Sistema de adquisición de datos de una unidad de navegación inercial y ROS como herramienta de visualización.

Data acquisition system of and inertial navigation unit and ROS as a visualization tool

David Albeiro Taborda Alvarez, Andrés Guillermo Velásquez Gómez
Ingeniera de Sistemas y Computación, Universidad tecnológica de Pereira, Pereira, Colombia
Correo-e: david@sirius.utp.edu.co, andres@sirius.utp.edu.co

***Resumen*— El presente articulo muestra los resultados de las fases de investigación en el desarrollo del "proyecto Libélula" del laboratorio Sirius. El proyecto Libélula tiene como objetivo principal el diseño, implementación y desarrollo de un prototipo de un vehículo aéreo no manejado (UAV por sus siglas en ingles), basado en un helicóptero aeromodelo, el cual tendrá como función posicionarse en cualquier lugar que le sea indicado a partir de un punto inicial.**

Esta fase de investigación tiene como objetivo el desarrollo del controlador de comunicación de la unidad de navegación inercial xsens MTI-G, este sistema posee un sensor de movimiento con seis grados de libertad, un sistema de referencia de actitud y rumbo, un sensor GPS y un pequeño procesador para navegación. De igual forma en este trabajo se muestra el sistema utilizado como apoyo para el desarrollo del software de recolección de datos y el sistema de visualización.

Palabras clave*— *detección de movimiento, mti-g, orientación, robótica, ROS, sistema de adquisición de datos, unidad de navegación inercial.

***Abstract*— This article shows the results of the research stages in the development of "Project Dragonfly" Sirius laboratory. The Dragonfly Project's main objective is the design, implementation and development of a prototype of an unmanned aerial vehicle handled (UAV for short in English), based on a helicopter model aircraft, which function will be positioned at any place that is indicated from a starting point.**

This phase of research aims to develop communication controller inertial navigation unit Xsens MTI-G, this system has a motion sensor with six degrees of freedom, a system of attitude and heading reference, a GPS sensor and a small processor for navigation. Likewise, this paper shows the system used to support software development for data collection and visualization system.

***Key Word* —data acquisition system, inertial navigation system, *motion detection,* mti-g, orientation, robotics, ROS.**

I. INTRODUCCIÓN

Para un robot o vehículo autónomo es primordial el conocer el entorno en el cual se encuentra, su posición, orientación y la cantidad de objetos que lo rodean. La detección del movimiento y de la actitud son primordiales para el desenvolvimiento de aparatos robóticos y de su correcta interacción con el ambiente que los rodea. En el mercado se encuentran muchas herramientas con las cuales es posible simular, modelar entornos, robots y sensores, de las cuales su gran mayoría son de uso comercial, ofreciendo licencias costosas y no muy asequibles a universidades o grupos de investigación con pocos recursos financieros. ROS es una de las pocas herramientas GPL, que permiten a los desarrolladores de software no solo simular robots y entornos, sino también desarrollar software de manera ágil y de fácil implementación en los sistemas embebidos.

La detección de movimiento en entornos robóticos permite a los programadores tener el conocimiento necesario para tomar decisiones y aplicar algoritmos de control en los actuadores del robot necesarios para un correcto desenvolvimiento en el lugar donde se encuentra. El movimiento puede ser detectado por sonido, opacidad (sensores infrarrojos, ópticos y procesadores de imágenes), geomagnetismo, reflexión de energía, inducción electromagnética y vibración.

De esta forma, al usar la información de movimiento tomada por una unidad de medida inercial (IMU) y gracias a las herramientas de visualización de ROS y su facilidad para la intercomunicación entre procesos es posible capturar y visualizar en tiempo real los movimientos del sensor y mostrarlos directamente el la pantalla de un equipo de computo.

