Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Facultad 4

Sistema de geoposicionamiento integral para el control del proceso de orientación de antenas parabólicas de grandes dimensiones.

Autor:

David Rodríguez Feitó

Tutor:

My. Lester Sans Pérez

La Habana. Junio, 2019

Declaración de autoría

Declaro ser el autor del presente trabajo de diploma y otorgo 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 _____.

_____	_____
Firma del Autor	**Firma del Tutor**
David Rodríguez Feitó	My. Lester Sans Pérez

Dedicatoria

Dedico el presente trabajo a mis padres Iraida y Argelio.

Agradecimientos

Ante todo, agradecer a mis padres Iraida y Argelio por haberme apoyado durante toda mi carrera y no perder la confianza en mí a pesar de mis tropiezos.

A mi familia, que siempre se preocupó por mis resultados y pude buscar motivación en ellos cada vez que la necesité.

A Yaima, que me ha acompañado en este viaje durante mucho tiempo y se mantuvo a mi lado tratando de hacerme una persona responsable.

A mi tutor por todo el apoyo brindado.

A Leydis, que me ayudó bastante durante la realización de este trabajo y siempre estuvo ahí para aclarar mis dudas.

A Yasiel, Alejandro, Luisito, el Lima, Gustavo, Yailyn, Shirley y Daylen con quienes compartí esos momentos que hacen a la universidad una etapa única en la vida.

A Richard y Liosvel que me acogieron cuando me quise convertir en un sin techo.

A Yosvani que sin él nunca me hubiese encontrado este tema de investigación.

A Gabriel, Felix, Rogelio, el Jhony, Mederos, Maikel, el Rafa, Iván, Liz Yanet, Mario, Mahy y todas esas personas que se me quedan, por haber compartido una parte de su tiempo conmigo.

A los profesores de las facultades 4 y 5.

A todos los que de una forma u otra ayudaron a la realización de este trabajo.

Resumen

Las tecnologías de la información y las comunicaciones juegan un papel relevante en la economía mundial, siendo estas un factor influyente en cualquier esfera de la sociedad. El continuo avance de estas ha influido en la necesidad de transmitir gran cantidad de información a nivel mundial. Entre los medios de comunicación disponibles para el intercambio de información se encuentra la comunicación vía satélite. Para el correcto funcionamiento de esta es necesario el uso de antenas parabólicas de diferentes tamaños, las que se deben orientar hacia los satélites. Durante esta investigación se desarrolló un sistema de geoposicionamiento integral para el control del proceso de orientación de antenas parabólicas de grandes dimensiones mediante el uso de una Raspberry Pi. Para el desarrollo del sistema se empleó la metodología de desarrollo AUP-UCI y el lenguaje de programación C++, así como un dispositivo MTi Xsens y la biblioteca de clases Xsens API. Como resultado se logró obtener un sistema que permite controlar el proceso de orientación de las antenas parabólicas remotamente, además de permitir la realización de dicho proceso de forma automática.

Palabras clave: antena, comunicación satelital, MTi Xsens, Raspberry Pi, sistema de control.

Abstract

Information and communication technologies play an important role in world economy, being an influential factor in any sphere of society. The continuous advance of these has influenced the need to transmit a large amount of information worldwide. Satellite communication is among the available means of communication for the exchange of information. For the correct operation of this, it is necessary to use parabolic antennas of different sizes, which must be oriented towards the satellites. During this investigation an integral geopositioning system for the control of the orientation process of large parabolic antennas using a Raspberry Pi was developed. For the development of the system, the AUP-UCI development methodology and the C ++ programming language were used, as well as an MTi Xsens device and the Xsens API class library. As a result, a system that allows controlling the orientation process of the parabolic antennas remotely is achieved, as well as allowing the realization of said process automatically.

Keywords: antenna, control system, MTi Xsens, Raspberry Pi, satellite communication.

Índice de contenidos

Índice de imágenes

Índice de tablas

Introducción

La necesidad de información nace con los primeros hombres que habitaron la Tierra. Con el paso del tiempo el hombre fue desarrollando diferentes formas de comunicarse, transitando desde el surgimiento del lenguaje hasta el desarrollo de la mayor red de interconexión existente: la internet (1).

Actualmente las Tecnologías de la Información y las Comunicaciones (TIC) juegan un papel relevante en la economía mundial, siendo estas un factor influyente en cualquier esfera de la sociedad (2). Las TIC se desarrollan a partir de los avances científicos producidos en los ámbitos de la informática y las telecomunicaciones. A medida que han transcurrido los años han llevado un desarrollo acelerado, siendo cada vez más necesario la transmisión de gran cantidad de información a nivel mundial (1). Como parte de los medios de telecomunicaciones disponibles para el intercambio de datos se encuentra la comunicación vía satélite.

Para el funcionamiento de la comunicación satelital es necesario el uso de satélites artificiales que son puestos en órbita por cohetes espaciales. Estos actúan como estaciones repetidoras encargadas de recibir la información transmitida desde las estaciones terrenas, amplificarla y enviarla de regreso a la tierra (3).

Para el envío y recepción de la información en las estaciones terrenas se usan antenas parabólicas. Estas antenas tienen forma de parábola, lo que provoca que las señales se reflejen e incidan sobre el foco de la parábola. Por este motivo las antenas deben ser orientadas hacia el satélite con el que se desea establecer la comunicación (4).

En Cuba se han desarrollado gran cantidad de aplicaciones informáticas con el objetivo de contribuir al desarrollo de la economía y la independencia tecnológica, así como a la informatización de la sociedad. Una de las empresas destacadas en este sentido es la Empresa de Tecnologías de la Información para la Defensa (XETID), la cual tiene como objetivo el desarrollo de soluciones informáticas siguiendo estrictos estándares de calidad y seguridad. La XETID cuenta con varios

centros productivos que se dedican a áreas de negocios especificas asegurando así la calidad de los productos desarrollados.

Como parte de los centros mencionados anteriormente se encuentra la Unidad Básica para la Investigación y Desarrollo (UBID) que se dedica a la creación de prototipos y dispositivos, además del software para los mismos, utilizando tarjetas de hardware abierto como Raspberry Pi o Arduino.

Actualmente la UBID dispone de antenas parabólicas de grandes dimensiones similares a las que se utilizan en Cuba para la conexión e intercambio de datos con satélites ubicados en la órbita terrestre. Las antenas deben ser orientadas hacia el satélite con el cual se desea establecer la conexión, para lo cual cuentan con un sistema de manejo que permite establecer la orientación de las mismas. Este sistema de manejo es totalmente manual, lo que complejiza y retarda en gran medida el proceso de orientación. Además, esto hace imposible el control remoto del mismo y provoca que la sintonía de las antenas requiera de personal altamente calificado y con vasta experiencia en el manejo de estaciones terrenas.

A partir de la situación problemática descrita anteriormente se plantea como **problema a resolver** ¿Cómo contribuir al proceso de orientación de antenas parabólicas de grandes dimensiones para la conexión e intercambio de datos con satélites ubicados en la órbita terrestre?

Definiéndose como **objeto de estudio:** El proceso de orientación de antenas parabólicas de grandes dimensiones.

Como **campo de acción:** Las tecnologías informáticas como soporte del proceso de orientación de antenas parabólicas de grandes dimensiones.

Para dar solución a la problemática planteada se define como **objetivo general:** Desarrollar un sistema de geoposicionamiento integral para el control del proceso de orientación de antenas parabólicas de grandes dimensiones.

Tareas de investigación:

1. Estudiar los principales conceptos relacionados con la comunicación satelital.

2. Seleccionar los estándares, protocolos de comunicación, metodologías, herramientas y tecnologías para el desarrollo del proyecto.
3. Definir los requisitos funcionales para el desarrollo de la solución.
4. Confeccionar los artefactos del análisis y el diseño.
5. Implementar la solución propuesta.
6. Elaborar pruebas y corregir errores en la solución.

Para desarrollar estas tareas se utilizaron los siguientes **métodos científicos:**

Métodos teóricos:

Analítico – sintético: con el objetivo de analizar y utilizar la información y la documentación más relevante para el desarrollo y la realización de este sistema.

Histórico – lógico: para constatar teóricamente y realizar un estudio de todo lo referente al proceso de orientación de antenas parabólicas de grandes dimensiones, la evolución y el nivel de desarrollo que tienen los sistemas.

Modelado: para representar por medio de diagramas los conceptos asociados a los procesos de orientación de antenas parabólicas de grandes dimensiones, obteniendo como resultado un mejor entendimiento de la posible solución a implementar.

Métodos empíricos:

Entrevista: para obtener información interactuando con personas que tienen un mayor conocimiento sobre antenas parabólicas de grandes dimensiones.

Estructuración del contenido

Capítulo 1: Fundamentación teórica: En este capítulo se presenta la fundamentación teórica para el Sistema de geoposicionamiento integral para el control del proceso de orientación de antenas parabólicas de grandes dimensiones. Además, se especifican las tecnologías, herramientas y metodología que se usarán para la solución del problema.

Capítulo 2: Análisis y diseño de la solución: En este capítulo se presenta un modelo general del sistema y una lista de características a desarrollar en la aplicación con un plan de desarrollo para la realización de las mismas.

Capítulo 3: Implementación y pruebas: En este capítulo se detalla el proceso de implementación y prueba realizado al sistema.

Capítulo 1: Fundamentación Teórica

En el presente capítulo se introducen las principales características y conceptos asociados a la comunicación satelital para una mejor comprensión del tema en cuestión. Además, se hace una descripción de las principales tecnologías que se emplean en el desarrollo del software, incluyendo los lenguajes de programación, el entorno de desarrollo y las herramientas empleadas.

1.1 Sistema de comunicaciones

Un sistema de comunicación se describe como el conjunto de elementos que ordenadamente relacionados entre sí, tienen la capacidad de establecer la transmisión de un mensaje entre dos puntos independientes (5).

Los elementos fundamentales o indispensables que intervienen en el principio de comunicación son (5):

a) **Emisor o transmisor:** es el elemento que inicia la comunicación; es el encargado de transmitir el mensaje en un lenguaje que el receptor o receptores puedan descifrar con facilidad para poder establecer el enlace de comunicación.
b) **Medio o canal:** es el medio utilizado por el transmisor para hacer llegar el mensaje al receptor.
c) **Receptor:** es el elemento encargado de recibir el mensaje transmitido por el emisor a través de un medio. Al recibirse el mensaje se cumple el ciclo de la comunicación.

1.1.1 Comunicación satelital

En telecomunicaciones, comunicación satelital se refiere al uso de satélites artificiales para proporcionar enlaces de comunicación entre varios puntos de la tierra. La comunicación satelital juega un rol vital en los sistemas de comunicación globales. Aproximadamente 2000 satélites artificiales orbitan alrededor de la tierra, transmitiendo voz, video y datos desde y hacia varias localizaciones en todo el mundo (3). Las antenas utilizadas preferentemente en las comunicaciones vía

satélite son las antenas parabólicas, las cuales tienen forma de parábola y la peculiaridad de que las señales se reflejan e inciden sobre el foco de la parábola.

1.1.2 Satélite artificial

Un satélite artificial es un elemento físico capaz de recibir y transmitir señales en forma analógica o digital de alta calidad, está colocado en órbita por las necesidades que tiene el hombre para recibir y transmitir información a cualquier punto de la Tierra. La mayoría de los satélites de comunicación se colocan en el arco satelital; es decir, se encuentran en la órbita geosíncrona o geoestacionaria (5). A continuación, se describen algunos tipos de orbitas existentes:

- **Órbita baja terrestre:** normalmente se encuentra a menos de 2.000 km de altitud y puede llegar a estar a solamente 160 km sobre la Tierra. Los satélites en esta órbita viajan a aproximadamente 7,8 km por segundo. A esta velocidad, un satélite tarda aproximadamente 90 minutos para dar una vuelta completa a la Tierra (6).
- **Órbita polar:** Dentro de las órbitas bajas se encuentran las órbitas polares. Este tipo de órbita pasa aproximadamente por encima de los polos y su trayectoria forma un ángulo aproximadamente recto al cruzar el ecuador (6).
- **Órbita heliosíncrona:** Un tipo particular de órbitas polares son las heliosíncronas. La característica distintiva de estas órbitas es que su plano orbital sigue la dirección del Sol (como un girasol). Normalmente se encuentran a una altitud de 600-800 km. Estas órbitas son útiles en situaciones en que se requieren condiciones de iluminación constante del satélite o del elemento observado (telescopios que observan otros satélites o basura espacial, instrumentos ópticos que observan la superficie de la Tierra, satélites meteorológicos, etc.) (6).
- **Órbitas excéntricas:** Todas las órbitas terrestres descritas anteriormente son circulares. Las órbitas excéntricas se caracterizan por ser de forma elíptica con un extremo muy cercano a la Tierra, por ejemplo, entre 500 y 2000 km, y el otro extremo muy alejado, hasta 150000 km (la mitad de la distancia a la Luna). Estas órbitas se utilizan para misiones en las que

interesa alejarse de los efectos gravitatorios y electromagnéticos de la Tierra, como pueden ser telescopios de rayos X (6).

- **Órbita geoestacionaria:** Una órbita geoestacionaria, comúnmente llamada órbita GEO, es aquella que mantiene una posición fija con respecto a la superficie de la Tierra. La mecánica de Newton hace que esta órbita tenga una altura constante de 36.000 km y la posición estacionaria necesariamente sobrevuela la línea del ecuador. Los satélites que se encuentran en esta órbita parecen inmóviles en el cielo y debido a esto, la órbita geoestacionaria es ideal para los satélites de telecomunicaciones, ya que permiten disponer de un “repetidor” de señal siempre visible desde las mismas localizaciones terrestres (6).

1.1.3 Antena parabólica

Una antena parabólica es una antena que usa un reflector parabólico (una superficie curvada con forma de parábola) para dirigir las ondas de radio. La principal ventaja de estas es que son altamente direccionales, aunque esto supone que deben ser orientadas hacia el punto donde se encuentra en transmisor/receptor. Se usan principalmente para la comunicación punto a punto, la transmisión de señales de televisión y la comunicación vía satélite Las antenas parabólicas pueden ser transmisoras, receptoras o *full dúplex*, llamadas así cuando pueden transmitir y recibir simultáneamente (7).

Para establecer la comunicación, la antena debe estar orientada hacia el transmisor/receptor con el que se desea intercambiar los datos (7). Los parámetros que determinan la orientación de la antena son comúnmente denominados azimut y elevación (o altura).

El **azimut** se define como la distancia angular entre el punto cardinal norte y la proyección vertical del objetivo en el horizonte (8). Se puede interpretar como la cantidad de grados que se debe girar desde la dirección norte hasta estar frente al objetivo. En conjunto con la altura se usa para determinar la posición de los cuerpos en el cielo.

La **altura** puede tener diferentes significados según la ciencia en la que se analice. En astronomía se llama altura a la medida angular del lugar en el que se encuentra un cuerpo en el cielo por encima del horizonte del observador (8).

El azimut y la altura tienen la característica de ser dependientes de la posición del observador, es decir, para dos puntos diferentes tendrán diferente valor, aunque se mire hacia el mismo objeto. Por esta razón se hace necesario conocer la posición exacta del observador si se desea calcular estas medidas (9).

Para determinar el azimut y la elevación, es posible hacer uso de un Sistema de Referencia de Actitud y Rumbo, el cual cuenta con la capacidad de proporcionar orientación en tres dimensiones con gran precisión (10).

