TRABAJO DE FIN DE GRADO

GRADO DE INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

Integración del dispositivo Stargazer en un robot social para la detección de marcas en el entorno

---

Autor: Juan Fernando del Arco Fernández de Heredia
Tutor: Álvaro Castro González

---

26 de septiembre de 2016

Agradecimientos

La primera persona a la que me gustaría dar las gracias en este documento es a mi abuelo por sus buenos consejos y su preocupación durante el desarrollo de este proyecto.

Me gustaría agradecer también al resto de mi familia y a los miembros del laboratorio de Social Robots, en especial a Álvaro (mi tutor) y a José Carlos, por su paciencia, ayuda y optimismo (sobre todo por su paciencia).

Gracias también a mi grupo de amigos, ya que sin sus... "ánimos"... en este último mes, probablemente estaría aún escribiendo la introducción.

Por último quisiera mencionar en estos agradecimientos a Michael Koval (aunque seguramente nunca vaya a ver esto) de la "Carnegie Mellon University", por su casi inmediato soporte con los *drivers* del Stargazer.

Resumen

Este trabajo pretende solucionar el problema de deslocalización que sufren algunos robots por la acumulación de pequeños errores en su sensorización, mediante la instalación de un dispositivo capaz de identificar marcadores pasivos situados en el entorno, el Stargazer de la marca coreana Hagisonic.

El objetivo final es conseguir integrar el dispositivo Stargazer en la navegación de un robot social, el Mbot, con el fin de desarrollar aplicaciones basadas en su uso que sirvan para mejorar el desplazamiento del robot.

El desarrollo del código necesario en la integracion del dispositivo se ha llevado a cabo sobre el framework ROS (Robot Operating System).

Para comprobar el correcto funcionamiento del dispositivo y sus aplicaciones se han llevado a cabo diversas pruebas tanto de precisión como de rendimiento.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Hace años el concepto de robot estaba ligado únicamente a la realización de trabajos concretos en la industria. La función de estos robots se limitaba a la realización de tareas repetitivas asociadas al montaje de productos tales como automóviles. A día de hoy, la robótica está cada vez más presente en la sociedad a todos los niveles. Esto implica que ya no solo hay robots en la industria, sino que se extienden por otros campos muy desligados de ésta. Existen robots de todo tipo: robots con aplicaciones en transporte militar terrestre [19], drones autónomos utilizados en meteorología [16], robots bailarines para actuaciones en eventos [21], exploradores espaciales [12] y un largo etcétera.

Hoy en día, se prevee que esta masificación de la robótica puede llevar a que los robots sean parte del día a día de muchas personas. La robótica de servicio es cada vez mas común, pudiendo encontrar un numero creciente de robots que desempeñan toda clase de tareas que implican una aproximación directa o indirecta hacia las personas. Según la IFR (Federación Internacional de Robótica), la definición de robot de servicio es la siguiente: "un robot de servicio es un robot que opera semi o totalmente autónomamente, para realizar servicios útiles para el bienestar de los seres humanos y equipos, con exclusión

de las operaciones de fabricación" [13]. Existen todo tipo de robots de servicio, desde robots domésticos encargados de la limpieza [5] hasta robots destinados a la recepción en locales. Dentro de este campo de la robótica se encuentran robots cuyo objetivo explicito es la interacción directa con personas, los robots sociales [4]. Un robot social es aquel que esta pensado para comunicarse con las personas e interaccionar con ellas utilizando sus reglas de comportamiento. Este tipo de robots se caracterizan por tener un aspecto amigable y por tener mecanismos que les dotan de las capacidades necesarias para comunicarse con las personas. En la actualidad, la comunicación hombre-robot se realiza mediante diálogos predefinidos por los desarrolladores de los robots. Utilizando este tipo de comunicación, estos robots se pueden emplear también para realizar ejercicios de memoria [9] o juegos simples.

Cuando estos robots son móviles, un factor a tener en cuenta en su desarrollo, es la manera en la que se desplazan[1], ya que lo hacen a través lugares por los que también transitan personas. La importancia del desplazamiento de un robot social, viene en parte ligada a la peligrosidad que éste puede suponer tanto para las personas de su entorno como para el propio robot. Esto hace necesario dotar a los robots de unos mecanismos de navegación fiables, especialmente en los entornos que comparten con las personas.

Por regla general, el desplazamiento de un robot autónomo se encuentra condicionado por la información que posee sobre su entorno, concretamente su posición y la de otras entidades. En mayor o menor medida, todos los robots son capaces de percibir el entorno que les rodea a fin de interaccionar con él. Cuanta más información sea capaz de captar un robot, mejor será su respuesta ante el mundo que le rodea. En el caso de los robots sociales, la percepción del medio cobra una importancia adicional ya que, en un caso ideal, estos robots

[1] Cómo calculan sus trayectorias, cómo saben a dónde deben dirigirse, etc.

deben ser capaces de interpretar distintas situaciones que se pueden presentar en torno a las personas con las que interaccionan y actuar en consecuencia.

En algunos casos, el medio por el que los robots perciben su entrono puede condicionar su funcionamiento. Un ejemplo de ésto se encuentra en la navegación de algunos robots. La navegación es el sistema mediante el cual el robot es capaz de guiarse y calcular sus trayectorias a la hora de desplazarse por el entorno. Para que este sistema funcione de forma correcta, es crucial que se tenga un conocimiento lo mas certero posible de la posición del robot, es decir, mantenerlo localizado [3]. Un pequeño error en la toma de datos, en principio no tiene por qué afectar al correcto comportamiento del robot, pero la acumulación de pequeños errores puede acabar derivando en problemas más graves como puede ser que el robot pierda su ubicación. Existen diversos métodos mediante los cuales se pueden mitigar estos efectos negativos. Uno de estos métodos es la utilización de sensores adicionales que contrasten la información obtenida por el robot.

El proyecto aquí desarrollado se centra en la integración de uno de estos sensores adicionales para permitir al robot percibir una serie de marcas externas que le van a ayudar durante la navegación.

1.1. Marco del trabajo

Este proyecto se ha llevado a cabo en el laboratorio "Social Robots" de la Universidad Carlos III de Madrid. Este laboratorio se encarga del desarrollo de robots sociales y de la realización de todo tipo de proyectos de investigación sobre ellos. Estos proyectos abarcan una gran cantidad de temas relacionados con la interacción Humano-Máquina y con la incorporación de nuevas tecnologías o de nuevas funcionalidades sobre los robots.

En este laboratorio se trabaja sobre tres robots principales: Mbot, Maggie y Mini. Mbot y Maggie son robots móviles capaces de desplazarse por el entorno de forma segura. Mini, en cambio es un robot de sobremesa capaz de mover sus extremidades para transmitir emociones.

El robot Mbot (figura 1.1), pertenece al proyecto europeo MOnarCH [6] [17], cuyos objetivos son los siguientes: (1) el desarrollo de un nuevo marco de trabajo para modelar sociedades mixtas humano-robot y (2) su demostración utilizando una red de robots heterogéneos y sensores en el área de pediatría de un hospital oncológico.

Figura 1.1: Mbot funcionando en el hospital. (https://goo.gl/f8ovqF)

Mbot es un robot social diseñado para interaccionar con niños, por ello se encuentra equipado con distintos dispositivos, tales como una pantalla y un proyector, que le permiten realizar tareas lúdicas y/o educativas con estos. Se trata de un robot de un metro de alto cuyo desplazamiento se ejerce mediante

cuatro ruedas omnidireccionales. Ésto dota al robot de una gran movilidad. A su vez cuenta con dos ordenadores a bordo, uno para gestionar la navegación del robot y otro para los sistemas de interacción Humano-Máquina.

