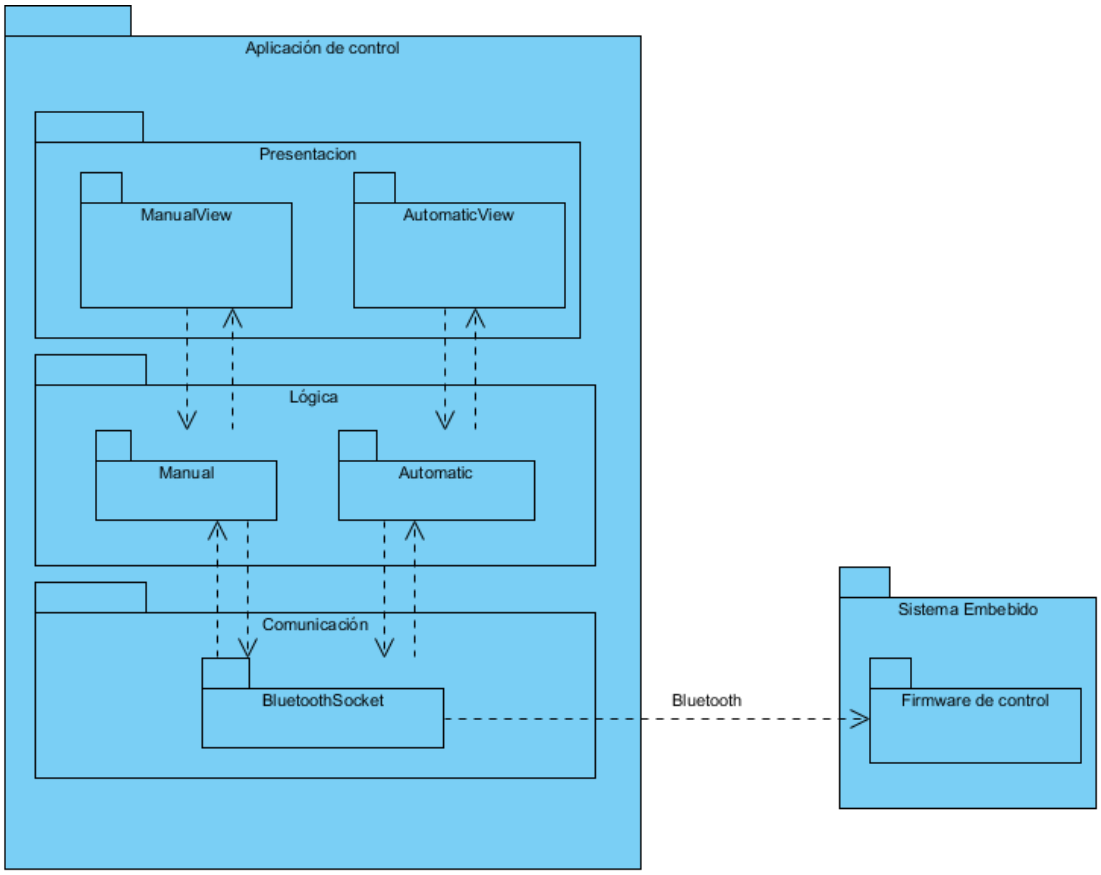

Ilustración 8. Representación del patrón arquitectónico

La capa física referente a la aplicación de control presenta una arquitectura basada en tres capas, donde intervienen las capas de presentación, lógica y comunicación, estas permiten contar con las funcionalidades necesarias para el desarrollo de la aplicación. La primera capa es la encargada de mostrar todos los componentes visuales de la aplicación de control, mientras que la segunda gestiona las peticiones de la capa de presentación, y por último la capa de comunicación permite establecer las conexiones entre el sistema embebido y la aplicación de control, además posibilita hacer uso del protocolo *Bluetooth* mediante el que se envían las órdenes de direccionamiento.

La implementación de la capa referente al sistema embebido respeta una arquitectura basada en eventos (EDA, por sus siglas en inglés), donde se mantiene un canal de comunicación al establecerse la conexión vía *Bluetooth*. Se aplica el modelo de control basado en interrupciones que recomienda (Sommerville, 2006). para la utilización en sistemas de tiempo real. Esta arquitectura propone que se establezcan dos componentes fundamentales: el agente emisor y el agente consumidor. En la propuesta el papel de emisor lo cumple la aplicación de control, quien se encarga de emitir las peticiones de

direccionamiento, mientras que el *firmware* implementa el agente consumidor que maneja las interrupciones o eventos emitidos, lo que permite cambiar el estado de la cámara.