La tecnología que permite conocer la ubicación de un punto u objetivo con gran precisión es conocida como GNSS (*Global Navigation Satellite System*, Sistema Global de Navegación por Satélite) (11), la cual se describe a continuación.

1.2 Sistema Global de Navegación por Satélite

Sistema Global de Navegación por Satélites (*Global Navigation Satellite System*, GNSS) se refiere a una constelación de satélites que transmite señales desde el espacio con datos de posicionamiento y tiempo. Los receptores usan esta información para determinar la localización (11).

Algunos ejemplos de GNSS incluye el sistema europeo Galileo, el sistema estadounidense NAVSTAR *Global Positioning System* (GPS), el sistema ruso *Global'naya Navigatsionnaya Sputnikovaya* (GLONASS) y el sistema chino *BeiDou* (11).

Entre los GNSS disponibles en la actualidad se encuentra el norteamericano GPS que es controlado por el Departamento de Estado de los Estados Unidos. A continuación, se describe con más detalle este sistema.

1.2.1 Sistema de Posicionamiento Global

El Sistema de Posicionamiento Global (GPS por sus siglas en inglés) es un sistema de navegación basado en satélites. Funciona en cualquier condición meteorológica, en cualquier lugar del mundo, las 24 horas del día sin ningún costo. Originalmente

el servicio fue destinado para uso militar, pero a partir del año 1980 se hizo disponible para su uso civil (12).

Los satélites GPS dan la vuelta a la Tierra dos veces al día en una órbita precisa. Cada satélite transmite una señal única y sus parámetros orbitales, lo que permite a los dispositivos GPS decodificar y calcular la posición exacta del satélite (13).

Para calcular la posición en dos dimensiones (latitud y longitud) y rastrear el movimiento, un receptor GPS debe estar enlazado con al menos tres satélites. Con cuatro o más, es posible calcular la posición en tres dimensiones (latitud, longitud y altitud). Normalmente, dependiendo de las condiciones del clima y el lugar, un receptor puede rastrear una media de 8 satélites (14).

1.2.2 Receptor GPS

Los receptores GPS se encargan de detectar, decodificar y procesar las señales recibidas desde los satélites con el objetivo de determinar el punto donde se encuentran situados. Existen receptores de dos tipos: portátiles y fijos. Los portátiles pueden ser tan pequeños como para caber dentro de un teléfono móvil. Los fijos, en cambio, son los que se instalan normalmente en automóviles, embarcaciones, aviones o cualquier otro tipo de vehículo (15).

Los más sencillos están preparados para determinar con un margen mínimo de error la latitud, longitud y altura desde cualquier punto de la tierra. Otros más completos muestran también el punto donde hemos estado e incluso trazan de forma visual sobre un mapa la trayectoria seguida (15).

Su funcionamiento se basa en el principio matemático de la triangulación. Por tanto, para calcular la posición de un punto será necesario que el receptor GPS determine con exactitud la distancia que lo separa de al menos tres satélites (15).

1.3 Sistema de Referencia de Actitud y Rumbo

Un Sistema de Referencia de Actitud y Rumbo (AHRS por sus siglas en inglés) proporciona orientación en tres dimensiones integrando y fusionando los datos de giroscopios, acelerómetros y magnetómetros. La fusión de estos sensores garantiza

una precisión mucho mayor que otros sistemas como las Unidades de Medida Inercial (IMU por sus siglas en inglés) (10).

Para mejorar el rendimiento del AHRS es posible usar GNSS, lo que permite obtener mediciones más precisas en operaciones de lectura de larga duración. En este caso, la compañía Xsens ofrece el dispositivo MTi-G-710 (10).

1.3.1 Xsens MTi-G-710

El MTi-G-710 es un Sistema de Referencia de Actitud y Rumbo ayudado por GNSS que proporciona valores de alta calidad para la posición, velocidad, aceleración y orientación, incluso en ambientes desafiantes. Es capaz de medir con gran precisión la ubicación y orientación. El MTi-G-710 es desarrollado por Xsens Technologies, una compañía con veinte años de experiencia en el desarrollo de estos dispositivos (16).

Algunas de las características más destacadas son las siguientes (16)**:**

- Puede proporcionar datos de inclinación, cabeceo y dirección, además de la ubicación, con un solo dispositivo.
- Posee un diseño industrial.
- La carcaza protege los componentes internos ante fuertes golpes.
- Posee grado de protección IP67, por lo que es resistente al polvo y puede soportar una inmersión temporal de hasta un metro de profundidad.
- Se puede conectar por USB a un ordenador u otro dispositivo compatible, como la Raspberry Pi.

El kit de desarrollo incluye software que complementa el funcionamiento del MTi-G-710. Este incluye la aplicación MT Software Suite que permite la visualización de los datos transmitidos por el dispositivo, además de un conjunto de clases e interfaces para facilitar el desarrollo de aplicaciones personalizadas (16).

Por las características presentadas anteriormente, además de la disponibilidad existente en la UBID, se decide utilizar el dispositivo MTi-G-710 como el Sistema de Referencia de Actitud y Rumbo que brindara la información referente a la orientación y ubicación de las antenas parabólicas.

1.4 Sistemas de control de antenas parabólicas

En todo el mundo existe una gran cantidad de antenas dedicadas a la comunicación con los satélites, a las cuales se les puede acoplar un sistema de control que les permita auto orientarse en la dirección de mayor intensidad. Se realizó una búsqueda de estos sistemas encontrándose varias propuestas con diferentes características, las cuales se describen a continuación.

1.4.1 Sat-nms ACU

La compañía alemana SatService cuenta con un sistema de control y seguimiento de antena para satélites artificiales llamado sat-nms ACU (*Automatic Calling Unit*) que cuenta con tres módulos (17):

- Sat-nms ACU-ODM (*Outdoor module*)
- Sat-nms ACU-IDU (*Indoor module*)
- Sat-nms ACU19

Esta familia de productos puede cubrir el control de antenas que va desde aplicaciones de posicionado hasta sistemas de seguimiento.

Ilustración 1: Módulo externo sat-nms ACU (17)

Este sistema cuenta con un posicionador de antena de tres ejes con capacidad de seguimiento que proporciona todo tipo de interfaz para los controladores de tres motores, y es usado para medir los ángulos de azimut, elevación y polarización. Además, incluye un servidor web y provee una interfaz amigable al operador mediante un buscador web usando el protocolo Ethernet (17).

1.4.2 7134 Antenna Controller

El sistema de control de antenas **General Dinamics SATCOM Technologies (GDST) Model 7134** es un sistema basado en microprocesadores que incorpora tecnología digital para el posicionamiento preciso de antenas con alta fiabilidad, flexibilidad y una interfaz de usuario práctica y sencilla (18).

Ilustración 2: GD Satcom Technologies Model 7134 (18)

Este sistema de control está especialmente diseñado para aplicaciones de antenas de pequeña y mediana apretura que requieran capacidad para el acceso automático a múltiples satélites, control remoto, y opcionalmente, control de seguimiento de trayectoria automático. Entre las características principales se encuentran: lectura de posición y mensaje de reporte de falla en tiempo real, múltiples modos de control, autodiagnóstico interno y control remoto (18).

1.4.3 Sistema de automatización de la antena satelital de 4.5m de Jaruco

Este sistema, desarrollado en la Universidad Central de Las Villas como trabajo de diploma, permite el control automático de la antena satelital de 4.5m ubicada en Jaruco perteneciente a la Empresa de Telecomunicaciones de Cuba (ETECSA). Al diseñarse para permitir la automatización de una antena en específico, se desarrolló basado en un hardware que solo permite su uso en antenas similares, lo que dificulta su aplicación antenas con diferentes características.

Aunque este sistema soluciona el problema de la orientación manual de la antena, permitiendo controlar el proceso de orientación mediante una interfaz amigable y automatizada, tiene funcionalidad limitada ya que solo permite la entrada manual de los parámetros azimut y elevación, lo que provoca que aun sea necesario la presencia de personal altamente calificado en la estación para el control del sistema.

1.4.4 Necesidad de desarrollo endógeno

Como es posible apreciar, los sistemas **Sat-nms ACU** y **7134 Antenna Controller** poseen una configuración de tipo caja negra donde los algoritmos de control están empotrados en sistemas comerciales y se desconoce el funcionamiento de los mismos. En un tema tan importante para el país como es las telecomunicaciones,

su seguridad depende del correcto funcionamiento de sus sistemas. Haciendo uso de tecnología cubana se reducen las importaciones, pero, sobre todo, se garantiza la independencia tecnológica y se contribuye a la soberanía del país.

En el caso del sistema desarrollado en la Universidad Central de Las Villas, al estar concebido para una antena en específico, hace imposible su uso en otras con características diferentes, lo que deja abierta la necesidad de un sistema que permita el control del proceso de orientación sin importar la posición o las características de la antena.

1.5 Protocolos de comunicación

Un protocolo de comunicación define un conjunto especifico de normas y reglas de transmisión que permiten ponerse de acuerdo a los equipos de comunicación acerca de cómo debe realizarse la comunicación a través de un canal determinado. En las redes de computadoras, algunos de los protocolos más usados son UDP y TCP.

El **Protocolo de Datagramas de Usuario** (UDP, por sus siglas en inglés) es un protocolo no orientado a conexión. Es decir, cuando un dispositivo A envía paquetes a un dispositivo B el flujo es unidireccional. La transferencia de datos se realiza sin prevenir al destinatario, y el destinatario recibe los datos sin enviar una confirmación al emisor (19).

Contrariamente a UDP, el **Protocolo de Control de Transmisión** (TCP, por sus siglas en inglés) está orientado a conexión, lo que significa que se establece una conexión y se mantiene hasta que las aplicaciones terminen de intercambiar mensajes. Cuando un dispositivo A envía datos a un dispositivo B, el segundo es informado de la llegada de los datos, y confirma su recepción. Aquí intervine el control de verificación por redundancia cíclica (CRC por sus siglas en inglés) de datos que se basa en una ecuación matemática que permite verificar la integridad de los datos transmitidos. De este modo si los datos recibidos son corruptos, el protocolo TCP permite que los destinatarios soliciten al emisor que se vuelva a enviar los datos (20).

1.6 Sistemas embebidos

Un sistema embebido es un dispositivo basado en un microprocesador que está diseñado para realizar una o pocas funciones dedicadas y no para ser programado por el usuario final como una PC (21). Algunos ejemplos de sistemas embebidos son: un vehículo que posea microcontroladores para controlar los frenos, las bolsas de aire o las ventanillas eléctricas, una lavadora que controla el ciclo de lavado e incluso muestra información sobre el mismo.

Estos sistemas de procesamiento que integran hardware, software y comunicaciones son capaces de dotar a los objetos en los que están integrados de capacidades como confiabilidad, inteligencia, conectividad, gestión energética, interacción con el entorno y productividad (21).

1.6.1 Microcontroladores

Un microcontrolador es un circuito integrado digital que puede ser usado para muy diversos propósitos debido a que es programable. Está compuesto por una unidad central de proceso (CPU), memorias (ROM y RAM) y líneas de entrada y salida (periféricos) (22).

El microcontrolador **Arduino** es una plataforma de hardware libre basada en un hardware y software de fácil uso. Las placas Arduino son capaces de leer una entrada (luz en un sensor, o la acción de un botón) y convertirla en una salida (hacer girar un motor, o encender un LED). Para lograr esto se debe enviar un conjunto de instrucciones al microcontrolador presente en la placa (23).

1.6.2 Computadora de placa única

Una computadora de placa única (o SBC por sus siglas en inglés) son placas que contienen todos o la mayor parte de los componentes de un ordenador. La principal característica de los SBC son sus reducidas dimensiones, además de ser muy económicas. Normalmente estas placas no suelen sobrepasar los 100 dólares (24).

Ejemplos de placas SBC (24)**:**

- **Raspberry Pi:** Es la placa SBC más popular. Es una pequeña placa que cuenta con varias versiones y posee una amplia comunidad. Actualmente

gracias a su comunidad se puede hacer prácticamente cualquier cosa con esta placa, desde un servidor hasta una pesada Tablet.

- **BeagleBone Black:** Es la alternativa estadounidense a Raspberry Pi. Por lo general no suele existir mucha diferencia entre la potencia de esta placa con el resto.
- **PcDuino:** PcDuino está basado en los esquemas de Arduino e incorpora lo necesario para ser una placa SBC, es decir: procesador y memoria RAM. A diferencia del resto, PcDuino es bastante grande, alcanza los 12 cm de largo por 6 cm de ancho. El último modelo de esta placa admite y soporta Ubuntu y Android.

1.6.3 Plataformas electrónicas empleadas en la Unidad Básica de Investigación y Desarrollo (UBID)

Las principales plataformas para el desarrollo de sistemas embebidos en la UBID son Arduino y Raspberry Pi. Ambas pueden ser empleadas para dar solución a problemas similares, pero la segunda destaca por ser una microcomputadora completamente funcional y tener un mayor rendimiento. En la tabla 1 se pueden apreciar las principales características de estas.

	Raspberry Pi 3	**Arduino**
Frecuencia del reloj	Quad Core 1.2 GHz	16 MHz
Memoria	1 GB	2-8 KB
Red	Ethernet, Wifi y Bluetooth	No
Sistema operativo	Raspbian	No
Multitarea	Si	No
Precio (en dólar)	~$45	~$25

Tabla 1: Comparación entre Raspberry Pi y Arduino.

Una vez realizada la comparación se decide utilizar Raspberry Pi debido a que esta posee mejor rendimiento, así como la capacidad de comunicación mediante Ethernet, Wifi y Bluetooth. Posee cuatro puertos USB, lo que la hace compatible con el dispositivo MTi-G-710 y al usar un sistema operativo basado en Linux permite la utilización de las clases e interfaces presentes en el SDK de dicho dispositivo.

Además, Raspberry Pi es de hardware y código abierto por lo que cumple con las políticas se software libre y soberanía tecnológica del país.

1.6.4 Raspberry Pi 3

La plataforma seleccionada para el desarrollo de la propuesta de solución es la Raspberry Pi Model B. Esta pequeña placa de 85 x 54 milímetros aloja un chip Broadcom BCM2837 con procesador ARMv8 de 64 bits de cuatro núcleos a 1.2GHz y 1Gb de RAM. Además, dispone de conectividad Wifi, Bluetooth, Ethernet y cuatro puertos USB 2.0, además de 40 pines GPIO (General Purpose Input/Output, Entrada/Salida de propósito general).

Como periféricos de salida cuenta con un conector DSI para la conexión de una pantalla táctil compatible, puerto de salida de video HDMI y un mini Jack compuesto que exporta audio y video.

Para la alimentación posee un puerto micro USB que actúa como fuente de energía, compatible con la mayoría de los cargadores de teléfonos móviles actuales.

Raspberry Pi puede considerarse como una computadora completamente funcional, aunque no logra alcanzar el rendimiento de una PC de escritorio debido a sus características inferiores. En cuanto a su precio, suele estar por debajo de los cuarenta dólares (25).

1.7 Tecnologías y herramientas para el desarrollo de la solución

Este epígrafe hace referencia a las tecnologías de comunicación, lenguaje y herramienta de modelado, lenguajes de programación y metodología de desarrollo a utilizar en el desarrollo de la solución propuesta.