II. XSENS MTI-G

El MTI-G en una muy pequeña y liviana unidad de medida compuesta por un GPS, un sistema de medida inercial (IMU) con Sistema de Referencia de Actitud y Rumbo (AHRS) y un sensor de presión (barómetro) integrado. Este sensor provee información de la ubicación (latitud, longitud, altitud), estimación de orientación 3D, aceleración 3D calibrada, tasa de

Fecha de Recepción: 02 de Marzo de 2012
Fecha de Aceptación: 25 de Octubre de 2013

giro, campo magnético 3D de la tierra y presión atmosférica, además:

- Calculo en tiempo real de la posición, velocidad y aptitud, haciendo uso de un procesador digital de señales embebido.
- Receptor GPS de 16 canales.
- Sensibilidad de seguimiento de -158dBm.
- Soporte SBAS (WAAS, EGNOS, MSAS).
- Orientación 3D completa. Dentro de los 360°.
- Aceleración 3D, manejos de datos 3D dentro del campo magnético terrestre.
- Alta frecuencia de trabajo (100Hz para el DSP, 512Hz para los datos inerciales).
- Bajo consumo
- Sensores de estado solido.

El MTI-G usa un protocolo de comunicación binaria llamado "MTI comunication protocol" (Protocolo de comunicación MTI). Conocer este protocolo es necesario si se desea establecer comunicación de bajo nivel directa con el dispositivo a través de la interfaz RS-232. Los mensajes basados en el protocolo de comunicación MTI permiten al usuario cambiar la configuración de la unidad y recibir los datos de salida.
El MTI-G tiene dos estados, configuración (Config state) y medida (Measurement state). En el primer estado se realizar diferentes configuraciones de lectura y escritura. En el estado de medida el sensor entrega mensajes de datos que contienen la información dependiendo de la configuración.
Hay dos diferentes formas de entrar en los estados de medida y configuración. Al conectarse el dispositivo este inicia el procedimiento de "WakeUp" (despertar) y envía un mensaje de "wakeup". Si el mensaje no se responde con el mensaje "WakeUpAck" dentro de los siguientes 500 milisegundos, el sensor entra en el estado de "Measurement".
La comunicación con el MTI-G se realiza a través de mensajes de acuerdo con la estructura estándar, estos tienen dos estructuras; una con una longitud estándar y otra con una longitud extendida.
La longitud estándar del mensaje tiene un máximo de 254 bytes y es el más usado. Si la longitud del mensaje excede los 254 bytes necesita ser usada la longitud extendida [2].

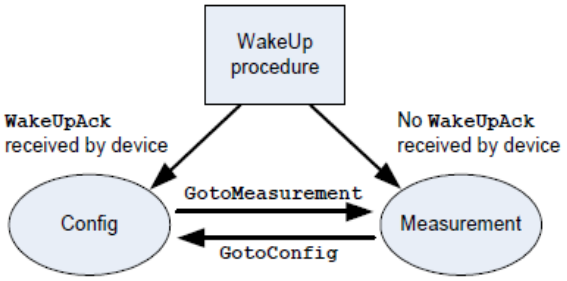

Figura 1: Estados del mti-g.

A. CONFIG STATE (ESTADO CONFIGURACIÓN)

El estado de configuración es usado para establecer y/o obtener varias opciones al MTI-G. La mayor parte de los ajustes cambian la configuración que define la funcionalidad en el estado de medida. Algunos de los ajustes que se pueden realizar son de velocidad de transferencia (baudrate), periodo de muestra y propiedades de sincronización entre otras. [3]

B. MEASUREMENT STATE (ESTADO MEDIDA)

En el estado medida, el sensor envía los datos al host en un solo sentido dependiendo da la configuración definida. Un solo mensaje llamado MTData es usado por todos los diferentes estados de salida, por eso es importante que el host conozca de antemano la configuración del sensor.
Si el host no responde el mensaje "despertar" al conectar el dispositivo, el MTI-G entra en el estado de medida. Justo antes de entrar en ese estado, este envía un mensaje de configuración el cual contiene los ajustes por defecto almacenados en la memoria interna [3].

III. MENSAJES

A. ESTRUCTURA DE LOS MENSAJES

La comunicación con el MTI-G se realiza a través de mensajes de acuerdo con la estructura estándar.

Un mensaje MTComm contiene los siguientes campos para longitud estándar:

Preamble	bid	mid	tamaño	datos	checksum

Tabla1: mensaje de longitud estándar [3].

Un mensaje MTComm extendido contiene los siguientes campos:

Preamble	bid	mid	tamaño	tamaño extendido	datos	checksum

Tabla 2: mensajes de longitud extendida [3].