Para obtener una correcta implementación se hace necesario tener en cuenta las mejores soluciones conocidas, con el objetivo de diseñar las clases y las funcionalidades a codificar. Para ello se analizan y proponen algunos patrones de diseño que responden ante las necesidades de desarrollo.

2.6 Patrones de Diseño

Los patrones de diseño son un conjunto de estrategias, o buenas prácticas, que pueden facilitar el trabajo en muchas situaciones a la hora de realizar una aplicación orientada a objetos. Además, son los encargados de identificar clases, instancias, roles, colaboraciones y la distribución de responsabilidades. (Fernandez, 2009)

Para la asignación general de responsabilidades en el software existen patrones, *General Responsability Assignment Software Patterns* (GRASP), que describen los principios fundamentales de asignación de responsabilidades a objetos, expresadas como patrones. (Larman, 2002)Seguidamente se exponen algunos patrones utilizados dentro del contexto del sistema a implementar:

Experto:

Asignar responsabilidades al experto de la información, es decir, a la clase que tiene la información necesaria para llevar la tarea a cabo. Este patrón se evidencia en las clases Manual.Java y Automatic.Java debido a que son las clases responsables del control manual y el control automático respectivamente. A continuación, una representación del patrón experto en la clase Manual.

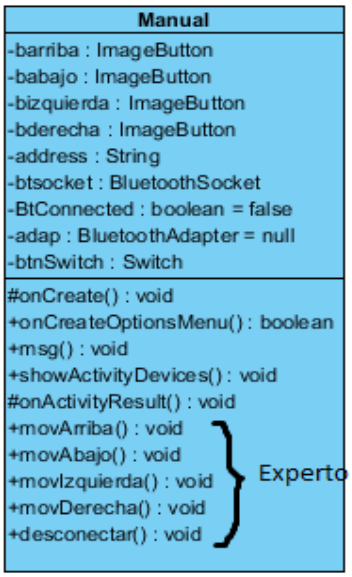

Ilustración 9 Representación del patrón experto en la clase manual

Bajo acoplamiento: Se asignan las responsabilidades de manera que el acoplamiento sea bajo. Un componente con bajo acoplamiento no mantiene fuertes dependencias con otros componentes. Cuando este es alto se pueden presentar estos problemas**:**

- Los cambios locales son difíciles de realizar.
- Son difíciles de analizar de forma independiente.
- Se vuelven difíciles de reutilizar.

Este patrón se evidencia en las clases Manual.java y Automatic.java, de tal manera que, son clases que no presentan ninguna dependencia entre ellas, garantizando así que si se produce algún cambio en alguna de ellas no se necesita modificar nada en las clases restantes.

2.7 Conclusiones Parciales

En este capítulo se han abordado los aspectos referentes a la concepción del producto a desarrollar y sus características funcionales. Esto permitió arribar a las siguientes conclusiones:

• A partir de la definición de las HU se pudieron identificar las principales funcionalidades a desarrollar en correspondencia con los movimientos que realiza la cámara de seguridad.

• Fue posible identificar los patrones GRASP para el desarrollo de la solución.

Con los aspectos arquitectónicos y de diseño a utilizar definidos, queda establecida la vía para llevar a cabo la implementación y prueba del control sobre la cámara.

Capítulo 3 Implementación y Pruebas

3.1 Introducción

Durante el presente capítulo se realiza el diseño de la implementación del sistema propuesto y se verifica el cumplimiento de los requisitos funcionales de la aplicación mediante pruebas, para evaluar la calidad del producto desarrollado.

3.2 Implementación del sistema

3.2.1 Diagrama de componentes

Este tipo de diagramas representa cómo un sistema de software es dividido en componentes, que pueden ser archivos, módulos, bibliotecas, ejecutables o paquetes y muestra las dependencias entre ellos. En la ilustración 10 podemos observar el diagrama de componentes de la propuesta de solución. Está compuesto por los componentes que conforman el sistema de control, dígase la aplicación de control y el firmware de control. La aplicación de control se asocia con los componentes referentes al tipo de movimiento que le permitirá al usuario realizar ya sea automático o manual estableciendo la conexión con el firmware de control mediante la biblioteca BluetoothSocket.