1.7.1 Tecnologías de comunicación

Actualmente la conexión entre dispositivos para el intercambio de datos se puede establecer de manera alámbrica e inalámbrica. En este caso la comunicación con la aplicación será posible tanto vía Ethernet como Wifi.

El instituto de Ingenieros Eléctricos y Electrónicos (IEEE por sus siglas en inglés) define **Ethernet** como el protocolo 802.3. Ethernet define las características del cableado y señalización de nivel físico y los formatos de tramas de datos del nivel

de datos del modelo OSI. Es un estándar de redes que emplea el método CSMA/CD (Acceso Múltiple por Detección de Portadora con Detector de Colisiones) y se emplea en la instalación de redes de área local. Puede alcanzar velocidades de 10/100/1000 megabits por segundo y los tipos de cables más comunes son categoría 5 (o CAT5) que se conectan al dispositivo y/o enrutador mediante el conector RJ45 (26).

Wifi es una tecnología de comunicación inalámbrica que permite conectar a internet equipos electrónicos, como computadoras, tablets y teléfonos inteligentes mediante el uso de radiofrecuencias o infrarrojos para la transmisión de la información.

La tecnología Wifi es una solución que comprende un conjunto de estándares para redes basados en las especificaciones IEEE 802.11, lo cual asegura la compatibilidad e interoperabilidad de los dispositivos certificados bajo esta denominación.

Para su funcionamiento se necesita de un enrutador dotado de una antena que distribuye la señal de manera inalámbrica dentro de un radio determinado. Mientras más cerca se encuentren los equipos de la fuente de la señal, mejor será la conexión (27).

1.7.2 Lenguaje Unificado de Modelado (UML) v8.0

UML es un lenguaje que proporciona un vocabulario y unas reglas que se centran en la representación gráfica de un sistema. Este lenguaje nos indica cómo crear y leer los modelos (28). Los objetivos de UML se pueden sintetizar en sus funciones:

- **Visualizar:** permite expresar de una forma gráfica un sistema de manera que otro lo puede entender.
- **Especificar:** permite especificar cuáles son las características de un sistema antes de su construcción.
- **Construir:** a partir de los modelos especificados se pueden construir los sistemas diseñados.
- **Documentar:** Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden ser útiles para su futura revisión.

En UML, un diagrama es la representación gráfica de un conjunto de elementos con sus relaciones. En concreto, un diagrama ofrece una vista del sistema a modelar. UML incluye los siguientes diagramas:

- Diagrama de casos de uso
- Diagrama de clases
- Diagrama de objetos
- Diagrama de secuencia
- Diagrama de colaboración
- Diagrama de estados
- Diagrama de actividades
- Diagrama de componentes
- Diagrama de despliegue

1.7.3 Visual Paradigm para UML 8.0

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) permiten una rápida construcción de aplicaciones de mejor calidad y a un costo inferior. Visual Paradigm es una herramienta CASE profesional que soporta el ciclo de vida completo del desarrollo de software. Dentro de sus funcionalidades se encuentran:

- Diseño de diagramas UML
- Diseño de Bases de Datos, incluyendo Diagrama Entidad Relación.
- Cientos de plantillas para diagramas
- Construcción de Diagramas de Procesos de Negocio.
- Distribución automática de diagramas, reorganización de las figuras y conectores de los diagramas UML (29).

1.7.4 Lenguajes de programación

Un lenguaje de programación es un conjunto de comandos, instrucciones y sintaxis que se usan en la creación de software. Usualmente los lenguajes empleados por los programadores son conocidos como lenguajes de alto nivel. Este código puede

ser compilado a un lenguaje de bajo nivel que puede ser reconocido directamente por el hardware de la computadora.

Ejemplos de lenguajes de alto nivel incluyen C++, Java, Perl y PHP. Los lenguajes como C++ o Java se conocen como “lenguajes compilados” ya que deben ser procesados por un compilador antes de ser ejecutados. Perl y PHP son conocidos como “lenguajes interpretados” ya que se pueden ejecutar a través de un intérprete sin necesidad de compilar el código (30).

1.7.5 Lenguaje de programación C++

C++ es un lenguaje de programación orientado a objetos de propósito general, desarrollado por Bjarne Stroustrup, y es una extensión del lenguaje C. Se considera que C++ es un lenguaje de nivel medio ya que encapsula características de lenguajes de alto y bajo nivel.

C++ es uno de los lenguajes más populares y se utiliza principalmente para el desarrollo de aplicaciones, controladores, aplicaciones cliente-servidor y sistemas embebidos. Los conceptos principales de la programación en C++ incluyen polimorfismo, plantillas, espacios de nombre y punteros (30).

1.7.6 *Framework* Qt

En los sistemas informáticos, un *framework* es una estructura que indica qué tipo de programas pueden o deben ser construidos y cómo se interrelacionan. Algunos marcos de trabajo de sistemas informáticos también incluyen programas reales, especifican interfaces de programación u ofrecen herramientas de programación para usar los marcos (31).

Qt es un *framework* de desarrollo de aplicaciones multiplataforma para escritorio, móviles y sistemas embebidos. Las plataformas que soporta incluyen Linux, OS X, Windows y Android. El *framework* Qt está escrito en C++ e incluye un preprocesador conocido como Compilador de Meta-objetos (*Meta-Object Compiler*, MOC) que extiende la funcionalidad de C++ con características como señales y slots.

Qt es desarrollado y mantenido por “The Qt Company” en conjunto con la comunidad. Está disponible bajo varias licencias, “The Qt Company” vende licencias

comerciales, pero además se encuentra disponible como software libre bajo distintas versiones de GPL y LGPL (32).

1.7.7 Sistema operativo

Un sistema operativo es el conjunto de programas informáticos que permite la administración eficaz de los recursos de un dispositivo. Estos programas comienzan a trabajar apenas enciende el equipo, ya que gestionan el hardware desde los niveles más básicos y permiten además la interacción con el usuario (33).

Raspberry Pi soporta cualquier sistema operativo capaz de funcionar sobre la arquitectura ARM, incluyendo Android, Linux o Windows. Los más destacados son Windows IoT Core, Ubuntu y Raspbian, siendo este último el recomendado por los creadores (34).

1.7.8 Raspbian

Raspbian es un sistema operativo libre basado en Debian optimizado para el hardware de la Raspberry Pi. El mismo provee además de un sistema operativo puro alrededor de 35,000 paquetes y software precompilado para una fácil instalación en el dispositivo (35).

Raspbian trae preinstalado software para la educación, programación y uso general. Incluye Python, Scratch, Sonic Pi y Java.

1.8 Metodología de desarrollo

Una metodología de desarrollo de software tiene como objetivo aumentar la calidad del software que se produce, haciendo énfasis en torno a la idea de producir con la mayor calidad en el menor tiempo posible (36).

La Empresa de Tecnologías de la Información para la Defensa cuenta con la metodología **Prodesoft** (Proceso de Desarrollo y Gestión de Software), la cual tiene como meta la producción de software más robusto, predecible, reutilizable y de fácil mantenimiento. Esta define un ciclo de vida que está compuesto por cinco fases: inicio, modelación, construcción, explotación experimental y despliegue (37).

Prodesoft se basa en un enfoque iterativo-incremental, donde cada iteración comprende la modelación de procesos del negocio, definición de requisitos, diseño de la arquitectura, diseño detallado, implementación y pruebas (37).

La UBID es un centro que se dedica principalmente a la asimilación y desarrollo de nuevas tecnologías, por lo que en ocasiones los proyectos están encaminados a desarrollar soluciones que se puedan aplicar en distintos escenarios, razón por la cual no se realiza el modelado de los procesos del negocio. Esto provoca que la metodología Prodesoft no sea la más indicada para guiar el proceso de desarrollo.

La metodología recomendada para el desarrollo en la Universidad de las Ciencias Informáticas es Proceso Unificado Ágil para la Universidad de las Ciencias Informáticas (AUP-UCI).

AUP-UCI es una variación de la metodología AUP de forma que se adapte al ciclo de vida que se define para la actividad productiva de la UCI. Esta metodología persigue como objetivo mejorar la calidad del software que se produce, para lo que se apoya en el modelo de calidad CMMI-DEV v1.3, el cual constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora (36).

Contiene cuatro escenarios para modelar el sistema en los proyectos, este trabajo se centrará en el escenario 4.

Escenario No 4: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos, ya que una historia de usuario no debe poseer demasiada información (36).

La metodología AUP-UCI comprende tres fases que se describen a continuación (36):

- **Inicio:** Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información

fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.

- **Ejecución:** En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del software.
- **Cierre:** En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Además**,** propone siete disciplinas para el ciclo de vida de los proyectos (36):

1. **Modelado de negocio:** El Modelado del Negocio es la disciplina destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito.
2. **Requisitos:** El esfuerzo principal en la disciplina requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto.
3. **Análisis y diseño:** En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Además, en esta disciplina se modela el sistema y su forma para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Los modelos desarrollados son más formales y específicos.
4. **Implementación:** En la implementación, a partir de los resultados del Análisis y Diseño se construye el sistema.

5. **Pruebas internas:** En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas.
6. **Pruebas de liberación:** Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.
7. **Pruebas de aceptación:** Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.

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.
- En la UBID no se realiza modelado de negocio.

1.9 Conclusiones parciales

Durante el desarrollo de este capítulo se realizó un análisis de los conceptos referentes a la comunicación satelital lo que posibilitó una mejor comprensión del marco teórico de la investigación. El estudio de soluciones similares permitió corroborar la necesidad del desarrollo de la solución que se propone, así como obtener una guía de las características y funcionalidades que se deben incluir en el sistema. Se definió el dispositivo Xsens MTi-G-710 como el Sistema de Referencia de Actitud y Rumbo a utilizar en el desarrollo de la solución debido a su disponibilidad y características. Se decidió emplear Raspberry Pi 3 como hardware para controlar el proceso de orientación y ofrecer información acerca del mismo, C++ como lenguaje de programación y Qt como marco de trabajo. La metodología

que guiará el proceso de desarrollo será AUP-UCI atendiendo a las características del proyecto y la flexibilidad y documentación que brinda la misma.

Capítulo 2: Análisis y diseño de la solución propuesta

En el presente capítulo se expone la propuesta de solución para el sistema de geoposicionamiento integral para el control del proceso de orientación de antenas parabólicas de grandes dimensiones, utilizando la metodología de desarrollo AUP-UCI en el escenario 4. Este capítulo se centrará en el análisis y representación de las disciplinas: requisitos, análisis y diseño.

2.1 Análisis de la solución

Los parámetros que controlan la orientación de las antenas parabólicas de grandes dimensiones son el azimut y la elevación. La solución debe ser capaz de capturar los datos necesarios para controlar el proceso de orientación y mostrarlos al usuario. Además, el sistema debe ser lo más móvil posible, para lo cual funcionará sobre un dispositivo Raspberry Pi, y se hará uso de una brújula y GPS para capturar todos los datos requeridos. La solución debe permitir el control de los movimientos de la antena mediante la interfaz Ethernet o Wifi y permitir la visualización en todo momento los parámetros relativos a la ubicación de la antena. El sistema permitirá al usuario escoger el tipo de movimiento, los tipos posibles son:

- Manual: permite el control manual de la orientación tanto en el eje horizontal como vertical.
- Automático: permite el control automático de la orientación, esto es, dado el punto objetivo, el sistema orientara la antena hacia el mismo de manera automática.

Para controlar el proceso de orientación mediante Wifi o Ethernet se define un protocolo de comunicación sencillo que el cliente tendrá que implementar, lo que permite el envío de comandos y el intercambio de información entre el cliente y la antena. El protocolo definido se describe en la próxima sección.

2.2 Protocolo de comunicación

El protocolo empleado por la antena para la comunicación con el cliente consta de dos posibles variantes, que permiten el envío de comandos y datos para la configuración del dispositivo.

Para el envío de comandos se hará uso de una cadena de la forma '*$<comando>;*' donde *comando* es una letra en mayúscula que hace referencia a alguna de las siguientes opciones:

Comando	**Respuesta**
I	Movimiento horizontal de la antena en sentido contrario a las agujas del reloj
D	Movimiento horizontal de la antena en el sentido de las agujas del reloj
S	Detener movimiento horizontal
A	Movimiento vertical de la antena hacia arriba
B	Movimiento vertical de la antena hacia abajo
E	Detener movimiento vertical
T	Comenzar movimiento automático hacia el objetivo establecido.
Cualquier otra	Detener movimiento vertical y horizontal.

Tabla 2: Comandos para el control de la antena.

El envío de los datos para la configuración de la orientación automática hace uso de una cadena que comienza por el carácter '**#**' y podrá tener las siguientes estructuras:

- **#<azimut>:<elevación>;** para el envío del azimut y elevación que se desea se posicione la antena. Se debe reemplazar los parámetros <*azimut*> y <*elevación*> por los valores deseados.
- **##<latitud>:<longitud>;** para el envío de las coordenadas si se desea que el sistema calcule el azimut y la elevación basado en la posición actual de la antena. Se debe reemplazar los parámetros <*latitud*> y <*longitud*> por los valores deseados.

En caso de recibir una cadena inválida el sistema ignora esa petición, sin enviar ningún tipo de retroalimentación al cliente, por lo que es responsabilidad del mismo validar los datos antes del envío.

2.3 Requisitos del sistema

Los requisitos de un sistema son la descripción de los servicios proporcionados por el mismo y sus restricciones operativas. Estos reflejan las necesidades de los clientes para que el sistema ayude a resolver un problema o conseguir un objetivo determinado. Los requisitos de un sistema pueden dividirse en requisitos funcionales y no funcionales.

2.3.1 Requisitos funcionales del sistema

Los requisitos funcionales son enunciados acerca de los servicios que el sistema debe brindar, de cómo debería reaccionar a entradas particulares y de cómo debe comportarse en situaciones específicas. En algunos casos, los requisitos funcionales también explican lo que el sistema no debe hacer (38).

No	Nombre	Descripción	Prioridad	Complejidad
RF1	Inicializar dispositivo.	El sistema al arrancar debe comprobar que la brújula esté conectada y accesible.	Alta	Media
RF2	Configurar dispositivo.	El sistema al arrancar debe configurar la brújula para que envíe los datos de orientación y ubicación.	Alta	Alta
RF3	Iniciar lectura.	El sistema al arrancar, luego de configurar la brújula, debe comenzar la lectura de los datos.	Alta	Baja
RF4	Leer datos de orientación.	El sistema al recibir datos de orientación desde la brújula debe procesarlos y mostrarlos.	Alta	Alta
RF5	Leer datos de ubicación.	El sistema al recibir datos de ubicación desde la brújula debe procesarlos y mostrarlos.	Alta	Alta
RF6	Rotar antena en	El sistema debe ser capaz de	Alta	Media

	el sentido de las manecillas del reloj.	rotar la antena en sentido de las manecillas reloj al recibir un comando enviado por el usuario.		
RF7	Rotar antena en sentido contrario a las manecillas del reloj.	El sistema debe ser capaz de rotar la antena en sentido contrario a las manecillas reloj al recibir un comando enviado por el usuario.	Alta	Media
RF8	Detener rotación horizontal.	El sistema debe ser capaz de detener la rotación al recibir un comando enviado por el usuario.	Alta	Media
RF9	Mover antena hacia arriba.	El sistema debe ser capaz de aumentar la elevación al recibir un comando enviado por el usuario.	Alta	Media
RF10	Mover antena hacia abajo.	El sistema debe ser capaz de disminuir la elevación al recibir un comando enviado por el usuario.	Alta	Media
RF11	Detener movimiento vertical.	El sistema debe ser capaz de detener el movimiento vertical al recibir un comando enviado por el usuario.	Alta	Media
RF12	Recibir azimut y elevación.	El sistema debe permitir al usuario definir el azimut y elevación hacia donde se desea orientar la antena.	Media	Alta
RF13	Calcular azimut y elevación dado	El sistema debe permitir calcular el azimut y la elevación hacia	Media	Alta

	ubicación.	donde se desea orientar la antena dadas las coordenadas del objetivo.		
RF14	Orientar antena automáticamente.	El sistema debe ser capaz de orientarse automáticamente hacia el objetivo definido.	Media	Alta

Tabla 3: Requisitos funcionales del sistema.