IV. AMBIENTE DE PROGRAMACIÓN

Para el desarrollo del software necesario para establecer comunicación con el dispositivo se utilizó una distribución de Linux (Kubuntu) configurado con software necesario para la compilación del código en C y C++. Ambos lenguajes son completos y comunes, soportados por el fabricante y los seleccionados para el desarrollo del resto del proyecto.
El sensor se comunica con el sistema cliente por medio del protocolo serial, el cable de comunicación del dispositivo posee un conversos USB-serial, los sistemas Linux reconocen este tipo de dispositivos como un USB común, lo que facilita la comunicación con el mo.

El software de se configuro con los parámetros solicitados para una comunicación estándar entre los dispositivos[1] definidos en la Tabla 3.

Propiedad	Valor
Modo de salida (Output mode)	Orientación
Ajustes de salida (Output settings)	Orientación en modo cuaternión.
Frecuencia de muestreo	100 Hz
Velocidad de transferencia (Baudrate)	115000 bps
SyncIn	Deshabilitado
SyncOut	Deshablitado

Tabla 3: parámetros para comunicación serial.

V. ROS (*Robot Operating System*)

Resultado de la colaboración entre en proyecto "STAIR" de Standford University y el programa "Personal Robots Program" de Willow Garage nació el proyecto ROS (Robot Operating System). ROS es un sistema operativo para robots, pero no es un sistema operativo en el sentido tradicional de la gestión de procesos y la programación, una herramienta que proporciona una estructura de comunicaciones sobre el sistema operativo anfitrión, este posee propiedades como abstracción del hardware, control de dispositivos de bajo nivel y administración de tráfico de mensajes entre procesos y paquetes, todo bajo licencia GPL [4].
ROS tiene tres niveles de abstracción, el sistema de archivos, computación gráfica y el nivel común.

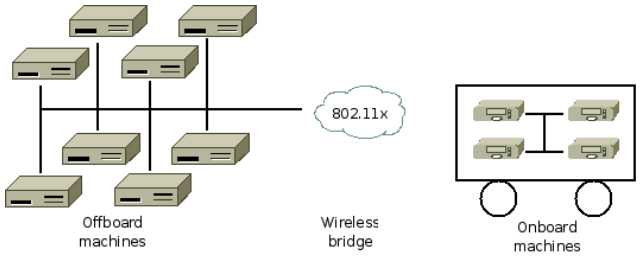

Figura 2: configuración de red típica de ROS

A. SISTEMA DE ARCHIVOS

Los conceptos de nivel de sistema de archivos, son recursos de ROS que se encuentran en el disco [5].

Nivel del sistema de Archivos:

- Paquetes (Packages): principal unidad de organización de software in ROS.
- Manifiesto (Manifests): el archivo (manifest.xml) contiene los metadatos del paquete, contiene la información de la licencia y dependencias, también contiene información específica del lenguaje de programación como banderas de compilación.
- Pilas (Stacks): colecciones de paquetes que proveen funcionalidades adicionales.
- Manifiesto de Pila(Stack Manifest): provee datos acerca de la pila.
- Tipos de mensajes (Mesage Types): descripción de mensajes, definidos como estructuras de mensajes enviadas a través de ROS.
- Tipos de Servicio (Service Types): descripción de servicios, definen las peticiones y servicios de las estructuras de datos de los servicios en ROS.

Nivel gráfico de procesos:

- Nodos (Node): los nodos son procesos, ROS esta diseñado para ser muy modular. El sistema de control de un robot comprende muchos nodos (ej: control de servomotores, control de posición, calculador de ruta).
- Maestro (Master): proceso principal de acción de ROS. Proporciona registro de nombres y de búsqueda para el resto del grafico de procesos.
- Parámetros del servidor: permite a los datos almacenarse mediante llaves en una ubicación central
- Mensajes (Messages): los nodos se comunican unos a otros enviando mensajes, un mensaje es una estructura de datos compuestos por tipos de archivos (enteros, punto flotante, caracteres, etc…). Utiliza estructuras como C y C++.
- Tópicos: los mensajes son ruteados y transmitidos a través de un sistema de transporte de modelo Publicación – Subscripción. Un mensaje se publica en un tópico. Un tópico es un nombre que se usa para identificar el contenido de los mensajes, de esta forma un nodo interesado en cierto tipo de mensaje puede subscribirse a un tópico y recibir de esta forma los datos del mensaje. Es posible publicar y suscribirse en diferentes tópicos.
- Servicios: El modelo publicar – Subscribir es un sistema muy flexible de comunicación, pero al ser diseñado como sistema "muchos a muchos" no es muy eficiente para interacciones solicitud – respuesta. Esas interacciones son realizadas por medio de servicios, definidos por dos estructuras de mensajes, solicitud y respuesta. Un nodo ofrece un servicio con un nombre y el cliente envía un mensaje de solicitud y espera la respuesta.