1.2. Motivación

Un hecho presente durante el funcionamiento del robot, es que en ocasiones se producen ciertos problemas de deslocalización en el programa de navegación de los robots debidos a la acumulación de errores en la percepción. Estos problemas son causados mayormente porque los datos de la navegación difieren de los datos captados por la percepción del robot. A causa de esto, el robot puede perder la noción de cuál es su posición actual y esta situación puede originar problemas graves como puede ser que el robot se aproxime a áreas peligrosas, como por ejemplo unas escaleras.

El proyecto aquí descrito, pretende solucionar este problema de desorientación a fin de evitar resultados catastróficos debidos a posibles errores en la navegación de los robots. Para ello se ha procedido a la instalación del dispositivo Stargazer de la marca coreana Hagisonic en el robot Mbot y se ha desarrollado una aplicación capaz de reposicionar a este robot en el mundo. Este dispositivo es capaz de llevar ésto a cabo detectando marcadores colocados en el techo. Se pretende que, en un futuro, los resultados de este proyecto puedan ayudar a realizar nuevas aplicaciones con este dispositivo para los robots del laboratorio.

Toda la programación existente en el proyecto se ha llevado a cabo sobre el sistema ROS, ya que éste es el utilizado sobre todos los robots del laboratorio dada su versatilidad. A su vez, los lenguajes de programación utilizados en el desarrollo del código son C++ y Python.

1.3. Trabajos relacionados

El dispositivo que se va a integrar en el robot, el Stargazer, pertenece a la familia de dispositivos conocidos como IPS (Indoor Positioning Systems) [2]. Estos dispositivos son aquellos que utilizan algún sistema de balizas u otras referencias en interiores a fin de localizar de forma inalámbrica entidades en espacios cerrados. Existen diversas tecnologías utilizadas en la navegación de interiores además de los sistemas de balizas. En este capítulo se mostrarán distintas tecnologías de todo tipo utilizadas en escenarios similares al de este proyecto.

En el centro "Nara Institute of Science and Technology" de Ikoma (Japón) el equipo de Junichi Ido [8] utilizó un sistema basado en la captura de imágenes mediante una cámara montada en la cabeza de un robot humanoide con el fin de hacer que éste navegara en el interior de un complejo de oficinas. Para ello, este robot grababa una secuencia de imagenes de la ruta deseada que procesaba y almacenaba en su memoria. Una vez hecho esto, el robot era capaz de utilizar estas imágenes para guiarse y reproducir el recorrido almacenado utilizando correlaciones entre la imagen percibida y las almacenadas.

En el "Institute of Geodesy and Photogrammetry" en Zurich, Tilch y Mautz desarrollaron el sistema CLIPS [20] (Camera and Laser based Indoor Positioning System). Este sistema utiliza una cámara en conjunto con un emisor láser. El láser emite rayos de luz cuya proyección forma un mallado de puntos sobre la superficie deseada. En base a esto, la cámara detecta la distorsión que aparece en la malla y es capaz de determinar la posición y orientación de la superficie sobre la cual se proyecta el láser. Se llegó a la conclusión de que se podía obtener la posición relativa de la cámara con éxito utilizando el Algoritmo de Stewénius de cinco puntos [18].

En 2015, en la "University of Rostock" (Alemania), se utilizó el dispositivo Stargazer en el desarrollo de un sistema de navegación multi-planta [1]. Se utilizó la capacidad que tiene este dispositivo de crear un mapa basado en marcadores pasivos para que ejerciera de base sobre la que sustentar la localización del modelo del sistema multi-planta.

1.4. Organización del documento

El contenido de este documento se organiza en capítulos cuyo contenido es el siguiente:

- **Introducción.** En este capítulo se da una vista general sobre el campo de estudio en el que se desarrolla el proyecto así como del marco en el que se sitúa. A su vez se trata su planteamiento y se muestra la organización de este documento.

- **Herramientas de desarrollo.** En este capítulo se explica a nivel básico el funcionamiento de ROS (Robot Operating System), así como sus características principales y conceptos a tener en cuenta a la hora de trabajar con él.

- **Integración del Stragazer.** En este capítulo se muestran las especificaciones técnicas del dispositivo. A su vez, también se explica su funcionamiento tanto a nivel software como hardware.

- **Aplicaciones de relocalización.** En este capítulo se expone la funcionalidad que se ha desarrollado como aplicación directa del dispositivo en la navegación así como su funcionamiento.

- **Pruebas.** En este capítulo se describen detalladamente las distintas pruebas de precisión y funcionamiento realizadas con el dispositivo, y los resultados obtenidos de ellas.

- **Conclusiones.** En este capítulo se consideran los resultados y se proponen posibles mejoras para el futuro del trabajo realizado con el dispositivo así como posibles trabajos futuros.

- **Planificación del proyecto.** En este capítulo del apéndice se muestra un desglose de las distintas etapas de desarrollo del proyecto asi como su planificación temporal.

- **Presupuesto.** En este capítulo del apéndice se muestra los fondos que son necesarios para financiar este proyecto.

- **Ficha técnica del Stargazer.** Este anexo incluye la ficha técnica completa del fabricante.

Capítulo 2

Framework de desarrollo. Robot Operating System

Conocer el funcionamiento de Robot Operating System (o ROS) es un requisito crucial para poder desarrollar cualquier programa que se vaya a utilizar en el robot. Por ello, antes de entrar de lleno en la integración del dispositivo Stargazer, es necesario tomar cierto contacto con la forma de trabajar en ROS.

En este capítulo se muestran los conceptos básicos que se deben conocer para trabajar con ROS así como su funcionamiento más básico. Ademas de ésto, se mostrará el concepto de las transformadas para relacionar sistemas de coordenadas dada su importancia en el proceso de integración del dispositivo Stargazer.

2.1. ¿Qué es ROS?

ROS [14] es un meta-sistema operativo de código abierto que se utiliza como marco de trabajo (o *framework*) con el propósito de facilitar el desarrollo de software para robots a distintos niveles de modo que el mismo código pueda ser utilizado entre varios ordenadores. Para ello cuenta con una serie de

herramientas y librerías propias que ayudan a llevar a cabo dicho propósito. Por todo esto, el uso de ROS se encuentra cada vez mas extendido por todo el mundo, lo cual ha dado lugar a una gran comunidad en constante crecimiento. A su vez, existe un gran numero de paquetes de software disponibles y que cubren un amplio numero de aplicaciones. Todas estas características hacen que ROS esté dotado de una gran versatilidad que le permite funcionar en un gran numero de arquitecturas muy diferentes entre si.

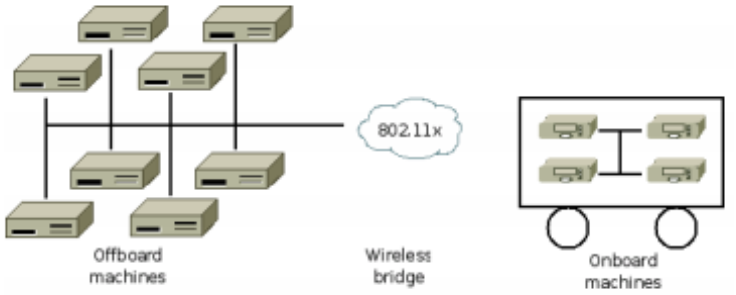


Figura 2.1: Ejemplo de red P2P.