2.3.2 Requisitos no funcionales del sistema

Los requisitos no funcionales son limitaciones sobre los servicios o funciones que ofrece el sistema. Estos incluyen restricciones como fiabilidad, respuesta en el tiempo y capacidad de almacenamiento. Los requisitos no funcionales se suelen aplicar al sistema como un todo y surgen de las necesidades del cliente, ya sea debido a las restricciones de presupuesto o las políticas de la organización (38).

Requerimientos de Software

- El sistema debe funcionar sobre el Sistema Operativo Raspbian, en su versión Stretch.

Requerimientos de Hardware

- Memoria RAM: 1 GB.
- Procesador: ARM QuadCore a 1.2Ghz o superior.
- 1 puerto USB 2.0
- Conectividad Wifi y/o Ethernet
- Funcionar sobre Raspberry Pi

Restricciones en el diseño y la implementación

- Se implementará usando el lenguaje de programación C++, en su versión 2011.
- Se utilizará el IDE Qt Creator en su versión 4.6.
- Se empleará como marco de trabajo Qt, en su versión 5.11.

Requerimientos de usabilidad

- El sistema debe proporcionar una interfaz gráfica sencilla, intuitiva y fácil de entender.

2.3.3 Historias de usuario

Es una técnica que se utiliza para especificar los requisitos del software. Son tarjetas donde el cliente describe brevemente las características que el sistema debe poseer, ya sean funcionales o no funcionales. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas.

<table>
<tr><td>Número: 1</td><td colspan="2">Nombre del requisito: Inicializar dispositivo</td></tr>
<tr><td colspan="2">Programador: David Rodríguez Feitó</td><td>Iteración Asignada: 1</td></tr>
<tr><td colspan="2">Prioridad: Alta</td><td>Tiempo estimado: 24 horas</td></tr>
<tr><td colspan="2">Nivel de complejidad: Media</td><td>Tiempo real: 24 horas</td></tr>
<tr><td colspan="3">Descripción:

1. Objetivo
Detectar si el dispositivo MTi-G-710 está conectado e iniciar la comunicación.
2. Acciones para lograr el objetivo (precondiciones y datos):
- El dispositivo debe conectarse al sistema.</td></tr>
<tr><td colspan="3">Observaciones: No aplica</td></tr>
<tr><td colspan="3">Prototipo de interfaz:</td></tr>
</table>

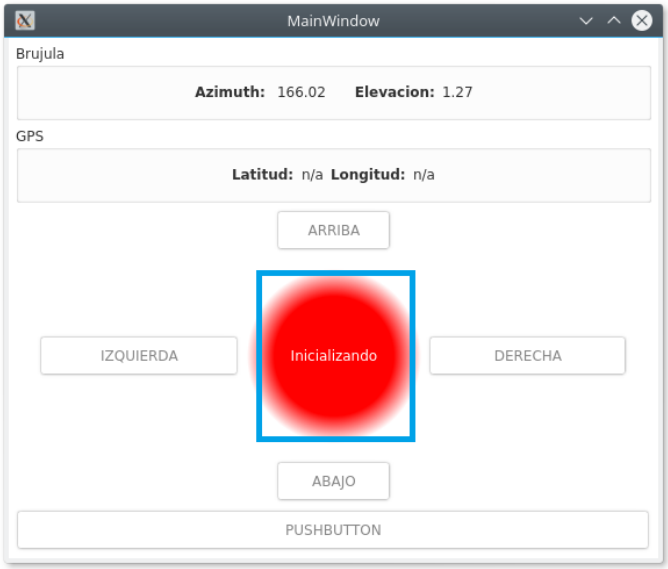


Tabla 4: Historia de usuario No.1

Número: 2	**Nombre del requisito:** Configurar dispositivo
Programador: David Rodríguez Feitó	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 40 horas
Nivel de complejidad: Alta	**Tiempo real:** 38 horas
Descripción: 1. **Objetivo:** Configurar el dispositivo MTi-G-710 para definir los datos que se desean recibir y el formato de los mismos. 2. **Acciones para lograr el objetivo (precondiciones y datos):** - El dispositivo MTi-G-710 debe conectarse al sistema. - El dispositivo debe estar inicializado.	
Observaciones: No aplica	
Prototipo de interfaz:	

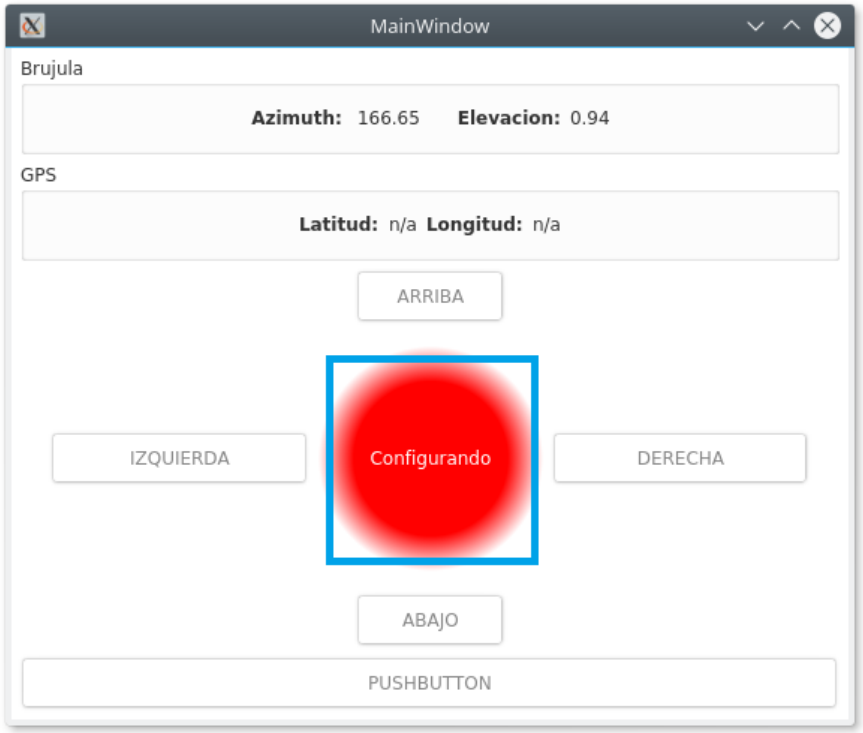


Tabla 5: Historia de usuario No.2

Número: 3	**Nombre del requisito:** Iniciar lectura
Programador: David Rodríguez Feitó	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 16 horas
Nivel de complejidad: Baja	**Tiempo real:** 16 horas
Descripción: 1. **Objetivo:** Poner el dispositivo MTi-G-710 en modo lectura y comenzar la captura de datos. 2. **Acciones para lograr el objetivo (precondiciones y datos):** - El dispositivo MTi-G-710 debe conectarse al sistema. - El dispositivo debe estar configurado.	
Observaciones: No aplica	
Prototipo de interfaz:	

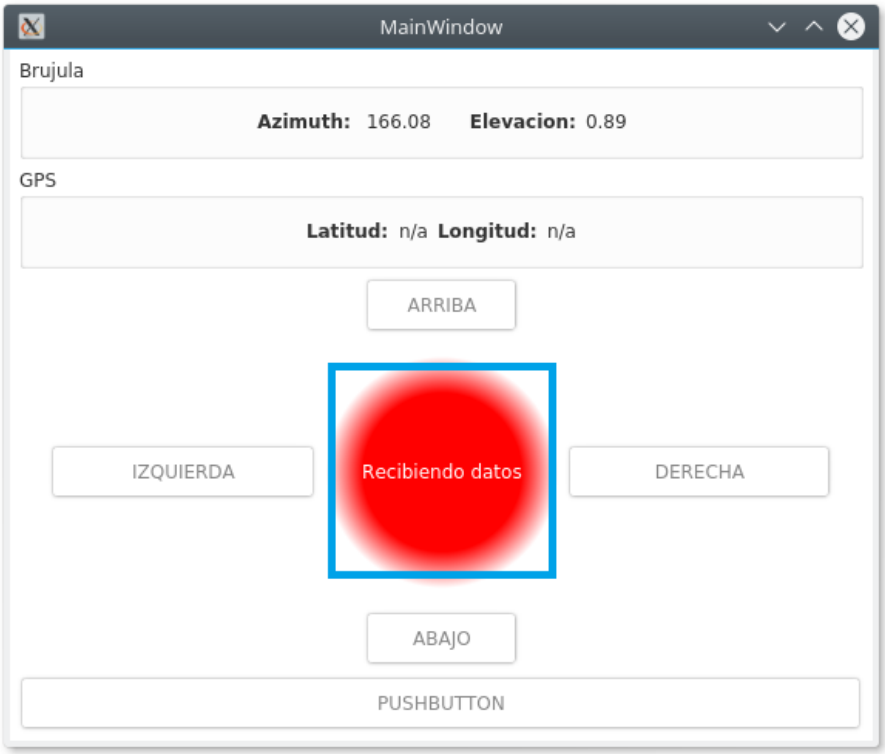


Tabla 6: Historia de usuario No.3

Número: 4	**Nombre del requisito:** Leer datos de orientación
Programador: David Rodríguez Feitó	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 20 horas
Nivel de complejidad: Alta	**Tiempo real:** 20 horas
Descripción: 1. **Objetivo:** Recibir, procesar y mostrar los datos de orientación enviados por el dispositivo MTi-G-710. 2. **Acciones para lograr el objetivo (precondiciones y datos):** - El dispositivo MTi-G-710 debe conectarse al sistema. - El dispositivo debe estar configurado.	
Observaciones: No aplica	

Prototipo de interfaz:

Tabla 7: Historia de usuario No.4

Número: 5	**Nombre del requisito:** Leer datos de ubicación
Programador: David Rodríguez Feitó	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo estimado:** 20 horas
Nivel de complejidad: Alta	**Tiempo real:** 20 horas

Descripción:

1. **Objetivo:**
   Recibir, procesar y mostrar los datos de ubicación enviados por el dispositivo MTi-G-710.
2. **Acciones para lograr el objetivo (precondiciones y datos):**
   - El dispositivo MTi-G-710 debe conectarse al sistema.
   - El dispositivo debe estar configurado.

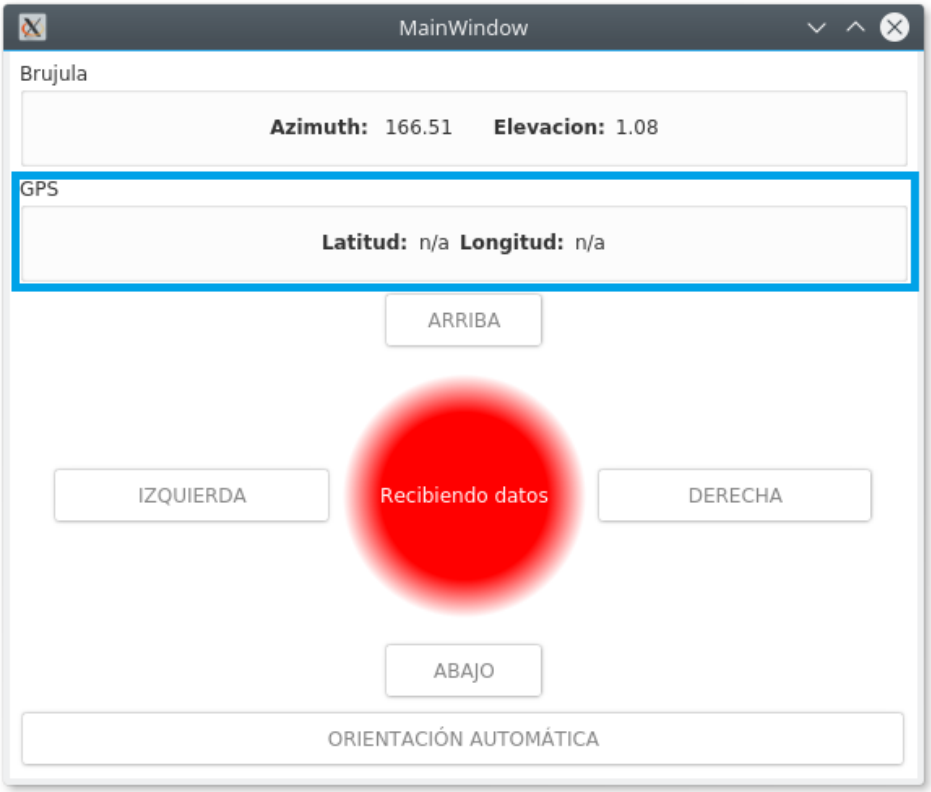
Observaciones: No aplica
Prototipo de interfaz: 

Tabla 8: Historia de usuario No.5

Número: 6	**Nombre del requisito:** Rotar la antena en el sentido de las manecillas del reloj
Programador: David Rodríguez Feitó	**Iteración Asignada:** 2
Prioridad: Alta	**Tiempo estimado:** 20 horas
Nivel de complejidad: Media	**Tiempo real:** 20 horas
Descripción: 1. **Objetivo:** Iniciar la rotación de la antena hacia la derecha.	

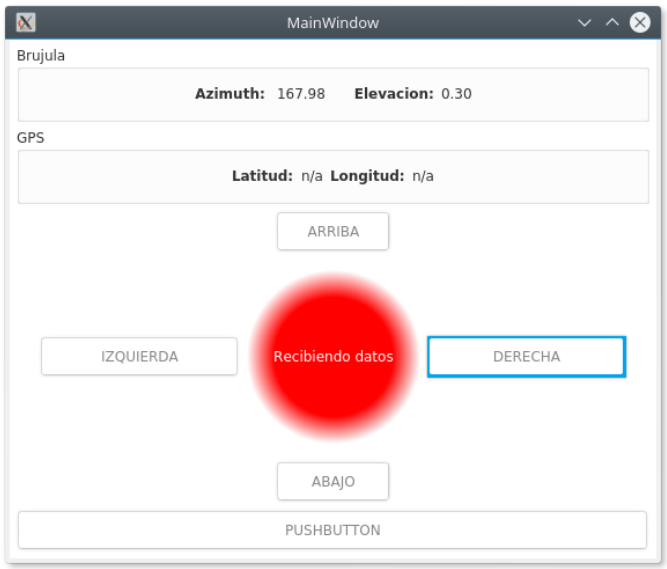
2. **Acciones para lograr el objetivo (precondiciones y datos):** - El usuario debe conectarse al sistema mediante Wifi o Ethernet. - El usuario debe enviar el comando “Rotar a la derecha”.
Observaciones: • La Raspberry Pi debe tener comunicación mediante Wifi con el dispositivo Arduino que controla los motores.
Prototipo de Interfaz: 

Tabla 9: Historia de usuario No.6

Número: 7	**Nombre del requisito:** Rotar la antena en sentido contrario a las manecillas del reloj
Programador: David Rodríguez Feitó	**Iteración Asignada:** 2
Prioridad: Alta	**Tiempo estimado:** 20 horas
Nivel de complejidad: Media	**Tiempo real:** 20 horas
Descripción:	

1. **Objetivo:**

   Iniciar la rotación de la antena hacia la izquierda

2. **Acciones para lograr el objetivo (precondiciones y datos):**

   - El usuario debe conectarse al sistema mediante Wifi o Ethernet.