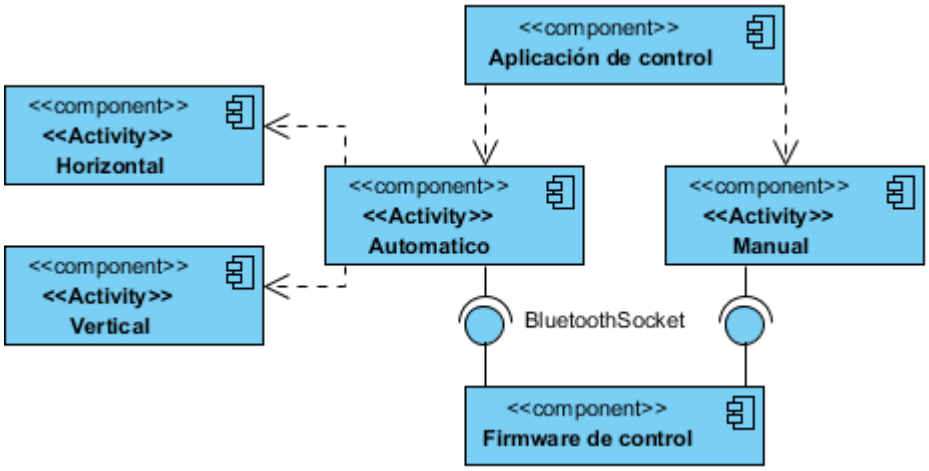

Ilustración 10 Representación del diagrama de componentes de la solución

3.2.2 Diagrama de despliegue

Un diagrama de despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos.

A continuación, se muestra el diagrama de despliegue del sistema propuesto, integrado por una aplicación de control y una cámara de vigilancia PT. La cámara de vigilancia está compuesta por una plataforma arduino que contiene un microcontrolador AVR con un firmware de control; además la cámara contiene un módulo bluetooth para establecer la comunicación con la aplicación de control y también dos motores de paso encargados del movimiento pant tilt. La comunicación entre la aplicación y la cámara es mediante el servicio bluetooth garantizando una comunicación confiable y segura; mientras que la conexión entre el modulo bluetooth y los motores de paso con la plataforma arduino será mediante los pines E/S.