[1] parámetros definidos por el fabricante [3].

- Bolsas (bags): las bolsas son formatos de almacenamiento y reproducción de mensajes de datos de ROS. Son un importante mecanismo de almacenamiento de datos, como la información de captada por los sensores, para posterior mente reproducirla y analizarla.

B. HERRAMIENTAS Y EJECUCIÓN DE ROS

Para ejecutar cualquiera de los procesos de ROS es necesario previamente mantener en ejecución el núcleo del sistema ROS "roscore" encargado de administrar toda la estructura de comunicaciones del mismo, acto seguido es posible ejecutar los nodos necesarios:

$ roscore[2]
$ rosrun [nombre del paquete] [nombre del nodo] [2]

ROS provee comandos que permiten realizar seguimientos a los nodos, tópicos y tipos de mensajes que se están ejecutando, permitiendo así una administración mas flexible del entorno ROS [5].
Algunos comandos útiles son:

- rosnode: Ofrece información de nodos.
- rostopic: Ofrece información de tópicos
- rxgraph: Permite observar gráficamente que esta pasando en el sistema. Lista los nodos tópicos y mensajes, de manera gráfica.
- rxplot: Muestra gráficamente una trama de desplazamiento temporal de los datos publicados en los tópicos.
- rviz: nodo que funciona como entorno de visualización para robots en ROS.

C. MTI-G Y ROS

ROS ya tiene predefinidas estructuras de mensajes de los tipos de sensores más utilizados en robótica, para utilizar el visualizador era necesario dar el mismo formato a los datos recibidos del sensor de navegación.
El mti-g es una unidad de navegación inercial (INS), provee datos de actitud y de posición, tipos de mensajes que ROS trabaja de manera independiente. A continuación se especifican las estructuras de mensajes de estos dos tipos [5]:
sensor_msgs/Imu.msg

- Header header
  - uint32 seq, time stamp, string frame_id
- geometry_msgs/Quaternion orientation
  - float64 x, y,z,w.
- float64[9] orientation_covariance
- geometry_msgs/Vector3 angular_velocity
  - float64 x, y,z.
- float64[9] angular_velocity_covariance
- geometry_msgs/Vector3 linear_acceleration
  - float64 x, y,z.
- float64[9] linear_acceleration_covariance

sensor_msgs/NavSatFix.msg

- uint8 COVARIANCE_TYPE_UNKNOWN=0
- uint8 COVARIANCE_TYPE_APPROXIMATED=1
- uint8 COVARIANCE_TYPE_DIAGONAL_KNOWN=2
- uint8 COVARIANCE_TYPE_KNOWN=3
- Header header
  - uint32 seq, time stamp, string frame_id
- sensor_msgs/NavSatStatus status
  - int8 STATUS_NO_FIX=-1, int8 STATUS_FIX=0, int8 STATUS_SBAS_FIX=1, int8 STATUS_GBAS_FIX=2 uint16 SERVICE_GPS=1, uint16 SERVICE_GLONASS=2, uint16 SERVICE_COMPASS=4, uint16 SERVICE_GALILEO=8, int8 status, uint16 service
- float64 latitude
- float64 longitude
- float64 altitude
- float64[9] position_covariance
- uint8 position_covariance_type

Luego de diseñar el software de captura de datos del mti-g se procedió a formatear los datos en base a las estructuras anteriormente descritas y se inicio la publicación de los datos en dos tópicos diferentes, en *"pub_umi"* los datos de orientación del dispositivo y en *"pub_nav"* los datos de navegación (latitud, longitud, altitud). Usando utilitario *"rostipoc echo[nombre_topico]"* es posible observar los datos publicados en ambos tópicos como se muestra en las figura 3.