   - El usuario debe enviar el comando “Rotar a la izquierda”.

Observaciones:

- La Raspberry Pi debe tener comunicación mediante Wifi con el dispositivo Arduino que controla los motores.

Prototipo de interfaz:

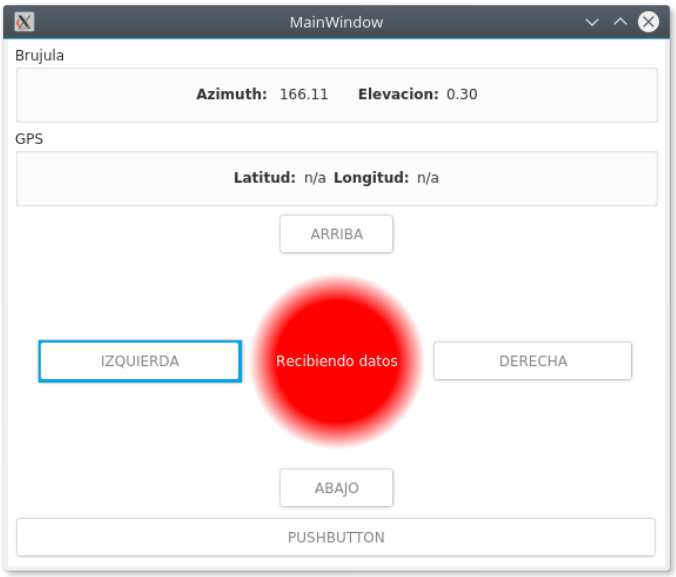


Tabla 10: Historia de usuario No.7

Número: 8	**Nombre del requisito:** Detener rotación horizontal
Programador: David Rodríguez Feitó	**Iteración Asignada:** 2
Prioridad: Alta	**Tiempo estimado:** 20 horas
Nivel de complejidad: Media	**Tiempo real:** 20 horas
Descripción:	

1. **Objetivo:**

   Detener la rotación de la antena.

2. **Acciones para lograr el objetivo (precondiciones y datos):**

   - El usuario debe conectarse al sistema mediante Wifi o Ethernet.

   - El usuario debe enviar el comando “Detener rotación”.

Observaciones:

- La Raspberry Pi debe tener comunicación mediante Wifi con el dispositivo Arduino que controla los motores.

Prototipo de interfaz:

Tabla 11: Historia de usuario No.8

Número: 9	**Nombre del requisito:** Mover antena hacia arriba
Programador: David Rodríguez Feitó	**Iteración Asignada:** 2
Prioridad: Alta	**Tiempo estimado:** 20 horas
Nivel de complejidad: Media	**Tiempo real:** 20 horas

Descripción:

1. **Objetivo:**
   Iniciar el movimiento vertical de la antena hacia arriba.
2. **Acciones para lograr el objetivo (precondiciones y datos):**
   - El usuario debe conectarse al sistema mediante Wifi o Ethernet.
   - El usuario debe enviar el comando “Aumentar elevación”.

Observaciones:

- La Raspberry Pi debe tener comunicación mediante Wifi con el dispositivo Arduino que controla los motores.

Prototipo de Interfaz:

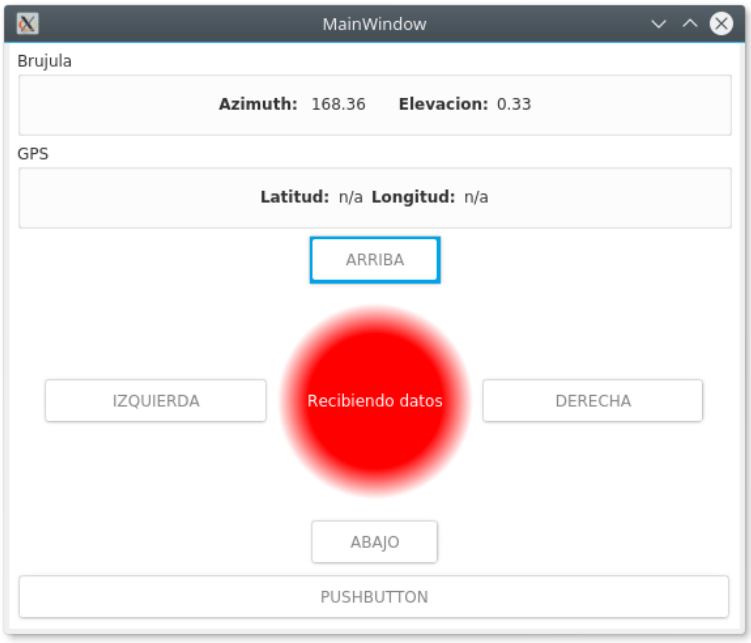


Tabla 12: Historia de usuario No.9

Número: 10	**Nombre del requisito:** Mover antena hacia abajo
Programador: David Rodríguez Feitó	**Iteración Asignada:** 2
Prioridad: Alta	**Tiempo estimado:** 20 horas

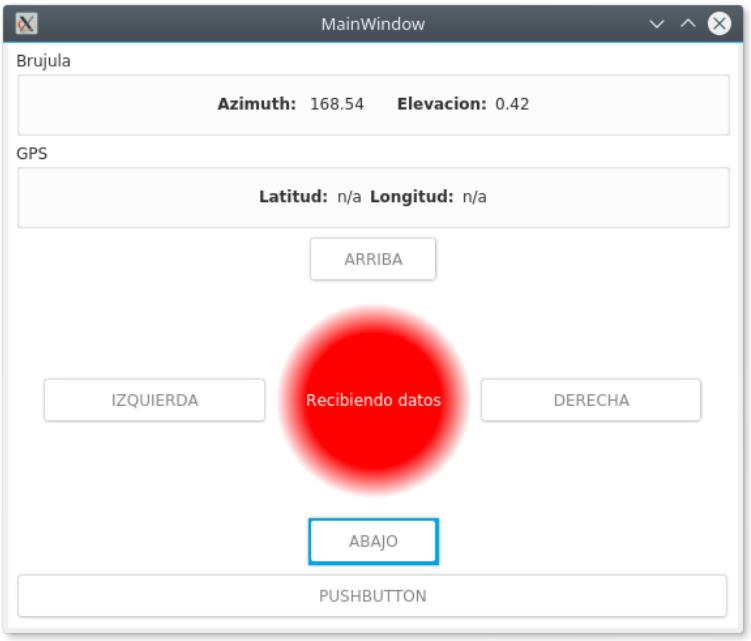
Nivel de complejidad: Media	**Tiempo real:** 20 horas
Descripción: 1. **Objetivo:** Iniciar el movimiento vertical de la antena hacia abajo. 2. **Acciones para lograr el objetivo (precondiciones y datos):** - El usuario debe conectarse al sistema mediante Wifi o Ethernet. - El usuario debe enviar el comando “Disminuir elevación”.	
Observaciones: • La Raspberry Pi debe tener comunicación mediante Wifi con el dispositivo Arduino que controla los motores.	
Prototipo de interfaz: 	

Tabla 13: Historia de usuario No.10

Número: 11	**Nombre del requisito:** Detener movimiento vertical
Programador: David Rodríguez Feitó	**Iteración Asignada:** 2

Prioridad: Alta	**Tiempo estimado:** 20 horas
Nivel de complejidad: Media	**Tiempo real:** 20 horas

Descripción:

1. **Objetivo:**
   Detener el movimiento vertical de la antena.
2. **Acciones para lograr el objetivo (precondiciones y datos):**
   - El usuario debe conectarse al sistema mediante Wifi o Ethernet.
   - El usuario debe enviar el comando “Detener elevación”.

Observaciones:

- La Raspberry Pi debe tener comunicación mediante Wifi con el dispositivo Arduino que controla los motores.

Prototipo de interfaz:

Tabla 14: Historia de usuario No.11

Número: 12	**Nombre del requisito:** Recibir azimut y elevación

Programador: David Rodríguez Feitó	**Iteración Asignada:** 3
Prioridad: Media	**Tiempo estimado:** 20 horas
Nivel de complejidad: Alta	**Tiempo real:** 20 horas
Descripción: 1. **Objetivo:** Establecer el azimut y elevación objetivos. 2. **Acciones para lograr el objetivo (precondiciones y datos):** - El usuario debe conectarse al sistema mediante Wifi o Ethernet. - El usuario debe enviar el comando “Establecer azimut y elevación”.	
Observaciones: No aplica	
Prototipo de interfaz: No aplica	

Tabla 15: Historia de usuario No.12

Número: 13	**Nombre del requisito:** Calcular azimut y elevación dado ubicación.
Programador: David Rodríguez Feitó	**Iteración Asignada:** 3
Prioridad: Media	**Tiempo estimado:** 20 horas
Nivel de complejidad: Alta	**Tiempo real:** 20 horas
Descripción: 1. **Objetivo:** Calcular el azimut y elevación del objetivo dadas sus coordenadas. 2. **Acciones para lograr el objetivo (precondiciones y datos):** - El usuario debe conectarse al sistema mediante Wifi o Ethernet. - El usuario debe enviar el comando “Establecer coordenadas”.	
Observaciones: No aplica	
Prototipo de interfaz: No aplica	

Tabla 16: Historia de usuario No.13

Número: 14	**Nombre del requisito:** Orientar antena automáticamente

Programador: David Rodríguez Feitó	**Iteración Asignada:** 3
Prioridad: Media	**Tiempo estimado:** 20 horas
Nivel de complejidad: Alta	**Tiempo real:** 20 horas

Descripción:

1. **Objetivo:**
   Orientar la antena automáticamente hasta alcanzar el objetivo.
2. **Acciones para lograr el objetivo (precondiciones y datos):**
   - El usuario debe conectarse al sistema mediante Wifi o Ethernet.
   - El usuario debe haber establecido el objetivo anteriormente.
   - El usuario debe enviar el comando “Orientación automática”.

Observaciones:

- La Raspberry Pi debe tener comunicación mediante Wifi con el dispositivo Arduino que controla los motores.

Prototipo de interfaz:

Tabla 17: Historia de usuario No.14

2.4 Análisis y diseño

Análisis y diseño es una disciplina en la metodología de desarrollo AUP-UCI en la que se refinan y estructuran los requisitos con el objetivo de lograr una mejor comprensión de estos. Además, en esta disciplina se modela el sistema, incluida su arquitectura, para que soporte todos los requisitos, incluidos los requisitos no funcionales.

2.4.1 Arquitectura del sistema

La arquitectura del software determina la estructura general del sistema. En su forma simple, la arquitectura define la organización de los componentes del programa, la forma en que estos interactúan y la estructura de los datos que utilizan (39).

Para el desarrollo del sistema de geoposicionamiento integral para el control del proceso de orientación de antenas parabólicas de grandes dimensiones se propone el uso del patrón de arquitectura de software Modelo-Vista-Controlador (MVC).

El patrón MVC es un patrón arquitectónico para el desarrollo de software que separa la lógica de negocio de la interfaz de usuario, lo que facilita la evolución de ambos aspectos e incrementa la reutilización y flexibilidad del sistema (40).

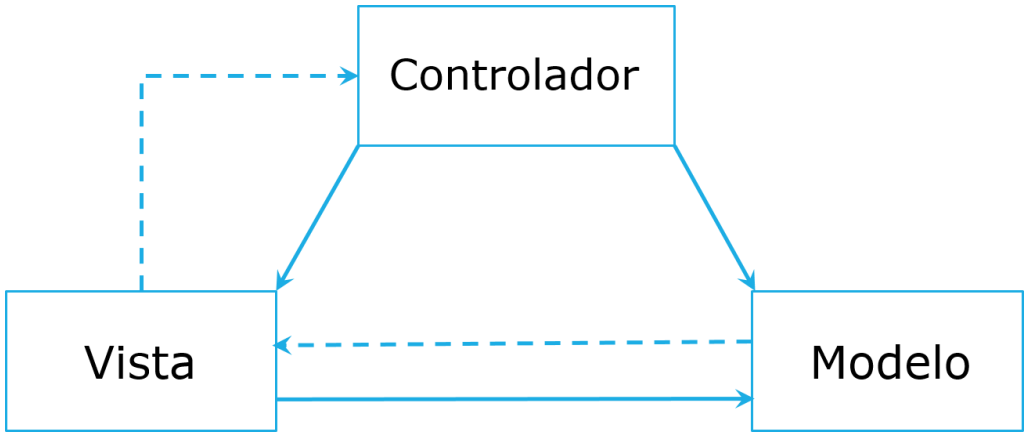


Ilustración 3: El patrón Modelo-Vista-Controlador [Elaboración propia]

El modelo maneja los datos del sistema y proporciona una interfaz para los otros componentes en la arquitectura. La comunicación depende del tipo de fuente de datos y de la forma en que se implementa el modelo.

Las vistas se encargan de manejar todos los aspectos gráficos de la aplicación; estas toman los datos del modelo y los muestran al usuario.

Los controladores contienen la interfaz entre las vistas, el modelo y los dispositivos de entrada (teclado, mouse, tiempo). Se encargan de procesar las entradas y procesarlas para realizar cambios en el modelo y actualizar las vistas.

Las clases del modelo presentes en la solución son DeviceClass y XsensReader, en las vistas se encuentra la clase MainWindow y en los controladores las clases XsensController y NetworkListener.

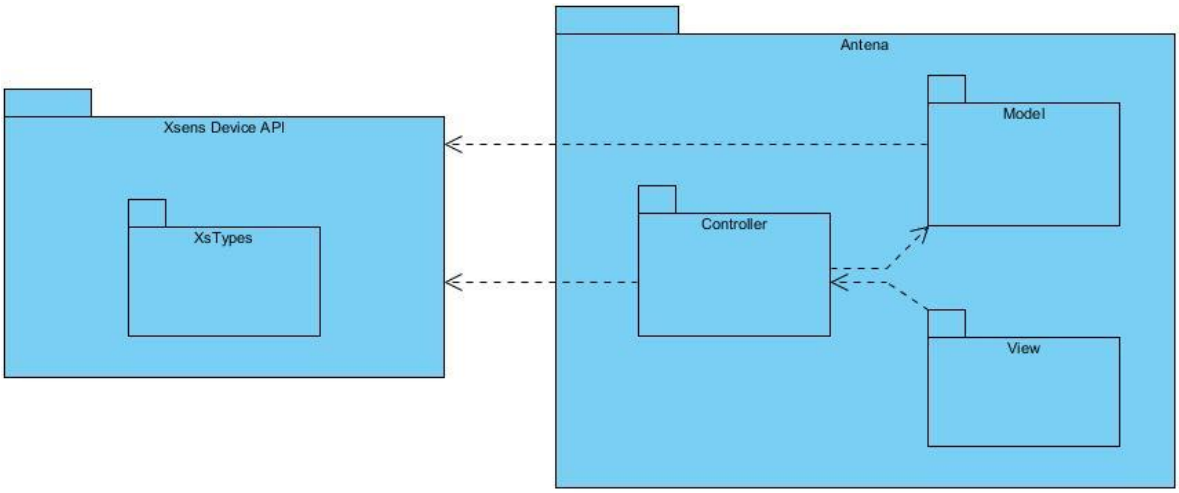


Ilustración 4: Diagrama de paquetes

La solución se estructura en dos partes, Xsens Device API y Antena. La primera parte contiene los manejadores para el dispositivo MTi Xsens, y a su vez contiene la subcarpeta XsTypes que contiene los tipos de datos manejados por el dispositivo y su Interfaz de Programación de Aplicaciones (API, Application Programming Interface por sus siglas en inglés). La segunda parte contiene las subcarpetas Controller, Model y View donde se ubican las entidades que representan las diferentes partes de la arquitectura de la aplicación.