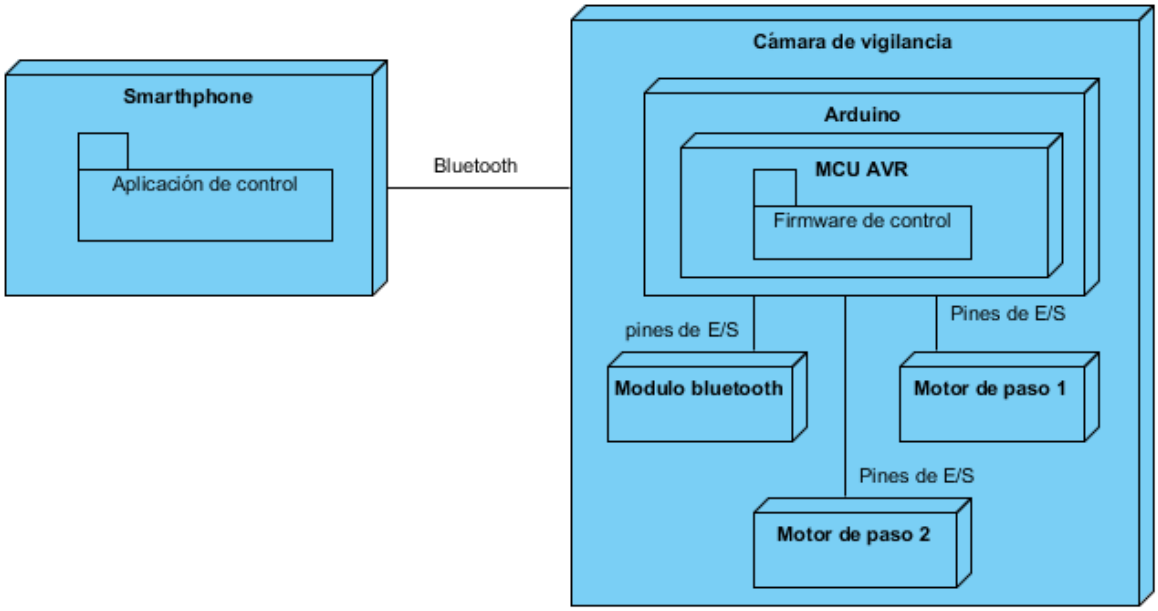

Ilustración 11 Representación del diagrama de despliegue

3.3 Pruebas

Las pruebas en el proceso de desarrollo de software representan un conjunto de actividades que se pueden llevar a cabo sistemáticamente. Estas persiguen como objetivo detectar los errores y fallos que se cometan durante el desarrollo, lo cual contribuye a elevar la calidad de los productos desarrollados y a la seguridad de los programadores a la hora de introducir cambios o modificaciones.

La metodología AUP-UCI divide las pruebas en tres disciplinas: pruebas internas, desarrolladas por los programadores, encargadas de verificar el código de forma automática, las pruebas de liberación diseñadas y ejecutadas por una entidad certificadora

de la calidad externa y las pruebas de aceptación, destinadas a evaluar si al final de una iteración se obtuvo la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la esperada por el cliente.Una prueba de software se dice que es exitosa si detecta un error que no se había encontrado hasta entonces ya que es este su objetivo. Para probar un software existen distintos métodos:

- Comprobar los requisitos del software: para ello se utiliza fundamentalmente el método de caja negra.
- Comprobar la lógica interna del software: para ello se utiliza fundamentalmente el método de caja blanca.

A través del método de prueba caja negra la cual se lleva a cabo sobre la interfaz del software. Se realizaron las pruebas de aceptación al sistema propuesto, verificando que el sistema cumpla con los requisitos establecidos por el usuario y demostrando que las funciones del software son operativas y que las entradas se aceptan de forma adecuada y se produce un resultado correcto. De esta forma se puede obtener el grado de satisfacción del cliente.

3.3.1 Pruebas de aceptación

Las pruebas de aceptación son creadas en base a las HU, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una HU ha sido correctamente implementada. Las pruebas de aceptación son consideradas como “pruebas de caja negra”, Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos (JOSKOWICZ, 2008). El cliente encargado de la validación de las pruebas de aceptación fue el 1er Te Yuniesky Orlando Vasconcelo Mir trabajador de la XETID.

A continuación, se detallan varios casos de prueba diseñados para comprobar el correcto funcionamiento de los requisitos descritos por las HU. Para ver los restantes casos de prueba de las otras iteraciones consultar los anexos 1

Caso de prueba de aceptación	
Código: HU2_P1	**Historia de usuario:** 2
Nombre: Direccionar de forma manual la cámara hacia la derecha.	

Descripción: Prueba para la funcionalidad que permite el movimiento hacia la derecha de la cámara.
Condiciones de ejecución: Debe estar activado el servicio bluetooth. Debe estar establecida la conexión del Smartphone con el arduino
Entrada/Pasos de ejecución: Manual->Conectar->"Botón derecha"
Resultado esperado: Movimiento de la cámara satisfactoriamente hacia la derecha
Evaluación de la prueba: Satisfactoria

Tabla 15 Prueba de Aceptación para la HU2

Caso de prueba de aceptación	
Código: HU6_P1	**Historia de usuario:** 6
Nombre: Direccionar de forma automática la cámara en el eje horizontal	
Descripción: Prueba para la funcionalidad que permite el movimiento hacia la izquierda/derecha de la cámara automáticamente.	
Condiciones de ejecución: Debe estar activado el servicio bluetooth. Debe estar establecida la conexión del Smartphone con el arduino	
Entrada/Pasos de ejecución: Manual->Conectar->"Seleccionar ángulo"->Mover	
Resultado esperado: Movimiento de la cámara satisfactoriamente en el eje horizontal.	
Evaluación de la prueba: Satisfactoria	

Tabla 16 Prueba de Aceptación para la HU6

La ejecución de las pruebas a las HU fue realizada en dos iteraciones, quedando distribuidas las historias de usuario de la siguiente manera.

Iteraciones	Historias de usuario
1	**HU1** Establecer comunicación entre el *Smartphone* y el arduino.