A grandes rasgos, el funcionamiento de ROS se basa en la intercomunicación de distintos procesos que pueden estar localizados en distintos ordenadores conectados a una red con tipología *peer-to-peer*.

Un ejemplo donde se pueden observar claramente este tipo de comunicaciones son los robots que tienen que permanecer en servicio durante largos periodos de tiempo. Éste es el caso que se muestra en la figura 2.1 que consta de varios ordenadores de mayor potencia fuera del robot encargados de realizar operaciones complejas comunicados en red LAN inalámbrica vía ROS con los ordenadores del robot.

Un aspecto importante de ROS es la utilización de paquetes. Los **paquetes** sirven para organizar los distintos proyectos con los que se esté trabajando. Un

paquete contiene desde los procesos ejecutables (nodos) hasta las librerías y dependencias del programa.

Toda la información disponible sobre ROS se puede encontrar en el siguiente enlace:

http : //www.ros.org/

2.2. Nodos y mensajes

Los distintos procesos que se computan en ROS se denominan **nodos**. Dichos nodos se combinan formando una red que los intercomunica utilizando tres formas diferentes: haciendo una difusión mediante *topics*, llamando a un proceso remoto mediante servicios o leyendo datos de un listado centralizado de parámetros. (Estos tipos de comunicación se verán en detalle en la sección 2.3).

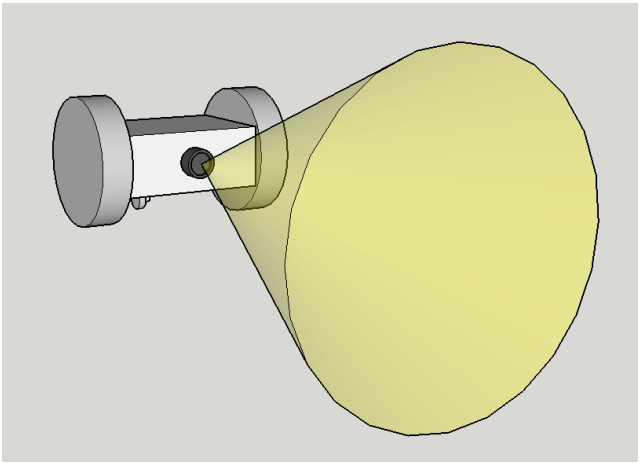


Figura 2.2: Ejemplo de robot simple.

Los nodos se encargan de separar las distintas funcionalidades del sistema. De este modo, cada uno de ellos realiza una tarea especifica dentro del con-

junto que conforma la red[1]. Por ejemplo, supongamos un robot como el de la figura 2.2. Se trata de un robot simple que consta de dos ruedas y una cámara frontal utilizada para detectar obstáculos en la dirección de avance. Un posible esquema de funcionamiento puede ser el siguiente: Un nodo puede encontrarse ejecutando los *drivers* de la cámara, mientras otro se encuentra procesando la imagen captada y transmitiendola a otro nodo que se encarga de identificar obstáculos y de enviar señales de STOP a un último nodo que controla los motores del robot.

El uso de nodos aporta varios beneficios al sistema. Al separarse en programas más pequeños, no se perciben los detalles específicos de cada nodo en el conjunto y de esta manera se puede tener una visión del sistema más clara y concisa. A su vez, como los nodos funcionan de forma independiente, el sistema adquiere una mayor tolerancia a los errores de funcionamiento que afectan a nodos aislados.

Para nombrar los diferentes nodos que forman el sistema se utiliza la siguiente nomenclatura: */node_name*. Se debe tener en cuenta que dos nodos no deben tener nunca el mismo nombre. Si, por motivos organizativos, se quieren agrupar a fin identificarlos bajo un nombre común, se pueden agrupar dentro de un paquete de ROS bajo un mismo tipo.

Los nodos se comunican intercambiando estructuras de datos llamadas **mensajes**. Dichos mensajes pueden estar compuestos por los tipos de datos estándar (integer, boolean, character, etc.) o por diversas estructuras de datos compuestas por estos mismos tipos. Un mensaje complejo puede a su vez contener más mensajes de modo que se aniden varias estructuras de datos. Un ejemplo de mensaje complejo puede ser *PoseWithCovariance.msg* el cual

[1]Un caso especifico es el nodo maestro, del cual se habla en detalle en la seccion 2.4

se encuentra entre los tipos de mensajes por defecto de ROS. Su estructura interna es la siguiente:

```
float64[36] covariance
geometry_msgs/Pose pose
        geometry_msgs/Point position
                float64 x
                float64 y
                float64 z
        geometry_msgs/Quaternion orientation
                float64 x
                float64 y
                float64 z
                float64 w
```

Como se puede observar, este mensaje esta compuesto por una matriz de tipo *float* y otro mensaje llamado *Pose*. Éste a su vez está compuesto por otros dos mensajes: *Point* y *Quaternion*. Por último tanto *Point* como *Quaternion* se componen de *floats*.

Este tipo de mensajes anidados permiten tener una visión rápida y ordenada de su contenido, lo cual es de gran utilidad a la hora de comprobar cualquier dato enviado por los nodos. Otra ventaja de este tipo de comunicación mediante mensajes es que se pueden crear mensajes con estructuras propias creadas por el desarrollador que incluyan la información que crea oportuna para cada uso.

En el ejemplo del robot con cámara, un posible mensaje para controlar el giro de las ruedas a la hora de evitar obstáculos podría ser éste:

```
boolean stop_izq
boolean stop_der
```

La información enviada en este mensaje será utilizada de modo que en el

momento que el robot detecte un obstáculo, gire en la dirección contraria al punto donde se encuentra dicho obstáculo (figura 2.3). Para transmitir esta información, el mensaje consta de dos datos booleanos cuya misión es indicar al nodo que controla los motores si debe detener la rueda izquierda, la derecha, las dos o ninguna.

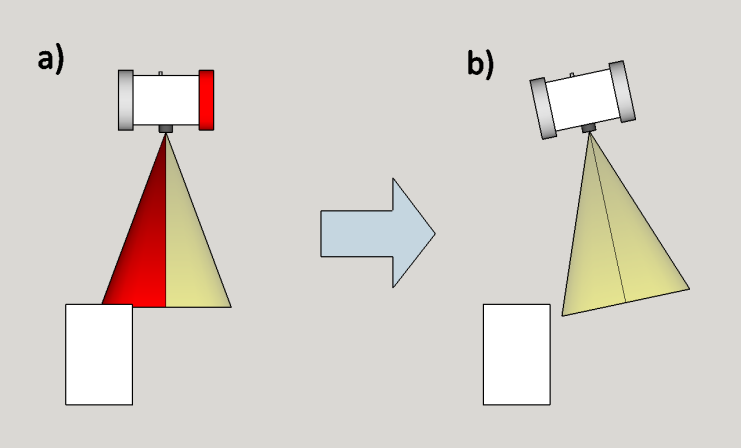


Figura 2.3: a)El robot detecta un obstáculo a su derecha y bloquea su rueda izquierda. b)El robot deja de detectar el obstáculo y desbloquea la rueda continuando su avance.

2.3. Topics y servicios

Una vez comprendidos los conceptos de nodo y mensaje es necesario comprender cómo se intercambian estos mensajes entre nodos. Existen dos vías posibles para dicho intercambio: los *topics* y los servicios.