2.4.2 Diagrama de clases del diseño

Un diagrama de clases muestra las clases de objetos en un sistema y sus relaciones. Incluye la siguiente información (38):

- Clases y sus asociaciones.
- Nombre de las clases de objeto.
- Atributos de las clases, y opcionalmente su tipo.
- Operaciones de las clases (también llamadas métodos).

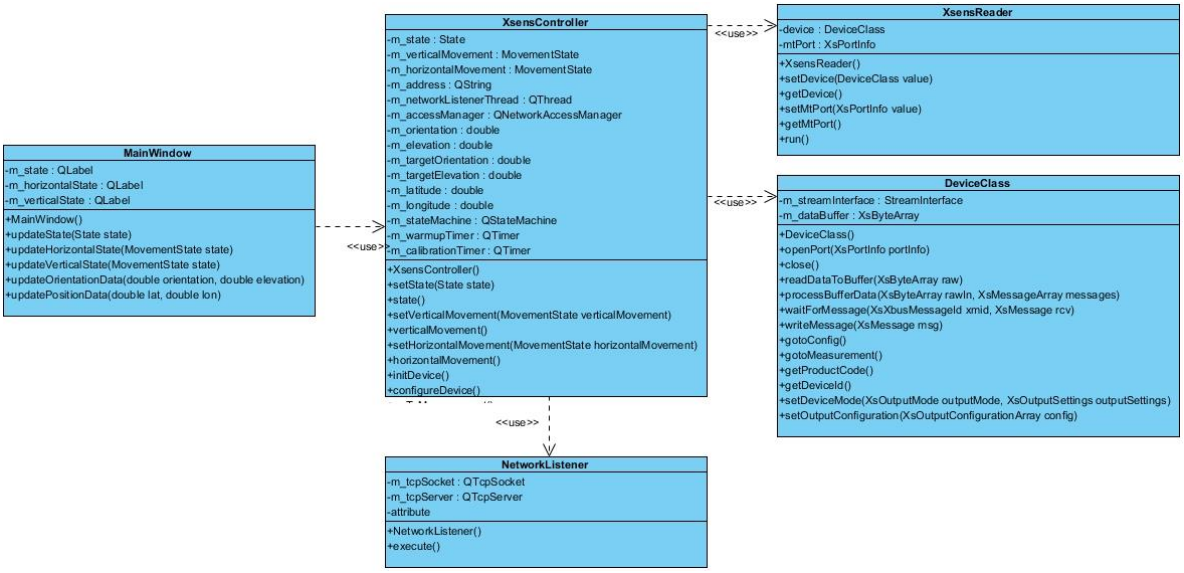


Ilustración 5: Diagrama de clases [Elaboración propia]

2.5 Patrones de diseño

Los patrones de diseño son soluciones a problemas comunes en el desarrollo de software. Estos brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Un patrón de diseño proporciona un esquema para refinar sus subsistemas o componentes, o las relaciones entre ellos. Describe la estructura de la solución de un problema que aparece repetidamente en componentes que se comunican entre ellos (41).

2.5.1 Patrones GRASP

Los patrones GRASP (General Responsability Assignment Software Patterns, por sus siglas en inglés) o Patrones Generales de Software para la Asignación de Responsabilidades indican cual es la manera de asignar responsabilidades a

objetos de software. Estos describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

Los patrones GRASP utilizados en la definición del sistema son:

- **Experto:** Asigna una responsabilidad a la clase que cuenta con la información necesaria para cumplirla (42). La aplicación de este patrón se manifiesta en todas las clases implementadas, porque tienen las responsabilidades que les corresponde en función de la información que manejan.
- **Creador:** Guía la asignación de responsabilidades relacionadas con la creación de objetos. Se aplica cuando una clase tiene la responsabilidad de crear una nueva instancia de otra (42). Su uso se manifiesta en la clase MainWindow encargada de crear una instancia de la clase XsensController.
- **Controlador:** Plantea asignar la responsabilidad del manejo de los eventos del sistema a una clase, donde un evento del sistema es un evento de alto nivel generado por un actor externo (42). Este patrón se evidencia en la clase NetworkListener.
- **Alta cohesión:** La cohesión es una medida de cuan relacionadas y enfocadas están las responsabilidades de una clase. Cada elemento debe realizar una labor única dentro del sistema que no sea desempeñada por el resto de los elementos (42). Este patrón se evidencia en todas las clases ya que contienen las mínimas responsabilidades necesarias y colaboran entre ellas para llevar a cabo una tarea.
- **Bajo acoplamiento:** El uso de los patrones experto y creador contribuye al bajo acoplamiento entre las clases del sistema. Se debe lograr que las clases estén lo menos ligadas entre sí que se pueda. De esta forma si se modifica alguna de estas, se tiene la menor repercusión posible en el resto de las clases (42). Este patrón se evidencia en todas las clases del sistema porque existe la mínima dependencia entre estas para permitir realizar modificaciones sin alterar su comportamiento.

2.5.2 Patrones GOF

Los patrones *Gang Of Four* (Pandilla de los Cuatro, por sus cuatro autores), o como se conocen comúnmente patrones GOF, descritos en el libro Design Patterns (43) definen un catálogo con 23 patrones básicos. Se clasifican según su propósito en:

- Creacionales: Abstracción del proceso de creación de instancias
- Estructurales: Muestran cómo las clases y los objetos son utilizados para componer estructuras de mayor tamaño.
- Comportamiento: Corresponde a los algoritmos y a la asignación de responsabilidades entre objetos.

En el desarrollo de la solución, con el objetivo de garantizar las buenas prácticas de programación, se emplearon los patrones GOF de comportamiento observador y estado. A continuación, se define como se describen estos patrones y su comportamiento:

- **Observador:** es un patrón de comportamiento que define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos (44). El uso de este patrón se ve reflejado en las funcionalidades *MainWindow::updateOrientationData* y *MainWindow::updatePositionData* las cuales reciben una notificación cuando los datos de la brújula y GPS sufren cambios.
- **Estado:** es un patrón de diseño que permite a un objeto cambiar su comportamiento en dependencia de su estado actual. Esto permite cambiar la funcionalidad de una clase u objeto sin alterar la interfaz que este expone (44). El uso de este patrón se manifiesta en la clase XsensController la cual cuenta con una máquina de estados interna y reacciona a las llamadas de sus métodos dependiendo del estado actual del dispositivo MTi Xsens y los motores.

2.6 Conclusiones parciales

Como resultado del desarrollo de este capítulo se obtuvo el diseño de la aplicación a implementar. Se definió un protocolo que permite el control remoto, ya sea manual o automático, del proceso de orientación de antenas parabólicas. Se logró identificar los requerimientos funcionales a implementar los cuales, según la metodología seleccionada, se describen como historias de usuario. De igual forma se identificaron los requisitos no funcionales que definen los elementos necesarios para que el sistema funcione correctamente. Se definió como modelo arquitectónico el patrón Modelo-Vista-Controlador lo que permitió estructurar y organizar los componentes del programa y se diseñó la solución mediante el uso de los patrones de diseño Observador y Estado. Todo esto permite a partir de este momento comenzar la construcción del sistema.

Capítulo 3: Implementación y pruebas

Este capítulo se centra en la implementación de la solución y la realización de pruebas a la misma, lo que permitirá validar de manera eficiente que la aplicación desarrollada está lista para su despliegue en un ambiente de producción.

3.1 Diagrama de componentes

Un diagrama de componentes proporciona una visión física de la construcción de un sistema. Muestra la organización de los componentes software, sus interfaces y las dependencias entre ellos. Puede incluir paquetes que permiten organizar la construcción del sistema en subsistemas y que recogen aspectos prácticos relacionados con la secuencia de compilación entre componentes y la agrupación de elementos en librerías (45).

Un componente es un módulo de software que puede ser código fuente, código binario, un ejecutable, o una librería con una interfaz definida. Una interfaz define las operaciones externas de un componente, las cuales determinan una parte del comportamiento del mismo (45).

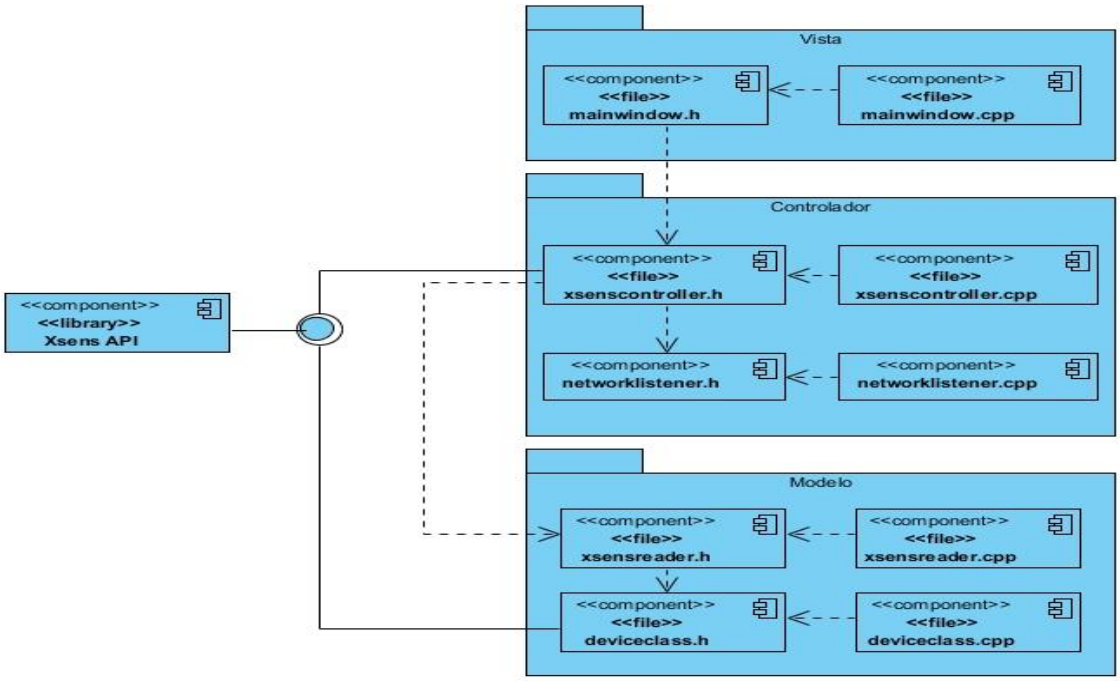


Ilustración 6: Diagrama de componentes [Elaboración propia]

Los componentes representados se describen a continuación:

- mainwindow.h/mainwindow.cpp: Componentes encargados de mostrar la interfaz gráfica de la aplicación y de transmitir las interacciones del usuario con la interfaz al controlador.
- xsenscontroller.h/xsenscontroller.cpp: Se encargan de recibir todas las entradas, tanto desde la interfaz gráfica como de red, y procesarlas para comunicarse con el modelo y generar una respuesta, ej. La actualización de un valor en la interfaz o la ejecución de un comando.
- networklistener.h/networklistener.cpp: Componentes cuyo objetivo es atender la interfaz de red y comunicar al controlador cuando se reciben comandos o información.
- xsensreader.h/xsensreader.cpp: Componentes encargados de comunicarse con el dispositivo MTi Xsens para la configuración y lectura de los datos.
- deviceclass.h/deviceclass.cpp: Actúan como interfaz ante los dispositivos de entrada/salida del sistema para detectar los periféricos conectados y por qué puertos se puede establecer la comunicación con estos.
- Xsens API: Interfaz para la comunicación con los dispositivos MTi Xsens.

3.2 Diagrama de despliegue

El diagrama de despliegue muestra las relaciones físicas entre los distintos nodos que componen un sistema. La vista de despliegue representa la disposición de las instancias de los componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución tal como un computador, teléfono inteligente u otro dispositivo (46).

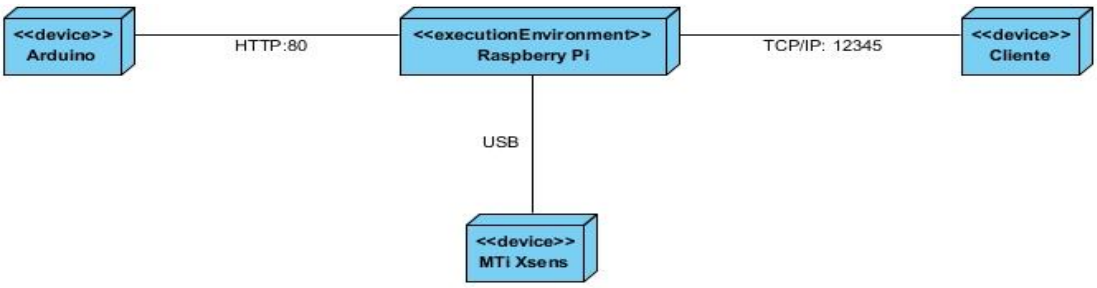


Ilustración 7 Diagrama de despliegue [Elaboración propia]

El cliente representa un dispositivo, ya sea una computadora o teléfono móvil, desde donde el usuario se conecta de forma remota a la aplicación usando el protocolo TCP/IP en el puerto 12345 para el envío de comandos e información. La aplicación se comunica mediante USB con el dispositivo MTi Xsens para leer los datos de la ubicación y orientación, haciendo uso de la biblioteca Xsens API. Además, se conecta con un dispositivo Arduino usando el protocolo HTTP en el puerto 80 para enviar las ordenes de movimiento a los motores de la plataforma.

3.3 Pruebas de software

Las pruebas de software son un elemento importante para garantizar la calidad del software. La meta de las pruebas es encontrar errores con la menor cantidad de tiempo y esfuerzo posible. Se considera una buena prueba aquella que tiene una alta probabilidad de encontrar un error, y aunque estas no pueden asegurar la ausencia de errores, si pueden demostrar que existen defectos, para que estos puedan ser corregidos a tiempo (39).

Estas actividades se planean con anticipación y se realizan de manera sistemática. Al aplicar pruebas a un software es necesario tener en cuenta el objetivo que se persigue. Debido a esto las pruebas son agrupadas por niveles que se encuentran en distintas etapas del proceso de desarrollo (39).

3.3.1 Niveles de prueba de software

De los niveles de pruebas que se mencionan a continuación, se exponen sus objetivos y principales características (39):

- **Pruebas unitarias:** se enfocan en la verificación de la unidad más pequeña del diseño del software: el módulo o componente. Están centradas en la lógica del procesamiento interno y las estructuras dentro de las fronteras de un componente.
- **Pruebas de integración:** su objetivo es identificar errores introducidos por la combinación de los componentes probados unitariamente. Estas describen como verificar que las interfaces entre componentes funcionan correctamente.