	HU2 Direccionar de forma manual la cámara hacia la derecha. **HU3** Direccionar de forma manual la cámara hacia la izquierda. **HU4** Direccionar de forma manual la cámara hacia arriba. **HU5** Direccionar de forma manual la cámara hacia abajo.
2	**HU6** Direccionar de forma automática la cámara en el eje horizontal. **HU7** Direccionar de forma automática la cámara en el eje vertical. **HU8** Detener la cámara de vigilancia. **HU9** Direccionar la cámara hacia su posición inicial de forma automática.

Tabla 17 Representación de las iteraciones para la realización de las pruebas

Las pruebas van dirigidas esencialmente a las funcionalidades del sistema, pero también contribuyen a la detección de errores no significativos durante su proceso como son: Errores ortográficos, errores de interfaz. Como resultado de la ejecución de este tipo de pruebas se obtienen los siguientes resultados:

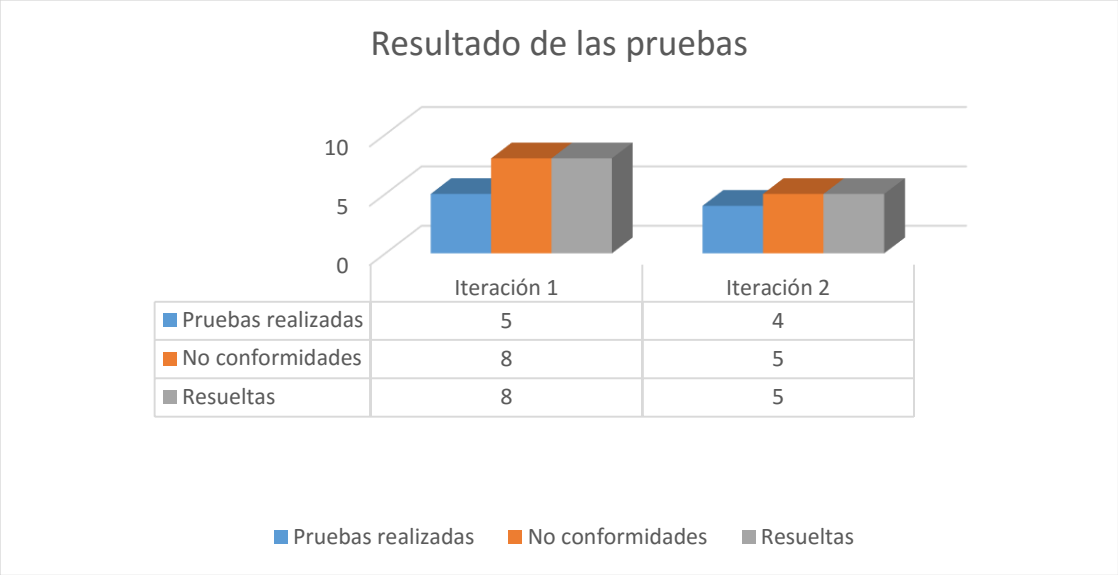

Ilustración 12 Resultado de las pruebas

En la primera ejecución se realizaron un total de 5 pruebas al sistema encontrando un total de 8 no conformidades (errores) distribuidos de la siguiente manera: cinco pertenecientes a los requerimientos del sistema y uno en cuanto al diseño de interfaz, ortografía e idioma respectivamente. Según propone la metodología AUP-UCI para pasar a la siguiente iteración de pruebas es necesario que todas las no conformidades detectadas en iteraciones anteriores sean corregidas satisfactoriamente. En la segunda ejecución se realizaron un total de 4 pruebas al sistema encontrando un total de 5 no

conformidades todas significativas ya que se trataban de requerimientos que el sistema debía cumplir.

A modo de conclusión, los resultados obtenidos luego de la realización de las pruebas fueron satisfactorios, debido a que las no conformidades encontradas durante su ejecución fueron deficiencias de tipo significativas y no significativas para el correcto funcionamiento de la aplicación, las cuales fueron corregidas.

3.4 Conclusiones parciales

- Mediante las pruebas de aceptación se pudo documentar las no conformidades detectadas a lo largo de todas las iteraciones del proyecto, las cuales fueron corregidas, permitiendo que el sistema tenga un correcto funcionamiento.

Conclusiones generales

Con el desarrollo de la presente investigación se arriba a la siguiente conclusión:

• La implementación del sistema de control propuesto permite un correcto control del movimiento de una cámara Pant Tilt, proporcionando un mejor monitoreo de las áreas a visualizar.