Un **topic** es un flujo de datos de un único sentido que funciona siguiendo un modelo de publicador/suscriptor. Los topic se nombran siguiendo el mismo convenio que los nodos (*/topic_name*) Aquellos nodos en los cuales se genera o procesa información, publican en uno o varios topics mensajes con dichos datos y por otro lado los nodos interesados en esa información se suscriben

a ellos de modo que reciben esa información[2]. Es decir, el funcionamiento de los topics es similar al funcionamiento de una retransmisión de radio, donde el emisor retransmite en una determinada frecuencia y el receptor sintoniza dicha frecuencia para que recibir esa información a tiempo real.

Al igual que estas emisiones de radio, en general, los nodos que publican en un determinado topic no conocen la identidad de los nodos suscritos a él. Este anonimato entre nodos sirve para desacoplar la consumo de datos de su producción. En el ejemplo del robot anterior, el mensaje que se enviaba a las ruedas seria un caso de comunicación vía topic.

Esta vía de transmisión de mensajes entre nodos cubre gran parte de las necesidades básicas de comunicación, pero existe un caso en el que esto no sirve. Es el caso en el cual un nodo necesita una respuesta a una solicitud realizada a otro. Este tipo de comunicación se hace por medio de los **servicios**.

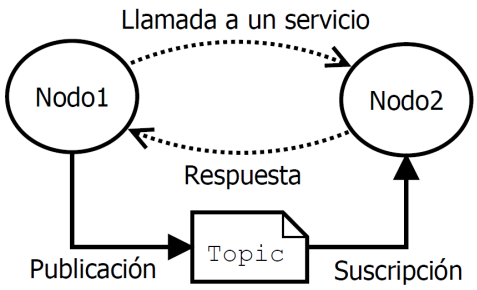


Figura 2.4: Ejemplo de topic y servicio (https://goo.gl/43AuCR)

Un servicio funciona con el intercambio de un mensaje compuesto por una solicitud y una respuesta. El nodo interesado en recibir la respuesta (cliente) envía ese mensaje a un nodo que está ofreciendo el servicio bajo un nombre específico. Cuando este último recibe dicho mensaje, procede a procesar la

[2]Un topic puede tener más de un suscriptor y más de un publicador

información recibida y envía los resultados al cliente. El ejemplo más sencillo de servicio es una suma, donde el cliente llama al servicio enviándole dos números y el nodo a cargo del servicio se encarga de sumarlos y devolver el resultado.

En la figura 2.4 se puede ver un esquema simplificado del funcionamiento de topics y servicios.

2.4. Maestro y servidor de parámetros

Existe un caso especifico de nodo que siempre debe estar en ejecución para el correcto funcionamiento del sistema, el **maestro**. El maestro es el encargado de llevar un registro de los nombres del resto de nodos y de hacer un seguimiento tanto de los topics como de los servicios del sistema. Esto es lo que permite a los nodos conocer qué otros nodos se encuentran en la red y comunicarse entre ellos.

Cuando un nodo quiere publicar en un topic, manda una solicitud al maestro el cual incluye ese topic en el sistema. Del mismo modo cuando un nodo quiere suscribirse a un topic primero se lo notifica al maestro. En este momento el maestro pone en comunicación ambos nodos (figura 2.5). Es por esto que sin el maestro no es posible la comunicación entre nodos.

Otro aspecto importante del nodo maestro es que en él se encuentra el **servidor de parámetros**. El servidor de parámetros es un diccionario de variables compartidas entre todos los nodos de la red. En él se almacenan datos estáticos del sistema tales como variables de configuración (parámetros de calibración, posición inicial, ejes de coordenadas, etc.). Esto es de gran utilidad porque, entre otras cosas, permite tener una visión global del estado actual del sistema. Los parámetros se nombran siguiendo el mismo convenio que los nodos y los topics. Para cargar estos parámetros en el servidor, un

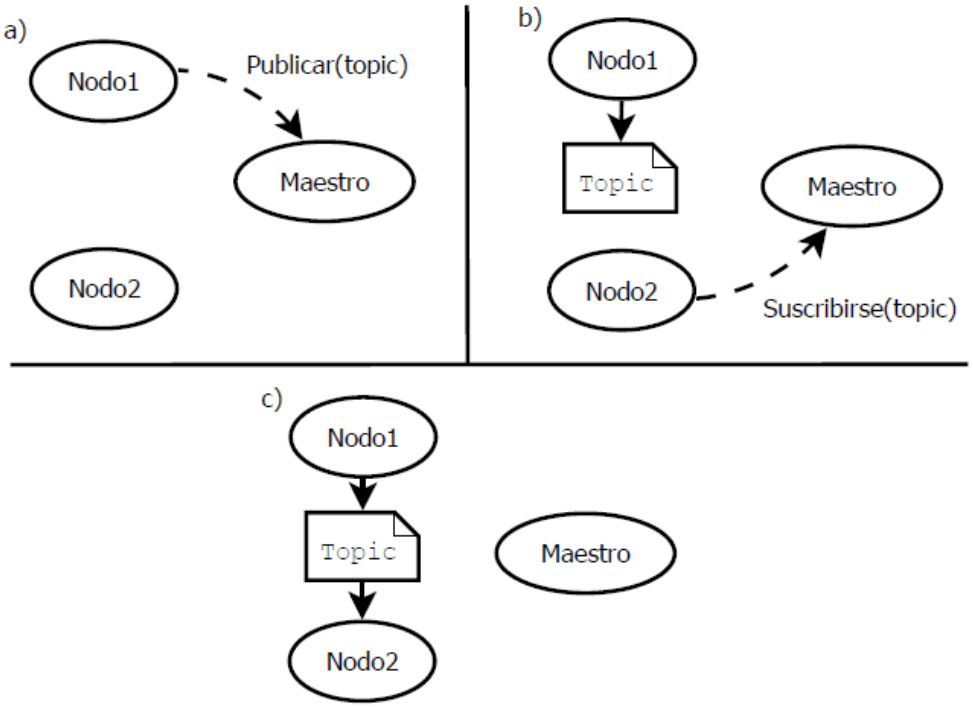


Figura 2.5: Función del maestro

método muy utilizado es el uso de ficheros con la extension .yaml, donde se organizan siguiendo una estructura fija.

2.5. Transformadas (TF's) y sistemas de coordenadas

Uno de los aspectos más importantes de ROS para este proyecto es el tratamiento que tiene éste para los sistemas de coordenadas.

Un robot puede contar con numerosos sistemas de coordenadas en puntos clave tales como su cabeza, sus brazos, sensores, etc. En muchos casos tener una visión clara de la posición de unas partes con respecto a otras facilita la programación de tareas que implican movimiento. Por ejemplo, un brazo robot

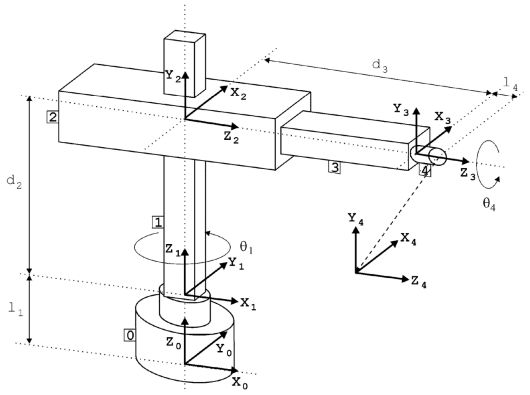


Figura 2.6: Sistemas de coordenadas brazo (http://goo.gl/vBlwEw)

como el de la figura 2.6 puede tener un sistema de coordenadas en su base desde donde se controlan todos su movimientos en el mundo. Pero conocer la posición de esta base no es suficiente para manipular correctamente su extremo. Para ello es necesario tener al menos otro sistema de coordenadas en este extremo y en cada uno de los eslabones que forman el brazo. De este modo se tiene el control completo de la posición de cada parte del brazo por separado y, utilizando transformadas entre cada uno de los sistemas de coordenadas, se puede conocer su posición en el mundo. Estas transformadas son la base del sistema de tf's de ROS[3].