- **Pruebas de regresión:** tienen como objetivo determinar si los cambios recientes en una parte de la aplicación tienen un efecto adverso en otras partes de la misma. Consisten en volver a realizar las pruebas que se efectuaron anteriormente para comprobar que los cambios realizados no afecten el funcionamiento de la aplicación.
- **Pruebas de aceptación:** las pruebas de aceptación están diseñadas para usuarios no técnicos como probadores, clientes y usuarios del sistema. Estas permiten al cliente validar el cumplimiento de todos los requerimientos. Su uso es recomendado cuando se realiza software a la medida para un cliente.

De los niveles de prueba mencionados anteriormente se seleccionan para comprobar la calidad de la solución desarrollada las pruebas de integración y de aceptación, teniendo en cuenta que estas son las pruebas recomendadas por la metodología de desarrollo seleccionada.

3.3.2 Pruebas de integración

El objetivo de las pruebas de integración es tomar los componentes probados de forma individual y verificar que se integren de manera adecuada y realicen las funciones asignadas correctamente (39).

En esta fase de integración también aparecen nuevos errores y se debe proceder siguiendo estrategias para facilitar la solución de los mismos. Las estrategias básicas de integración son las siguientes (39):

- **Integración Big Bang:** Todos los componentes se combinan por adelantado y se prueba el sistema como un todo. Tiene como objetivo reducir la cantidad de pruebas.
- **Integración descendente:** Consiste en comenzar la integración por los módulos superiores de la jerarquía de control (ej. programa principal) e ir incorporando los módulos subordinados.
- **Integración ascendente:** Comienza la construcción y prueba con módulos atómicos (es decir, componentes en los niveles inferiores de la estructura del

programa). Esto permite que las funcionalidades de un determinado nivel siempre estén disponibles y se elimina la necesidad de representantes.

Con el objetivo de realizar las pruebas de integración al sistema se selecciona la estrategia Big Bang debido a que es adecuada en un contexto donde el sistema es pequeño, estable y existen pocos componentes a integrar.

Para lograr la correcta integración del sistema fue necesario la realización de dos iteraciones. En la primera iteración se detectaron las siguientes no conformidades:

- Los comandos enviados al controlador de los motores no coincidían con los definidos en este último.
- El controlador detenía el movimiento horizontal y vertical simultáneamente con un solo comando.
- La dirección de giro de los motores era contraria a la solicitada en los comandos.

Estas no conformidades fueron resueltas y en una segunda iteración no se detectaron no conformidades.

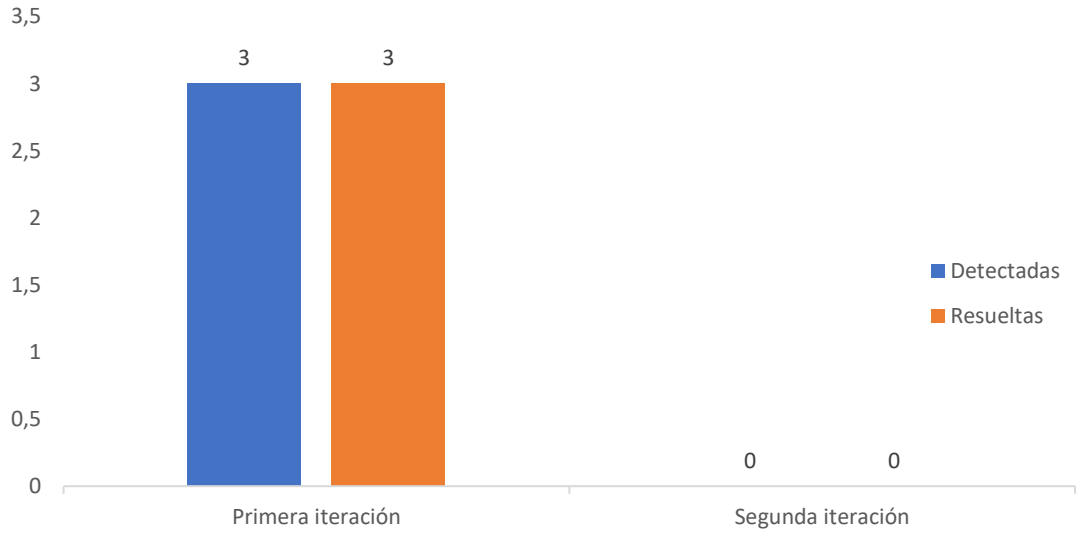


Ilustración 8: Resultados de las pruebas de integración

3.3.3 Pruebas de aceptación

El objetivo de las pruebas de aceptación es validar el comportamiento lógico de un escenario o caso de uso. Su característica principal es que se realiza con el cliente para validar que la aplicación permite completar las tareas para las que fue diseñada. Estas pruebas las realiza el propio cliente acompañado por el equipo de desarrollo y están orientadas a las funcionalidades del sistema. Las pruebas de aceptación presentan las siguientes características:

- Participación del usuario.
- Enfocada hacia la prueba de los requisitos de usuarios especificados.
- Está considerada como la fase final del proceso para crear una confianza en que el producto es apropiado para su uso en producción.

Para representar las pruebas de aceptación se definen los siguientes elementos:

- **Código:** Representa al caso de prueba. Incluye en número de la HU, de la prueba y si posee diferentes escenarios.
- **Historia de usuario:** Numero de la(s) historia(s) de usuario a la cual pertenece.
- **Nombre:** Junto al código conforma el identificador del caso de prueba.
- **Descripción:** Acción que debe realizar el sistema.
- **Condiciones de ejecución:** Describe las características y elementos que debe contener el sistema para realizar el caso de prueba.
- **Entrada/Pasos de ejecución:** Incluye las entradas necesarias, además de los pasos para realizar el caso de prueba.
- **Resultados esperados:** Descripción de la respuesta del sistema ante el caso de prueba.
- **Evaluación de la prueba:** Clasificación de la prueba en satisfactoria o insatisfactoria.

Las pruebas de aceptación generan como artefacto los Casos de Prueba (CP). Un caso de prueba cubre el software más a fondo y con más detalle que un caso de uso. Estos incluyen todas las funciones que el programa es capaz de realizar y

deben tener en cuenta el uso de todo tipo de datos de entrada/salida., cada comportamiento esperado y cada clase de defecto. Los casos de prueba deben cubrir todos los requisitos definidos en el sistema.

Caso de Prueba de Aceptación	
Código: 1	**Historias de usuario:** 1,2,3
Nombre: Inicio de la aplicación	
Descripción: Inicia la aplicación, lo que incluye el arranque del dispositivo, la configuración y el comienzo de la lectura de los datos.	
Condiciones de Ejecución: El dispositivo MTi Xsens debe estar conectado.	
Entradas/ Pasos de Ejecución: - Iniciar la aplicación.	
Resultado esperado: La aplicación a iniciado y configurado el dispositivo correctamente, y comienza a leer los datos de orientación y ubicación.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 18: Caso de prueba #1

Caso de Prueba de Aceptación	
Código: 2	**Historia de usuario:** 4
Nombre: Leer datos de orientación.	
Descripción: Muestra en tiempo real los datos de orientación leídos desde el dispositivo MTi Xsens.	
Condiciones de Ejecución: El dispositivo MTi Xsens debe estar conectado.	
Entradas/ Pasos de Ejecución: - Iniciar la aplicación.	

Resultado esperado: Se muestra en la interfaz el azimut y la elevación.
Evaluación de la prueba: Realizada la primera iteración se detectó una no conformidad, a la cual se le da solución y se realiza otra iteración que arrojó una evaluación satisfactoria.

Tabla 19: Caso de prueba #2

Caso de Prueba de Aceptación	
Código: 3	**Historia de usuario:** 5
Nombre: Leer datos de ubicación.	
Descripción: Muestra en tiempo real los datos de ubicación leídos desde el dispositivo MTi Xsens.	
Condiciones de Ejecución: El dispositivo MTi Xsens debe estar conectado.	
Entradas/ Pasos de Ejecución: - Iniciar la aplicación.	
Resultado esperado: Se muestra en la interfaz las coordenadas en latitud y longitud.	
Evaluación de la prueba: Realizada la primera iteración se detectó una no conformidad, a la cual se le da solución y se realiza otra iteración que arrojó una evaluación satisfactoria.	

Tabla 20: Caso de prueba #3

Caso de Prueba de Aceptación	
Código: 4	**Historia de usuario:** 6, 7
Nombre: Rotar antena.	
Descripción: Hace girar la antena hacia la izquierda y derecha.	
Condiciones de Ejecución: El usuario debe estar conectado a la aplicación.	
Entradas/ Pasos de Ejecución: - Iniciar la aplicación.	

- Enviar los comandos de movimiento horizontal por Wifi o Ethernet usando el puerto 12345.
Resultado esperado: La antena gira hacia la izquierda o la derecha según los comandos enviados.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 21: Caso de prueba #4

Caso de Prueba de Aceptación	
Código: 5	**Historia de usuario:** 8
Nombre: Detener movimiento horizontal	
Descripción: Detiene el movimiento de la antena en el eje horizontal.	
Condiciones de Ejecución: El usuario debe estar conectado a la aplicación.	
Entradas/ Pasos de Ejecución: - Iniciar la aplicación. - Enviar los comandos de movimiento horizontal por Wifi o Ethernet usando el puerto 12345. - Enviar el comando de detener el movimiento horizontal	
Resultado esperado: Se detiene el movimiento horizontal	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 22: Caso de prueba #5

Caso de Prueba de Aceptación	
Código: 6	**Historia de usuario:** 9, 10
Nombre: Inclinar antena	
Descripción: Aumenta o disminuye la inclinación de la antena en el eje vertical (elevación)	

Condiciones de Ejecución: El usuario debe estar conectado a la aplicación.
Entradas/ Pasos de Ejecución: - Iniciar la aplicación. - Enviar los comandos de movimiento vertical por Wifi o Ethernet usando el puerto 12345.
Resultado esperado: La antena aumenta o disminuye la inclinación según los comandos enviados.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 23: Caso de prueba #6

Caso de Prueba de Aceptación	
Código: 7	**Historia de usuario:** 11
Nombre: Detener movimiento vertical.	
Descripción: Detiene el movimiento vertical de la antena.	
Condiciones de Ejecución: El usuario debe estar conectado a la aplicación.	
Entradas/ Pasos de Ejecución: - Iniciar la aplicación. - Enviar los comandos de movimiento vertical por Wifi o Ethernet usando el puerto 12345. - Enviar el comando de detener movimiento vertical	
Resultado esperado: La antena detiene el movimiento vertical.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 24: Caso de prueba #7

Caso de Prueba de Aceptación	
Código: 8	**Historia de usuario:** 12
Nombre: Recibir azimut y elevación.	
Descripción: Recibe el azimut y elevación para posteriormente orientar la antena automáticamente.	
Condiciones de Ejecución: El usuario debe estar conectado a la aplicación.	
Entradas/ Pasos de Ejecución: - Iniciar la aplicación. - Enviar azimut y elevación por Wifi o Ethernet usando el puerto 12345	
Resultado esperado: Se recibe y almacena correctamente los datos enviados.	
Evaluación de la prueba: Realizada la primera iteración se encuentran 2 no conformidades, a las cuales se les da solución y se realiza otra iteración que arrojó una evaluación satisfactoria.	

Tabla 25: Caso de prueba #8

Caso de Prueba de Aceptación	
Código: 9	**Historia de usuario:** 13
Nombre: Calcular azimut y elevación dado ubicación.	
Descripción: Recibe coordenadas en formato de latitud y longitud, y calcula el azimut y elevación necesarios para dirigir la antena hacia ese objetivo.	
Condiciones de Ejecución: El usuario debe estar conectado a la aplicación.	
Entradas/ Pasos de Ejecución: - Iniciar la aplicación. - Enviar coordenadas por Wifi o Ethernet usando el puerto 12345	

Resultado esperado: Se recibe las coordenadas y se calcula correctamente el azimut y elevación.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 26: Caso de prueba #9

Caso de Prueba de Aceptación	
Código: 10	**Historia de usuario:** 14
Nombre: Orientar antena automáticamente.	
Descripción: Mueve la antena tanto horizontal como verticalmente hasta llegar al azimut y elevación establecidos.	
Condiciones de Ejecución: El usuario debe estar conectado a la aplicación.	
Entradas/ Pasos de Ejecución: - Iniciar la aplicación. - Enviar coordenadas o azimut y elevación por Wifi o Ethernet usando el puerto 12345 - Enviar el comando de orientar antena automáticamente.	
Resultado esperado: Comienza el movimiento de la antena hasta detenerse orientada hacia el objetivo especificado.	
Evaluación de la prueba: Realizada la primera iteración se detectó una no conformidad, a la cual se le da solución y se realiza una segunda iteración que arrojó 2 nuevas no conformidades las cuales fueron solucionadas, y se realiza una tercera iteración que arrojó una evaluación satisfactoria.	

Tabla 27: Caso de prueba #10

Fue necesaria la realización de tres iteraciones para alcanzar un estado correcto del sistema implementado. En la primera iteración se detectaron las siguientes No Conformidades (NC):

- Los datos de orientación se mostraban con un formato incorrecto, con 6 decimales en lugar de dos.
- Los datos de ubicación no se mostraban correctamente, debido a que ocupaban más espacio en la pantalla que el disponible.
- El puerto para la comunicación por Wifi y Ethernet era incorrecto.
- Al recibir los datos estos no se guardaban en memoria para su posterior uso.
- Al recibir el comando “Orientar antena automáticamente” la rotación se realizaba siempre en la misma dirección, sin tener en cuenta en qué sentido sería más rápida la orientación.

Se solucionaron las 5 NC anteriores y en una segunda iteración se detectaron las siguientes:

- El movimiento vertical de la orientación automática estaba invertido, es decir, cuando debía orientarse hacia arriba lo hacía hacia abajo y viceversa.
- Debido a la variación en los decimales de la lectura la antena realizaba varios movimientos cortos al alcanzar el objetivo de la orientación automática.

Después de solucionadas las 2 NC anteriores se realiza una nueva iteración donde no se detectaron nuevas NC.

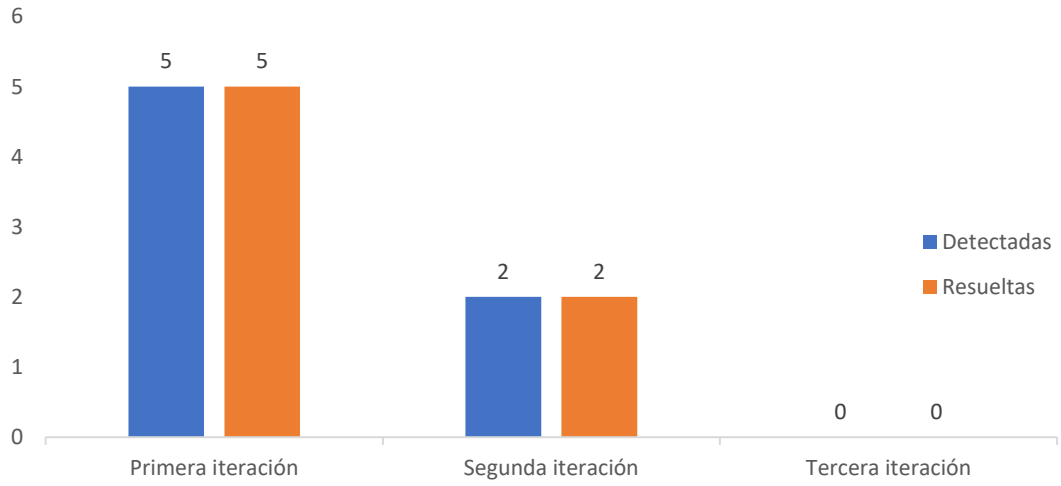


Ilustración 9: Resultado de las pruebas de aceptación

3.4 Conclusiones parciales

En este capítulo se presentó la distribución física del sistema y sus componentes mediante la realización de los diagramas de componentes y despliegue, lo que permitió un mayor entendimiento de la distribución física y lógica de la aplicación. Además, se ejecutó las pruebas de aceptación al sistema, lo que permitió validar el correcto cumplimiento de todos los requerimientos funcionales y no funcionales. A partir del resultado arrojado por las pruebas se determinó que la aplicación se encuentra lista para su puesta en funcionamiento en un ambiente de producción.