Recomendaciones

- Incorporar en la aplicación un módulo de visualización del video capturado por la cámara a controlar.
- Incorporar al sistema la funcionalidad de permitir configurar múltiples movimientos de la cámara.

Anexo

Caso de prueba de aceptación	
Código: HU1_P1	**Historia de usuario:** 1
Nombre: Establecer conexión del *Smartphone* con el arduino.	
Descripción: Prueba para la funcionalidad que permite la conexión entre el Smartphone y la placa arduino.	
Condiciones de ejecución: Debe estar activado el servicio bluetooth.	
Entrada/Pasos de ejecución: Manual->Conectar o Automático->Conectar	
Resultado esperado: El Smartphone ha sido conectado satisfactoriamente.	
Evaluación de la prueba: Satisfactoria	

Tabla 18 Prueba de Aceptación para la HU1

Caso de prueba de aceptación	
Código: HU3_P1	**Historia de usuario:** 3
Nombre: Direccionar de forma manual la cámara hacia la izquierda.	
Descripción: Prueba para la funcionalidad que permite el movimiento hacia la izquierda de la cámara.	
Condiciones de ejecución: Debe estar activado el servicio bluetooth. Debe estar establecida la conexión del Smartphone con el arduino	
Entrada/Pasos de ejecución: Manual->Conectar->"Botón izquierda"	

Resultado esperado: Movimiento de la cámara satisfactoriamente hacia la izquierda
Evaluación de la prueba: Satisfactoria

Tabla 19 Prueba de Aceptación para la HU2

Caso de prueba de aceptación	
Código: HU4_P1	**Historia de usuario:** 4
Nombre: Direccionar de forma manual la cámara hacia arriba.	
Descripción: Prueba para la funcionalidad que permite el movimiento hacia arriba de la cámara.	
Condiciones de ejecución: Debe estar activado el servicio bluetooth. Debe estar establecida la conexión del Smartphone con el arduino	
Entrada/Pasos de ejecución: Manual->Conectar->"Botón arriba"	
Resultado esperado: Movimiento de la cámara satisfactoriamente hacia arriba	
Evaluación de la prueba: Satisfactoria	

Tabla 20 Prueba de Aceptación para la HU4

Caso de prueba de aceptación	
Código: HU5_P1	**Historia de usuario:** 5
Nombre: Direccionar de forma manual la cámara hacia abajo.	
Descripción: Prueba para la funcionalidad que permite el movimiento hacia abajo de la cámara.	
Condiciones de ejecución:	

Debe estar activado el servicio bluetooth. Debe estar establecida la conexión del Smartphone con el arduino
Entrada/Pasos de ejecución: Manual->Conectar->”Botón abajo”
Resultado esperado: Movimiento de la cámara satisfactoriamente hacia abajo
Evaluación de la prueba: Satisfactoria

Tabla 21 Prueba de Aceptación para la HU5

Caso de prueba de aceptación	
Código: HU6_P1	**Historia de usuario:** 6
Nombre: Direccionar de forma automática la cámara en el eje horizontal.	
Descripción: Prueba para la funcionalidad que permite el movimiento derecha/izquierda de la cámara automáticamente según el grado seleccionado por el usuario.	
Condiciones de ejecución: Debe estar activado el servicio bluetooth. Debe estar establecida la conexión del Smartphone con el arduino.	
Entrada/Pasos de ejecución: Manual->Conectar->”Seleccionar ángulo”->Mover	
Resultado esperado: Movimiento de la cámara satisfactoriamente hacia la izquierda/derecha	
Evaluación de la prueba: Satisfactoria	

Tabla 22 Prueba de Aceptación para la HU6

Caso de prueba de aceptación	
Código: HU7_P1	**Historia de usuario: 7**

Nombre: Direccionar de forma automática la cámara en el eje vertical.
Descripción: Prueba para la funcionalidad que permite el movimiento arriba/abajo de la cámara automáticamente según el grado seleccionado por el usuario.
Condiciones de ejecución: Debe estar activado el servicio bluetooth. Debe estar establecida la conexión del Smartphone con el arduino.
Entrada/Pasos de ejecución: Manual->Conectar->"Seleccionar ángulo"->Mover
Resultado esperado: Movimiento de la cámara satisfactoriamente arriba/abajo automáticamente.
Evaluación de la prueba: Satisfactoria

Tabla 23 Prueba de Aceptación de la HU7

Caso de prueba de aceptación	
Código: HU8_P1	**Historia de usuario:** 9
Nombre: Repetir movimiento de la cámara de vigilancia..	
Descripción: Prueba para la funcionalidad que permitirá repetir el movimiento automático de la cámara.	
Condiciones de ejecución: Debe estar activado el servicio bluetooth. Debe estar establecida la conexión del Smartphone con el arduino.	
Entrada/Pasos de ejecución: Manual->Conectar->Repetir	
Resultado esperado: Movimiento repetido.	