El sistema de tf's organiza los sistemas de coordenadas y las transformadas necesarios para modelar el robot en su totalidad. Esta organización sigue una estructura de tipo árbol (ejemplo en la figura 2.7). Otra característica de las tf's es que, además de permitir conocer los datos de los sistemas de coordenadas a tiempo real, también guardan un registro de la posición de dichos sistemas en el tiempo. De este modo se pueden hacer referencias a posiciones anteriores de una operación. Por ejemplo llevar el brazo robot a su posición inicial.

En ROS se utilizan los topics para compartir la información de las trans-

[3]Todo este sistema se encuentra implementado en el paquete tf de ROS.

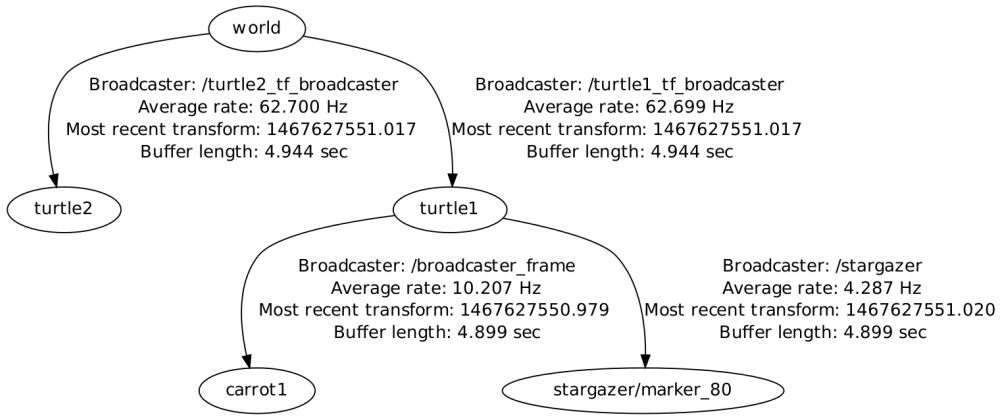


Figura 2.7: El sistema de coordenadas *world* es la referencia que tienen *turtle*1 y *turtle*2 para saber su posición en el mundo. A su vez *carrot* y *stargazer/marker*_80 toman su posición de turtle1.

formadas entre nodos. Mientras algunos nodos publican la información de las transformadas, otros nodos se suscriben a los topics que llevan dicha información formando una imagen del conjunto del robot. De este modo la información de las transformadas se comporta siguiendo la filosofía de ROS: mantener todo lo mas simple posible y modular. Los sistemas de coordenadas se nombran mediante una cadena de caracteres del modo más representativo posible, por ejemplo: *codo_derecho*. El nombre elegido para cada sistema de coordenadas puede ser cualquiera siempre y cuando no se repita [15].

Los mensajes que se transmiten en el topic */tf* son del tipo *tf/tfMessage*, el cual se trata de un tipo de mensaje complejo que incluye toda la información relacionada con las transformadas y su posición en el árbol y tiene la siguiente estructura:

```
//Vector de transformadas
  geometry_msgs/TransformStamped[] transforms
          std_msgs/Header header
```

```
            uint32 seq
            time stamp
            string frame_id
        string child_frame_id
        geometry_msgs/Transform transform
            geometry_msgs/Vector3 translation
            geometry_msgs/Quaternion rotation
```

La figura 2.8 muestra un ejemplo de mensaje en el topic $/tf$ en la consola de Ubuntu, concretamente la transformada del ejemplo de la figura 2.7 que hay entre el sistema de coordenadas del mundo (*world*) y el sistema de coordenadas *turtle*1.

```
transforms:
  -
    header:
      seq: 0
      stamp:
        secs: 1473175219
        nsecs: 590393721
      frame_id: world
    child_frame_id: /turtle1
    transform:
      translation:
        x: 5.544444561
        y: 5.544444561
        z: 0.0
      rotation:
        x: 0.0
        y: 0.0
        z: 0.0
        w: 1.0
---
```

Figura 2.8: Ejemplo de tfMessage.

El mensaje (para una única transformada) se compone de tres campos principales. El primer campo (header) es donde se muestran el momento en el que se ha tomado el dato (stamp) y el nombre del sistema de coordenadas de origen (frame_id). El segundo campo es el nombre del sistema de coordenadas al que se hace la transformada. Por último el tercer campo es la transformada

propiamente dicha compuesta de traslación en los ejes x, y, z y la rotación en cuaternios de un sistema con respecto al otro.

Capítulo 3

Integración del Stargazer

En este capitulo se realiza tanto una descripción del dispositivo que se va a integrar, el Stargazer, como una breve descripción del paquete que contiene el software del dispositivo. A su vez, se explica el proceso de integración del dispositivo en el robot.

3.1. Descripción del dispositivo

El dispositivo Stargazer (figura 3.1), es un sensor óptico de posición pensado para utilizarse en la localización de robots móviles inteligentes en interiores [7]. Su funcionamiento se basa en la detección de infrarrojos reflejados en marcadores pasivos mediante una cámara.

El sensor cuenta con una cámara en su parte superior mediante la cual detecta luz infrarroja. Alrededor de ésta, hay un anillo de LED's emisores de infrarrojos. En la versión del dispositivo utilizada, toda la electrónica se encuentra en el interior de un encapsulado metálico. En los laterales de dicho encapsulado se encuentran un interruptor que activa la alimentación, la entrada de esa alimentación y un puerto serie por el cual se puede comunicar con otros

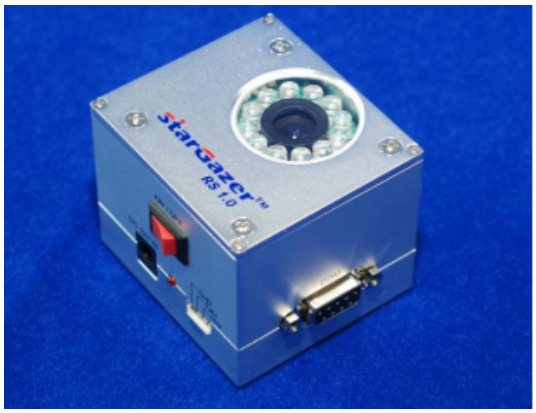


Figura 3.1: Dispositivo Stargazer.

dispositivos.

El dispositivo funciona de la siguiente manera (figura 3.2): los LED's emiten un haz de luz infrarroja hacia el techo, donde se refleja en unos marcadores pasivos con un patrón determinado. La cámara detecta la luz reflejada en el marcador y es capaz de identificar su patrón y el ángulo de incidencia del haz de luz. Basándose en estos datos, el dispositivo es capaz de calcular a qué marcador pertenece ese patrón y cual es su posición y orientación [11].