Concusiones generales

Con el desarrollo de la presente investigación se arriba a las siguientes conclusiones:

1. El empleo de la metodología, tecnologías y herramientas definidas, así como el estudio del estado del arte referente a la comunicación satelital, permitió obtener un producto acorde a las necesidades de la UBID.
2. Fue implementado un sistema de geoposicionamiento integral que permite el control remoto y automático del proceso de orientación de antenas parabólicas de grandes dimensiones.
3. La aplicación de pruebas a la solución permitió detectar y corregir a tiempo y con poco esfuerzo los errores existentes, obteniéndose un producto con la calidad requerida.

Recomendaciones

Se recomienda en la investigación:

- Ampliar el protocolo definido añadiendo la capacidad de enviar los datos de orientación a los clientes conectados a la antena.
- Incorporar control de acceso al sistema para garantizar la seguridad del mismo.
- Añadir soporte para otros sistemas de referencia de actitud y rumbo.

Referencias

1. LINARES COLUMBIÉ, Radame, PATTERSON HERNÁNDEZ, Mariela y VICIEDO TIJERA, Larisa. La información a través del tiempo. [en línea]. [Accedido 3 diciembre 2018]. Disponible en: http://bvs.sld.cu/revistas/aci/vol8_3_00/aci09300.htm

2. MARIANETTI, Ing. Osvaldo, GARCÍA GARINO, Dr. Carlos y TAGUA DE PEPA, Lic. Marcela. Las TIC, su importancia en la actualidad y el mercado laboral. [en línea]. [Accedido 5 diciembre 2018]. Disponible en: http://www.unidiversidad.com.ar/las-tic-su-importancia-en-la-actualidad-y-el-mercado-laboral

3. Satellite communication. [en línea]. [Accedido 21 noviembre 2018]. Disponible en: https://www.britannica.com/technology/satellite-communication

4. HUIDOBRO, José Manuel. Antenas de telecomunicaciones. *2013*. P. 18.

5. MOLERO SUÁREZ, Ing. Luis Guillermo. Fundamentos de la Comunicación Satelital. [en línea]. [Accedido 20 noviembre 2018]. Disponible en: https://www.urbe.edu/info-consultas/web-profesor/12697883/articulos/Comunicaciones%20Satelites%20y%20Celulares/Fundamentos%20de%20la%20Comunicacion%20Satelital.pdf

6. MARTÍNEZ FADRIQUE, Francisco. Tipos de órbitas. [en línea]. 7 octubre 2010. [Accedido 22 noviembre 2018]. Disponible en: https://www.gmv.com/blog_gmv/language/es/hay-distintos-tipos-de-orbitas/

7. PÉREZ VEGA, Constantino. Antenas con reflector parabólico. *2008* [en línea]. [Accedido 4 diciembre 2018]. Disponible en: https://www.scribd.com/document/374746716/Antenas-Con-Reflector-Parabolico-V4

8. SERRA-RICART, Dr. Miquel. Localización de astros mediante coordenadas (Altura y Acimut). [en línea]. [Accedido 22 noviembre 2018]. Disponible en: http://www.iac.es/adjuntos/www/actividad-altura-acimut.pdf

9. What is azimuth and elevation? *WhatIs.com* [en línea]. [Accedido 20 enero 2019]. Disponible en: https://whatis.techtarget.com/definition/azimuth-and-elevation

10. Xsens. AHRS Attitude Heading Reference System. *Xsens 3D motion tracking* [en línea]. [Accedido 23 noviembre 2018]. Disponible en: https://www.xsens.com/tags/ahrs/

11. European GNSS Agency. What is GNSS? [en línea]. 1 marzo 2016. [Accedido 23 noviembre 2018]. Disponible en: https://www.gsa.europa.eu/european-gnss/what-gnssGlobal Navigation Satellite

System (GNSS) refers to a constellation of satellites providing signals from space that transmit positioning and timing data to GNSS receivers. The receivers then use this data to determine location.

12. What is GPS? [en línea]. [Accedido 23 noviembre 2018]. Disponible en: https://www8.garmin.com/aboutGPS/

13. GPS.gov: Space Segment. [en línea]. [Accedido 4 diciembre 2018]. Disponible en: https://www.gps.gov/systems/gps/space/

14. GARCÍA ÁLVAREZ, José Antonio. Principio de funcionamiento del GPS. [en línea]. [Accedido 14 enero 2019]. Disponible en: http://www.asifunciona.com/electronica/af_gps/af_gps_10.htm

15. GARCÍA ÁLVAREZ, José Antonio. Tipos de receptores GPS. [en línea]. [Accedido 4 diciembre 2018]. Disponible en: http://www.asifunciona.com/electronica/af_gps/af_gps_9.htm

16. Xsens. MTi-G-710 - Xsens 3D motion tracking. [en línea]. [Accedido 25 noviembre 2018]. Disponible en: https://www.xsens.com/products/mti-g-710/

17. SatService. SatService: sat-nms Antenna Control and Tracking System [ACU] - en. [en línea]. [Accedido 24 mayo 2019]. Disponible en: https://satservicegmbh.de/en/products/antenna-control-and-tracking-system-acu.html

18. GDST. GD Satcom Technologies Model 7134 Satellite Antenna Control System. [en línea]. [Accedido 24 mayo 2019]. Disponible en: https://www.digisat.org/gd-satcom-technologies-model-7134-satellite-antenna-control-system

19. CCM. *CCM* [en línea]. [Accedido 25 noviembre 2018]. Disponible en: https://es.ccm.net/faq/1559-cual-es-la-diferencia-entre-los-protocolos-tcp-y-udp

20. TechTarget. What is TCP (Transmission Control Protocol)? [en línea]. [Accedido 25 noviembre 2018]. Disponible en: https://searchnetworking.techtarget.com/definition/TCP

21. HEATH, Steve. *Embedded Systems Design* [en línea]. [sin fecha]. [Accedido 27 noviembre 2018]. Disponible en: https://books.google.com/books/about/Embedded_Systems_Design.html?hl=es&id=BjNZXwH7HlkC

22. Xbot. ¿Qué es un microcontrolador? [en línea]. [Accedido 27 noviembre 2018]. Disponible en: http://sherlin.xbot.es/microcontroladores/introduccion-a-los-microcontroladores/que-es-un-microcontrolador

23. Arduino. Introduction to Arduino. [en línea]. [Accedido 27 noviembre 2018]. Disponible en: https://www.arduino.cc/en/Guide/Introduction#

24. GARCÍA COBO, Joaquin. ¿Qué es una placa SBC? *Hardware libre* [en línea]. 3 noviembre 2014. [Accedido 27 noviembre 2018]. Disponible en: https://www.hwlibre.com/que-es-una-placa-sbc/

25. Raspberry Pi. Raspberry Pi 3 Model B. [en línea]. [Accedido 20 mayo 2019]. Disponible en: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

26. ¿Qué es Ethernet? *Linksys* [en línea]. [Accedido 5 diciembre 2018]. Disponible en: http://www.linksys.com/es/r/resource-center/que-es-ethernet/¿En qué consiste exactamente el Ethernet? ¿Qué puedes hacer con el Ethernet? Wi-Fi vs. Ethernet. Descúbrelo todo aquí

27. ¿Qué es Wifi? [en línea]. [Accedido 27 noviembre 2018]. Disponible en: https://www.significados.com/wifi/

28. HERNÁNDEZ ORALLO, Enrique. El Lenguaje Unificado de Modelado. [en línea]. Disponible en: http://www.disca.upv.es/enheror/pdf/ActaUML.PDF

29. Visual Paradigm. [en línea]. [Accedido 26 noviembre 2018]. Disponible en: https://www.visual-paradigm.com/

30. Programming Language Definition. [en línea]. [Accedido 27 noviembre 2018]. Disponible en: https://techterms.com/definition/programming_language

31. ¿Qué es Framework? [en línea]. [Accedido 27 noviembre 2018]. Disponible en: https://searchdatacenter.techtarget.com/es/definicion/Framework

32. About Qt. [en línea]. [Accedido 28 noviembre 2018]. Disponible en: https://wiki.qt.io/About_Qt

33. Definición de sistema operativo. [en línea]. [Accedido 27 noviembre 2018]. Disponible en: https://definicion.de/sistema-operativo/

34. Raspberry Pi Foundation. Raspberry Pi - Software for the Raspberry Pi. [en línea]. [Accedido 28 mayo 2019]. Disponible en: https://www.raspberrypi.org/downloads/

35. FrontPage - Raspbian. [en línea]. [Accedido 5 diciembre 2018]. Disponible en: https://www.raspbian.org/FrontPage

36. RODRÍGUEZ SÁNCHEZ, Tamara. *Metodología de desarrollo para la Actividad productiva de la UCI*. 21 noviembre 2014.

37. CURBELO OLIVA, Lissa, ORTEGA RETURETA, Laura S., COLUMBIÉ CISNERO, Yoanna y GONZÁLEZ MILÁN, Yudelky. *Proceso de Desarrollo y Gestión de Proyectos de Software*. XETID, 2012.

38. SOMMERVILLE, Ian. *Ingeniería de software*. 9na edición. Pearson Educación, 2011. ISBN 978-607-32-0603-7.

39. PRESSMAN, Roger S. *Ingeniería de software: un enfoque práctico*. 7ma edición. 2010. ISBN 978-607-15-0314-5.

40. PAVÓN MESTRAS, Juan. *Estructura de las Aplicaciones Orientadas a Objetos: El patrón Modelo-Vista-Controlador (MVC)* [en línea]. [Accedido 16 abril 2019]. Disponible en: https://www.fdi.ucm.es/profesor/jpavon/poo/2.14.mvc.pdf

41. Patrones de diseño. [en línea]. [Accedido 1 mayo 2019]. Disponible en: https://es.ccm.net/contents/224-patrones-de-diseno

42. POLO USAOLA, Macario. *Patrones GRASP* [en línea]. [Accedido 1 mayo 2019]. Disponible en: www.inf-cr.uclm.es/www/mpolo/asig/0304/0102/patronesgrasp.pdf

43. GAMMA, E y OTROS. *Design Patterns. Elements of Reusable Object-Oriented Software*.

44. MATTSON, Timothy G., SANDERS, Beverly A. y MASSINGILL, Berna L. *Patterns for Parallel Programming*.

45. Diagrama de Componentes. *manuel.cillero.es* [en línea]. [Accedido 7 mayo 2019]. Disponible en: https://manuel.cillero.es/doc/metrica-3/tecnicas/diagrama-de-componentes/

46. Diagrama de Despliegue. *manuel.cillero.es* [en línea]. [Accedido 2 mayo 2019]. Disponible en: https://manuel.cillero.es/doc/metrica-3/tecnicas/diagrama-de-despliegue/

Anexos

Anexo 1: Estándar de codificación

Clases, enumeradores e interfaces: para estos identificadores se hará uso de la variante *UpperCamelCase*. Todas las palabras que componen dichos identificadores empezarán con mayúscula. Ejemplos:

- class XsensController { }
- enum MovementState { }
- enum State { }

Métodos: para los identificadores de métodos de una clase se empleará la variante *camelCase*. La primera palabra comenzara en minúscula, luego las demás palabras empezaran con mayúscula. Ejemplos:

- void initDevice()
- void configureDevice()
- void goToMeasurement()

Atributos de una clase: para los identificadores de atributos de una clase se usará la variante *camelCase* con el prefijo *m_*. Ejemplos:

- double m_targetOrientation
- double m_targetElevation
- XsensReader *m_reader
- QStateMachine m_stateMachine

Buenas Prácticas

- Usar 4 espacios de sangría en el código.
- Cada línea de código no debe sobrepasar los 80 caracteres.
- Declarar todos los atributos de la clase al principio de esta.
- Declarar todas las variables de los métodos al inicio de este, excepto las variables que definen las estructuras repetitivas.

Anexo 2: Prototipo del sistema

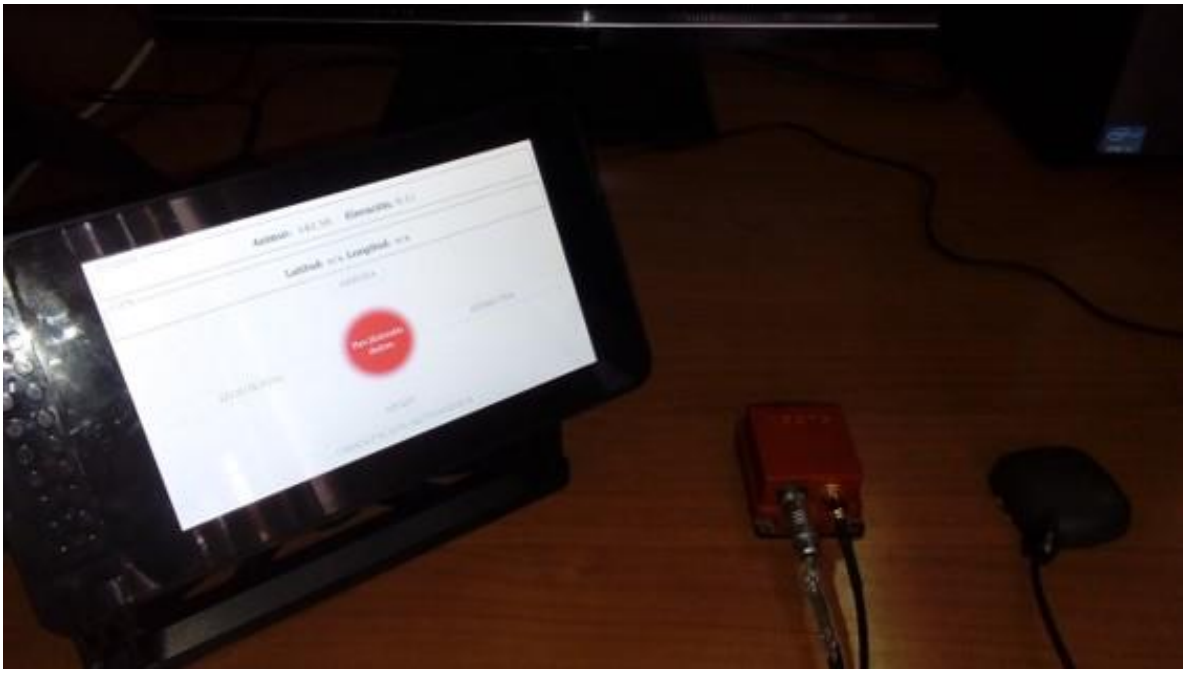


Ilustración 10: Prototipo del sistema