Figura 3: datos publicados de a) navegación y b) orientación.

Las librerías de ROS están estandarizadas para trabajar con datos de orientación en formato de cuaterniones, asi que se

[2] Nota: Cada línea debe ser ejecutada en diferentes consolas de comandos

configuro inicialmente el INS para que proporcionara las coordenadas de orientación en este formato.

D. RVIZ

Es un nodo de visualización en ROS, este permite visualizar sensores, construir entornos virtuales para simulación y construir modelos tridimensionales de robots a partir de figuras geométricas básicas. En él es posible mostrar transmisión de cámaras, dibujar ejes y cuadriculas, imágenes, rutas, de desplazamientos, polígonos, figuras geométricas y marcos de coordenadas. Las figuras geométricas se identifican como marcadores (Markers), estos marcadores poseen propiedades de color, posición, tipo, escala entre otras.

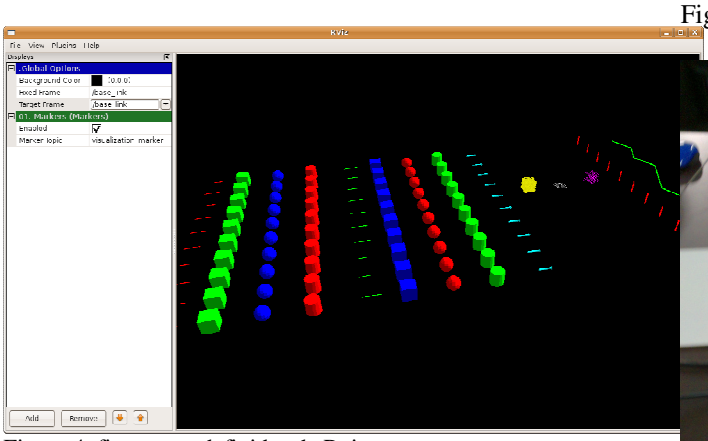

Figura 4: figuras predefinidas de Rviz.

Para observar la captura de los datos del INS se utilizaron tres cubos de tres tamaños y colores diferentes con el objetivo de visualizar de mejor forma los movimientos tomados del sensor, los cubos leen del tópico del INS actualizando su orientación. Luego cada cubo publica su estado como marcador para ser visualizado en Rviz y se agregan los marcadores al mismo.

VI. RESULTADOS

En las figuras 5, 6 y 7 se puede apreciar como los cubos se mueven en sincronía con el INS, girando con las mismas velocidades y sentidos.

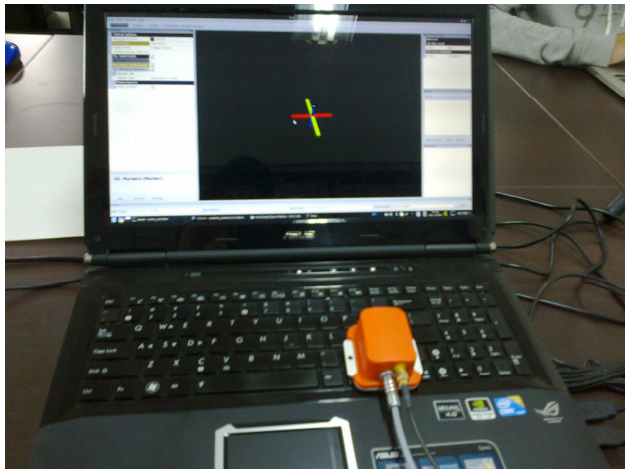

Figura 5: captura del sensor en posición estática.

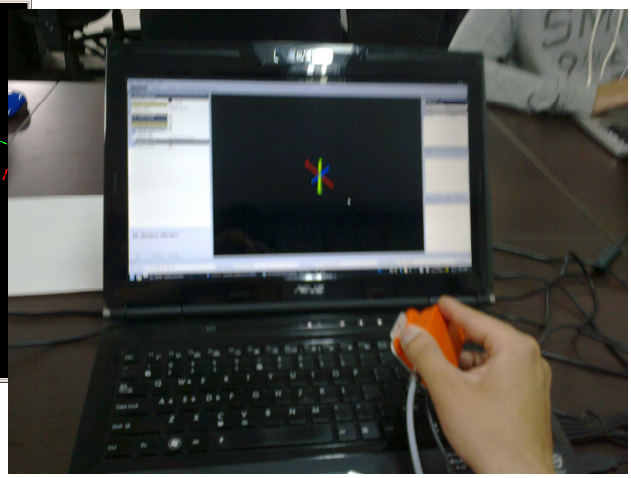

Figura 6: captura girando el sensor a la derecha.