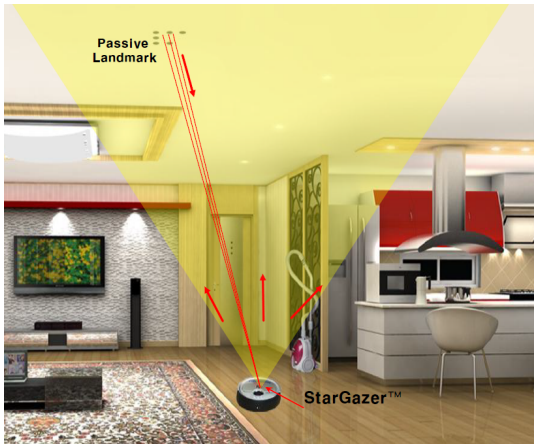


Figura 3.2: Funcionamiento del sensor (http://goo.gl/eqUzHz)

El dispositivo tiene dos modos de funcionamiento: modo solitario (alone

mode) y modo mapeo (map mode). En el modo solitario, el dispositivo envía continuamente información relativa al marcador detectado. Esta información está formada por el ID del marcador y la posición y la orientación del dispositivo con respecto al marcador. En el modo mapeo el dispositivo almacena el ID y posición de varios marcadores creando un mapa con distintos puntos. La comunicación con el dispositivo se lleva a cabo por medio de comandos enviados por el puerto serie. Estos comandos definen el modo de funcionamiento del sensor así como los parámetros de configuración del mismo.

Las especificaciones técnicas más relevantes facilitadas por el fabricante son las siguientes:

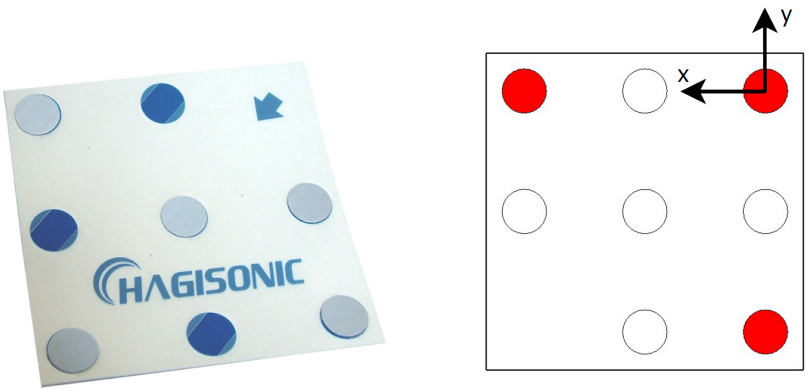
- Tamaño: $50x50x28$ mm
- Rango de la cámara (por marcador): 2.5∼5 m de diámetro (para alturas entre 2∼6 m)
- Precisión: 2 cm
- Interfaz hardware: UART(TTL 3.3V), 115.200bps
- Consumo: 5V 300mA, 12V 70mA
- Velocidad de muestreo: 10 muetras/s

La ficha técnica completa se puede consultar en el anexo C.

3.2. Descripción de los marcadores

Los marcadores son etiquetas cuadradas de PVC con puntos de material reflectante que forman patrones en forma de matriz cuadrada. Existen dos

familias principales de marcadores: los que tienen una matriz $3x3$ (HLD-S) y los que tienen una matriz $4x4$ (HLD-L)[1]. Dentro de estas dos familias los distintos marcadores se encuentran clasificados según las necesidades de altura del entorno donde se va a utilizar el dispositivo (cuadro 3.2). En este caso, únicamente se van a utilizar marcadores del tipo HLD1-S (figura 3.3.a) ya que se van a utilizar menos de 31 marcadores diferentes y la altura del techo no sobrepasa los 2.5 m.



(a) Fotografía (http://goo.gl/LsKu7Q) (b) Esquema

Figura 3.3: Fotografía y esquema del marcador HLD1-S

En la matriz de estos marcadores , en una de las esquinas siempre falta un punto (esta esquina se encuentra señalizada con una flecha). Los puntos de las tres esquinas restantes son utilizados para determinar la posición y orientación del marcador. El origen sistema de coordenadas del marcador se encuentra en el centro del punto de la esquina opuesta a la esquina vacía. Los puntos restantes de la matriz son los que forman el identificador del marcador (figura 3.3.b) [10].

[1]La única diferencia entre estas dos familias es el número de ID's que pueden tener.

HLD-S (31 combinaciones)		HLD-L (4095 combinaciones)	
HLD1-S	1.1∼2.9 m	HLD1-L	1.1∼2.9 m
HLD2-S	2.9∼4.5 m	HLD2-L	2.9∼4.5 m
HLD3-S	4.5∼6 m	HLD3-L	4.5∼6 m

Tabla 3.1: Clasificación de marcadores

3.3. Drivers

A fin de utilizar el dispositivo Stargazer junto con ROS, se ha utilizado un paquete que contiene el software necesario para realizar la comunicación con el sensor, es decir, sus *drivers*. Utilizando la información captada sobre los marcadores junto con la información especificada en un fichero de parámetros, este software se encarga de calcular las transformadas necesarias para obtener la posición del robot en el que se integre el Stargazer con respecto al mundo. Este paquete ha sido desarrollado por la "Carnegie Mellon University" en Pittsburg (EEUU)[2]

Los parámetros utilizados para configurar el modo de funcionamiento del dispositivo y el tipo de marcadores que utiliza son los siguientes:

- ThrVal: Nivel de umbral utilizado para evitar las distorsiones externas producidas en la imagen que capta el sensor. Depende del entorno. Su valor recomendado esá entre 210 y 240.
- MarkHeight: Distancia vertical entre el Stargazer y el marcador. Se utiliza para introducir manualmente este valor en caso de no querer que el dispositivo lo calcule automáticamente a fin de evitar errores.
- IDNum: Numero de marcadores utilizados para definir el mapa del modo mapeo.

[2]Puede encontrarse en este enlace: https://github.com/personalrobotics/stargazer

- RefID: ID del marcador de referencia en el modo de mapeo. Aquel que se utiliza como origen de coordenadas del mapa.
- Version: Versión del firmware.
- ThrAlg: Parámetro que determina cómo se obtiene el valor del parámetro ThrVal. Puede ser Manual o Automático.
- MarkType: Define cual de las dos familias de marcadores se está utilizando.
- MarkDim: Define cual es el tipo de marcador dentro de la familia seleccionada.
- MapMode: Determina si el modo de mapeo se encuentra activo o no.
- MarkMode: Determina si el modo en solitario se encuentra activo o no.

Todos estos parámetros no se encuentran en ningún fichero, sino que se configuran a mano sobre el código de los *drivers*. Cuando el software se inicializa, carga todos estos parámetros en el dispositivo a través de la comunicación serial. Dado que el modo en el que se ha utilizado el dispositivo es el modo solitario, los parámetros referentes al modo mapeo no se han utilizado.

Durante la inicialización del software, se cargan los parametros que definen los nombres de las tf's del robot y el dispositivo en el servidor de parámetros de ROS, junto con la información correspondiente a la posición de los marcadores en el mundo. Estos parámetros se encuentran definidos en un fichero con la extension *.yaml*.