Evaluación de la prueba: Satisfactoria

Tabla 24 Prueba de Aceptación de la HU9

Caso de prueba de aceptación	
Código: HU9_P1	**Historia de usuario:** 10
Nombre: Direccionar de forma automática la cámara hacia su posición inicial mediante un *Smartphone*.	
Descripción: Prueba para la funcionalidad que permite el movimiento de la cámara automáticamente hacia su posición inicial.	
Condiciones de ejecución: Debe estar activado el servicio bluetooth. Debe estar establecida la conexión del Smartphone con el arduino.	
Entrada/Pasos de ejecución: Manual->Conectar->Restablecer	
Resultado esperado: Movimiento de la cámara satisfactoriamente hacia su posición inicial	
Evaluación de la prueba: Satisfactoria	

Tabla 25 Prueba de Aceptación de la HU10

Bibliografía

Díaz , María Eugenia . 2009. Teoría de control. Teoría de control. [En línea] 9 de 11 de 2009. [Citado el: 15 de 2 de 2016.] http://teoriadecontrol.blogspot.com/2009/11/clasificacion-de-los-sistemas-de.html.

Galiana Llinare, Antonio Nadal. 2005. Sistemas Embebidos. 2005.

la Red Martínez, David Luis. 2001. "SISTEMAS OPERATIVOS". Argentina : UNIVERSIDAD NACIONAL DEL NORDESTE, 2001.

MORALES BEJARANO , JOSÉ DANIEL y VARGAS BURGOS, DAVID GONZALO. 2012. DISEÑO, CONSTRUCCIÓN E IMPLEMENTACIÓN DE UN SISTEMA DE TELEMETRÍA PARA UN VEHÍCULO DE KARTING. Quito : s.n., 2012.

NÚÑEZ MORI, JOSE GERMÁN . 2010. USABILIDAD EN METODOLOGÍAS ÁGILES. 2010.

ROCHA NÚÑEZ , JOSÉ MANUEL y Lara Hernandez, ELIZABETH . Introducción a los sistemas de control.

39A, Grupo de investigacion. 2012. INFLUENCIA DE LOS SMARTPHONE EN LOS ESTILOS DE VIDA DE LOS JOVENES UNIVERSITARIOS Y JÓVENES EJECUTIVOS EN LA CIUDAD DE BOGOTA. Bogotá : s.n., 2012.

Android, Acedemia. 2005. academia android. [En línea] Digital Learning, 2005. http://academiaandroid.com/ide-android-intellij-android-studio-aide/.

Anton, Otilia , Gelineau, Brice y Sauget, Jeremy . 2012. Firmware and bootloader. 2012.

Apitz, Javier Colmenares. 2007. Sistema de bibliotecas. Sistema de bibliotecas. [En línea] 2007. www.Monografias.com..

Arduino. 2016. Arduino. [En línea] Arduino, 2016. https://www.arduino.cc/en/Main/Software.

Axis. 2016. Axis Communications. Axis Communications. [En línea] Axis, 2016. [Citado el: 20 de 2 de 2016.] http://www.axis.com/global/es/learning/web-articles/technical-guide-to-network-video/types-of-network-cameras.

Barrios Frometa, Antuanett y Pérez Attanaff, Yordan Carlos. 2015. Sistema de monitoreo basado en microcontroladores AVR. Habana : s.n., 2015.

CAMSEG. 2016. Cámara de Seguridad. Cámara de Seguridad. [En línea] CAMSEG, 2016. [Citado el: 5 de 3 de 2016.] http://camara-seguridad.com.mx/blog/consejos/140-tipos-de-camaras-de-seguridad-mas-populares.

Canto, Carlos. 2013. Motores de Paso o Steppers Motors. 2013.