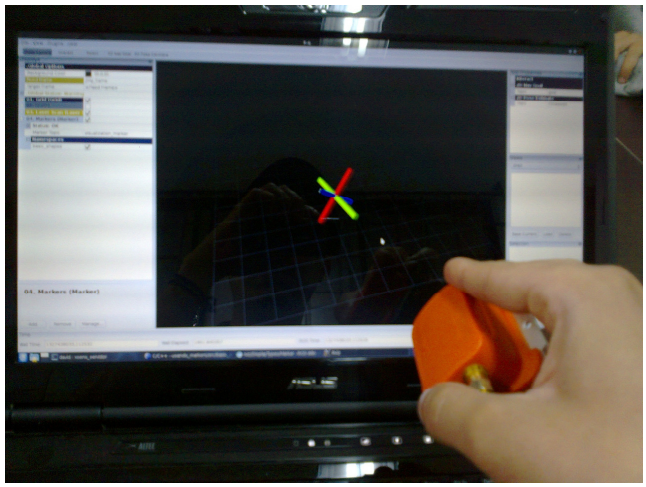

Figura 7: captura girando el sensor a la izquierda.

VII. CONCLUSIONES

La sensibilidad de este tipo de sensores es visible en las pruebas, sus componentes internos fabricados en estado solido permiten ubicarlos sobre una gran variedad de robots y en una gran variedad de escenarios, generando buenos resultados.

El filtro de Kamlan integrado en el dispositivo evita en cierta medida la implementación de filtros de corrección de posición adicionales en el software en aplicaciones de navegación.

ROS es un software libre de muy fácil acceso, que a pesar de llevar un poco más de 4 años de madurez es muy estable y esta bien soportado. Este software esta siendo cada vez más utilizado gracias a su fácil instalación y uso.

La facilidad que tiene ROS para intercomunicar sus procesos facilita en gran medida la integración e implementación de sensores en proyectos aplicados de robótica.

La visualización directa de los datos del sensor sobre la pantalla del computador permite a los desarrolladores realizar análisis y aplicar filtros o algoritmos para permitir a sus desarrollos robóticos interactuar de una mejor manera con el entorno en el cual se encuentren.

ROS tiene hasta el momento una numerosa lista de sensores y estructuras de datos definidas de manera estándar en sus librerías, lo que facilita a los desarrolladores el diseño de sus proyectos robóticos

La posibilidad de integrar los datos de este sensor con otro tipo de dispositivos como sensores de visión permite no solo georeferenciar los datos sino también construir directamente entornos virtuales y visualizarlos directamente sobre la aplicación para posteriores análisis.

REFERENCIAS

[1]. Xsens Technologies B.V.. MTI-G. [En línea] 2009. http://www.xsens.com/en/general/mti-g

[2]. Xsens Technologies B.V.. “MT Low-Level Communication Protocol Documentation”, Revisión J, Octubre 19, 2008.

[3] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, A Ng, ROS: an open-source Robot Operating System, Willow Garage, 2009.

[4] ROS:org. ROS, [En línea] 2009, http://www.ros.org/wiki/

[5]. J. Corbet, G. Kroah-Hartman, A. Rubini, *Linux Device Drivers,* 3rd Ed, O'Reilly, 2005.

[6]. N. Pons, *Linux principios básicos del uso del sistema,* Editions ENI, 2005.

[7]. H. M. Deitel, P. J. Deitel, Cómo programar en C/C++ y Java, 4ta Ed, Prentice Hall, 2004.

[8]. O. J. Woodma, “An introduction to inertial navigation”, University of Cambridge Computer Laboratory, Cambridge, MA, Tech. Rep. TR-696 (1476-2986), agosto 2007.