Este fichero tiene el siguiente formato interno:

```
device:
fixed_frame_id:
robot_frame_id:
stargazer_frame_id:
marker_frame_prefix:
covariance: [
              , 0.   , 0.   , 0.   , 0.   , 0.   ,
         0.   ,      , 0.   , 0.   , 0.   , 0.   ,
         0.   , 0.   ,      , 0.   , 0.   , 0.   ,
         0.   , 0.   , 0.   ,      , 0.   , 0.   ,
         0.   , 0.   , 0.   , 0.   ,      , 0.   ,
         0.   , 0.   , 0.   , 0.   , 0.   ,      ,
]
marker_map:
         'ID1':
         - [ , , , ]
         - [ , , , ]
         - [ , , , ]
         - [0,0,0,1]
         'ID2':
         - [ , , , ]
         - [ , , , ]
         - [ , , , ]
         - [0,0,0,1]

         .
         .
         .
```

Cada uno de estos parámetros, definen lo siguiente:

- *device*: Este parámetro hace referencia al puerto del ordenador al que se encuentra conectado el dispositivo.
- $fixed\_frame\_id$: Define el nombre del marco correspondiente al sistema de referencia del mundo sobre el que se define la posición de los marcadores.

- *robot_frame_id*: Define el nombre del marco correspondiente al sistema de referencia del robot, es decir, aquel que define su posición.
- *stargazer_frame_id*: Define el nombre del marco correspondiente al sistema de referencia del Stargazer con el eje Z en la dirección de la cámara.
- *marker_frame_prefix*: Define el prefijo que se utiliza junto con la ID de los marcadores detectados para nombrar sus sistemas de referencia.
- *covariance*: Matriz 6x6 de covarianza.
- *marker_map*: Relaciona la ID de cada marcador utilizado con su posición en el sistema de coordenadas definido por *fixed_frame_id*. Esta posición se define mediante una transformada homogénea en forma de matriz 4x4. En esta matriz se encuentra definida una submatriz (r) 3x3 con la orientación del marcador con respecto al eje Z y su desplazamiento en los ejes X, Y, Z.

$$\begin{pmatrix} r_{11} & r_{12} & r_{13} & X \\ r_{21} & r_{22} & r_{23} & Y \\ r_{31} & r_{32} & r_{33} & Z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Una vez que estos parámetros se encuentran en el servidor de parámetros de ROS, el programa queda a la espera de que el Stargazer envíe la información de algún marcador. En el momento en el que un marcador es detectado, se ejecuta el siguiente flujo de tareas:

1. En primer lugar se comprueba que el marcador detectado se encuentra

entre aquellos definidos en el fichero *.yaml*. Si no es así se muestra un mensaje por pantalla advirtiendo ésto.

2. Acto seguido se busca en el topic encargado de las tf's la transformada que relaciona el stargazer con el robot, avisando en caso de no encontrarla.

3. Por último, utilizando esta transformada junto con los datos enviados por el Stargazer y los datos que había en el fichero *.yaml* relativos al marcador detectado, se obtiene la transformada que relaciona la posición del robot y el marco definido como referencia del mundo.

$$Robot -> Marcador \qquad Marcador -> Mundo$$

$$Robot -> Mundo$$

La información relativa a esta posición (su transformada), se envía al topic *robot_pose* dentro del espacio de nombres *stargazer*. De forma paralela se calcula una predicción de la posición del robot con respecto a los demás marcadores definidos y se publica en el topic *robot_pose_array*[3], también en el espacio de nombres *stargazer*.

En la figura 3.4 se puede observar el diagrama de flujo que refleja el funcionamiento de los *drivers*. En él se incluyen los flujos de datos externos, tanto de entrada como de salida.

[3]Los datos enviados este topic no se han utilizado en este proyecto, pero se han mantenido en el código de los *drivers* ya que pueden ser de utilidad en aplicaciones futuras.

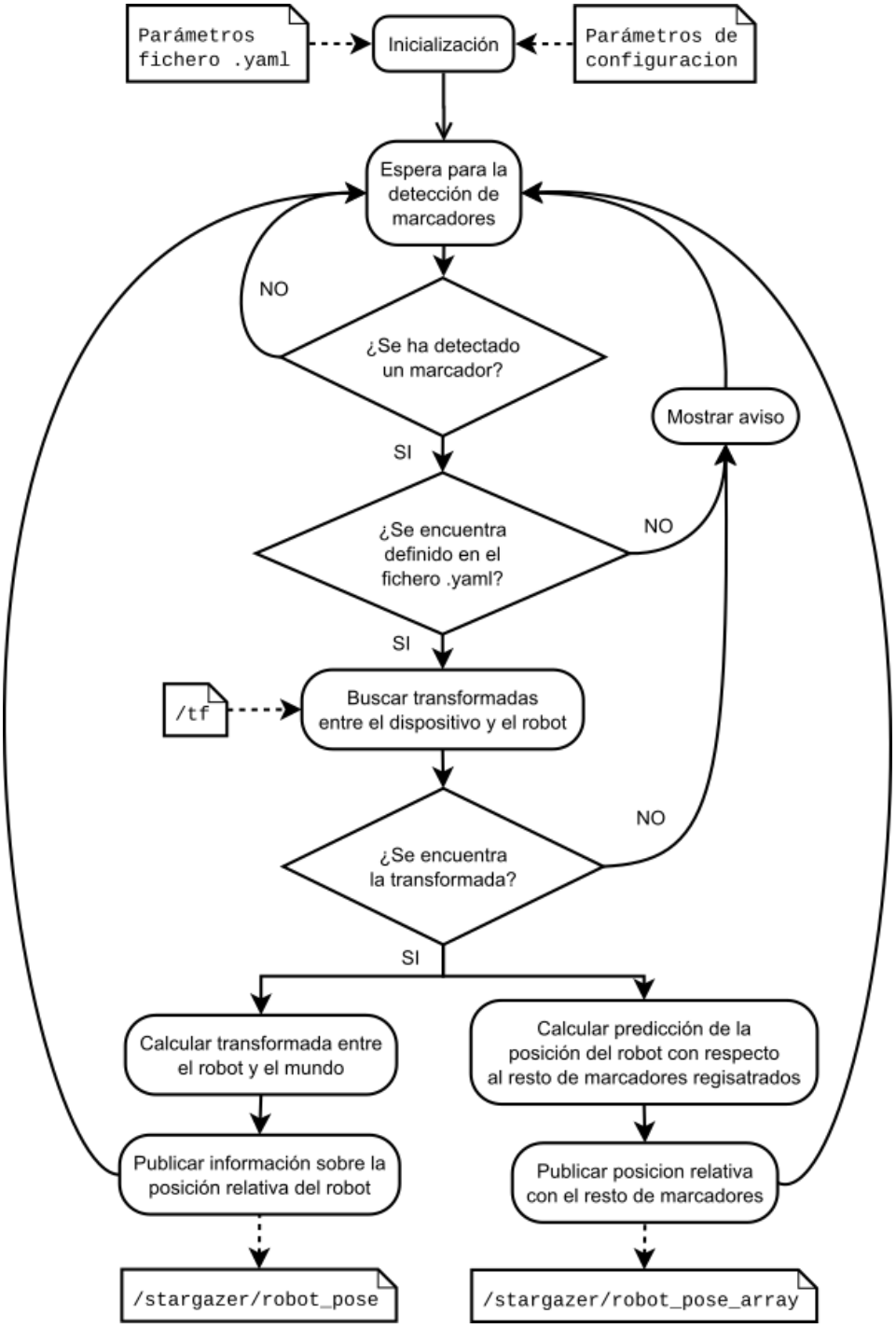


Figura 3.4: Diagrama de flujo del funcionamiento de los *drivers*

3.4. Integración en el robot

Se ha procedido a la instalación del dispositivo y sus paquetes sobre el robot Mbot. Para ello es necesario introducir físicamente el dispositivo en el robot y adaptar el software a los parámetros y espacios de nombres que utiliza el sistema ROS del robot.