Dorf, Richard y Bishop, Robert. 2001. Introduction to Control Systems. s.l. : Prentice Hall, 2001.

Durán Rocha, Andrés . 2015. HETPRO(Herramientas Tecnológicas Profesionales). HETPRO(Herramientas Tecnológicas Profesionales). [En línea] HETPRO, 2015.

eHow. 1999-2016. eHow en Español. eHow en Español. [En línea] 1999-2016. http://www.ehowenespanol.com/camara-ptz-sobre_76400/.

Electricaland. 2016. Electricaland Computer Engeneering. [En línea] NC State University, 2016. http://www.ece.ncsu.edu/research/cas/ecs.

Estudio, Electronica. 2006. Electronica Estudios.com. Electronica Estudios.com. [En línea] PICmicro®, 2006. http://www.electronicaestudio.com/microcontrolador.htm.

Fernandez, Luis H. 2009. Software Guisho. [En línea] 2009. http://software.guisho.com/patrones-de-diseño.

Gimson , Loraine . 2012. Metodologías ágiles y desarrollo basado en conocimiento. s.l. : UNIVERSIDAD NACIONAL DE LA PLATA, FACULTAD DE INFORMÁTICA, 2012.

Hacedores. 2016. Hacedores. Maker Comunity. [En línea] 2016. http://hacedores.com/arduino-o-raspberry-pi-cual-es-la-mejor-herramienta-para-ti/.

Ibarra Esquer, M.C. Jorge Eduardo Ibarra. 2012. Microprocesadores y Microcontroladores. 2012.

Jacobson, Ivar y Rumbaugh, James. 2000. El Proceso Unificado de Desarrollo de Software. Madrid : Madrid, 2000. 84-7829-036-2.

JOSKOWICZ, José. 2008. Reglas y prácticas en eXtreme Programming. 2008.

Larman, Crain. 2002. UML y Patrones. 2002.

Letelier, Patricio y Penadés, Maria Carmen. 2006. Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). Valencia : s.n., 2006.

Lopez Rodriguez, Julio Cesar. 2007. Estructura,funcionamiento y aplicacion de las Camaras IP. 2007.

Mario. 2015-2016. Neoteo. [En línea] Neoteo, 2015-2016. http://www.neoteo.com/modulo-bluetooth-hc-06-android.

Martínez, A. M. R. 2003. Guía a Rational Unified Process. 2003.

Mata Ramírez, Mario Pablo. 2013. Monografias. [En línea] 2013. http://www.monografias.com/trabajos43/tecnologia-bluetooth/tecnologia-bluetooth.shtml.

Nicolas Sosio. 2013. S.O.S Seguridad. S.O.S Seguridad. [En línea] 2013. [Citado el: 1 de 3 de 2016.] http://www.seguridadsos.com.ar/camaras-de-seguridad-tipos-y-modelos/.

Ramirez, Jose Miguel y Rosero Garcia, Esteban Romero . 2007. SISTEMAS DE CONTROL I. Santiago de Cali, Escuela de Ingeniería Eléctrica y Electrónica : UNIVERSIDAD DEL VALLE, 2007.

robotica, Tienda. 2014. GUÍA BÁSICA DE ARDUINO. 2014.

Rodríguez., Pedro Alberto Uriarte. Manejador para la comunicación con dispositivos de campo mediante el protocolo DF1.

Sánchez, Tamara Rodríguez. 2015. Metodología de desarrollo para la Actividad productiva de la UCI. 2015.

Sommerville, Ian. 2006. Software Engineering. 2006.

STD, IEEE. 1998. "Especificaciones de los requisitos del Software". 1998.

Suaza, Katerine Villamizar. 2013. Definición de equivalencias entre historias de usuario y especificaciones en UN-LENCEP para el desarrollo ágil de software. Medellín : s.n., 2013.

TOME, LUCIO MARTIN BENCOMO. 2011. Relevadores en Aplicaciones Para Sistemas de Control APSC. 2011.

Torrente Artero, Óscar . 2013. Arduino. Curso práctico de formación. México DF : Alfaomega Grupo Editor, 2013.

Úbeda Miñarro, Benito . 2009. Sistemas Embebidos. 2009.

VisualParadigm.com. 2016. What is Visual Paradigm? Visual-paradigm.com. [En línea] Visual Paradigm International, 2016. https://www.visual-paradigm.com/features/.