3.4.1. Integración del Hardware

La instalación física del dispositivo es simple. El dispositivo se sitúa sobre un bastidor en el interior de la cabeza del robot de modo que la lente de la cámara apunte en dirección vertical. A este bastidor llegan tanto el cable con la alimentación para el Stargazer, como el cable que se utiliza para realizar la comunicación en serie con el ordenador de navegación del robot. Esta comunicación parte del puerto serie del dispositivo y pasa por un conversor a USB antes de conectarse al ordenador (figura 3.5). En la tapa que cubre la cabeza del robot hay un agujero por el que, tanto la cámara del dispositivo como el anillo de LED's infrarrojos, dan al exterior. En la figura 3.6, se muestra cómo queda el dispositivo una vez instalado en el robot.

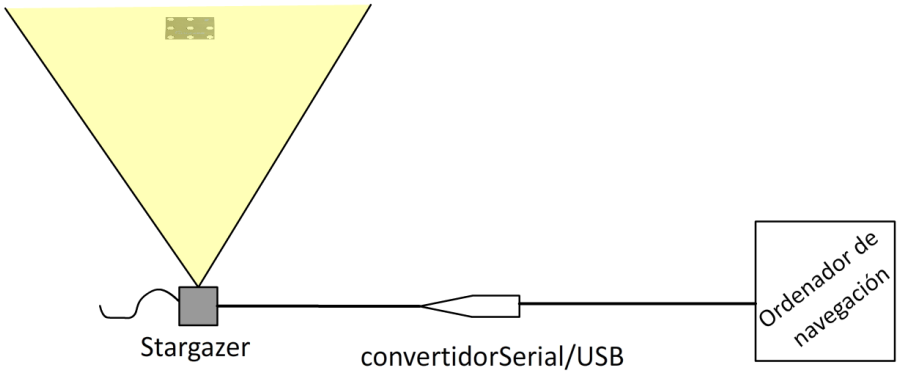


Figura 3.5: Esquema de conexión serial del Stargazer

(a) Cabeza abierta (b) Cabeza cerrada

Figura 3.6: Stargazer en el la cabeza del Mbot.

3.4.2. Integración del Software

Una vez instalado el dispositivo en el robot, se ha procedido a la instalación del paquete de ROS que incluye sus *drivers* en el ordenador de navegación (donde está conectado el dispositivo). Antes de poder ejecutarlo, es necesario realizar ajustes tanto sobre el programa y como sobre sus ficheros y para ello es necesario conocer cómo se encuentra definido el sistema de tf's del robot. Este sistema está definido por el fabricante.

El robot Mbot tiene dos sistemas de referencia principales. El primero de ellos, llamado *base_link*, se encuentra en la base del robot y es el utilizado para conocer su posición. A este sistema se encuentran referenciados todos los demás del robot. El segundo sistema de referencia principal, llamado *head*, es hijo del anterior y se encuentra en la cabeza del Mbot. A éste se referencian todos los dispositivos de la cabeza del robot.

El primer paso para integrar el software es definir un nuevo sistema de referencia en el lugar donde se sitúa el dispositivo. Este sistema de referencia se ha definido como hijo del sistema *head* y su nombre es *stargazer*. Ésto se ha hecho creando un nodo que define la transformada entre *head* y *stargazer*

desde el archivo de lanzamiento de los *drivers*.

El siguiente paso es definir en el fichero *stargazer.yaml* tanto el puerto al que se encuentra conectado el Stargazer, como los nombres de los sistemas de referencia del robot y como los identificadores de los marcadores que se van a utilizar y su posicion en el mundo. El fichero de configuración utilizado es el siguiente:

```
device: /dev/ttyUSB2
fixed_frame_id: map
robot_frame_id: base_link
stargazer_frame_id: stargazer
marker_frame_prefix: stagazer/marker_
covariance: [
        0.14, 0.   , 0.   , 0.   , 0.   , 0.   ,
        0.   , 0.14, 0.   , 0.   , 0.   , 0.   ,
        0.   , 0.   , 0.14, 0.   , 0.   , 0.   ,
        0.   , 0.   , 0.   , 0.14, 0.   , 0.   ,
        0.   , 0.   , 0.   , 0.   , 0.14, 0.   ,
        0.   , 0.   , 0.   , 0.   , 0.   , 0.14,
]
marker_map:
        '80':
        - [1.0, 0.0, 0.0, 0.000]
        - [0.0, 1.0, 0.0, 0.000]
        - [0.0, 0.0, 1.0, 2.000]
        - [0.0, 0.0, 0.0, 1.0  ]
```

Los nodos y *topics* de cada robot se encuentran bajo un espacio de nombres que los hace unívocos con el fin de evitar colisiones de nombres con otros robots que pudieran estar operando al mismo tiempo. En el caso del MBot, este espacio de nombre es */mbot*10 y por lo tanto es necesario integrar todos sus nodos y *topics* en él. Para ello se lleva a cabo una reasignación de estos utilizando el comando *remap* en su archivo de lanzamiento (*stargazer_launcher.launch*) de modo que, al ejecutar el software del dispositivo, todos ellos se encuentren

en ese dentro de */mbot*10.

El archivo de lanzamiento tiene la estructura mostrada en la figura 3.7. Como se puede observar en esta figura, este archivo es el encargado de ejecutar tres nodos: el nodo de los *drivers* (stargazer), el nodo de relocalización y el nodo encargado de definir la transformada del dispositivo con respecto a la cabeza del robot (sg_broadcaster). En este archivo tambien se encuentran las lineas de comandos utilizadas para determinar los espacios de nombres utilizados (*remap*).

```
<launch>
        <remap from="tf" to="mbot10/tf"/>

                <node pkg="tf" type="static_transform_publisher" name="sg_broadcaster" args= "0 0 0 0 0 1 0 head stargazer 100"/>

                <node pkg="stargazer" type="stargazer_publisher.py" name="stargazer"
                        clear_params="true" output="screen" respawn="true">
                        <remap from="robot_pose" to="stargazer/robot_pose"/>
                        <remap from="robot_pose_array" to="stargazer/robot_pose_array"/>
                        <rosparam command="load" file="$(find stargazer)/stargazer.yaml"/>
                </node>
                <node pkg="stargazer" type="relocalizacion.py" name="relocalizacion" output="screen">
                        <remap from="initialpose" to="mbot10/initialpose"/>
                </node>

</launch>
```

Figura 3.7: Archivo *stargazer_ launcher.launch*

Capítulo 4

Aplicacion de relocalización

En este capitulo se muestra la aplicación pensada para mejorar el funcionamiento del robot utilizando el sistema desarrollado para detectar marcas en el entorno. Este trabajo se ha centrado en aplicaciones relacionadas con la navegación, concretamente en la relocalización del robot en el mundo.

El robot Mbot tiene un sistema de navegación que utiliza para desplazarse de manera autónoma. Este sistema de navegación, utiliza dos sensores láser (LRF) que cubren 360º y un sensor RGB-D de profundidad para la detección de obstáculos. En la memoria del robot hay una imagen del plano del edificio que coordinada con los datos que el robot obtiene del entorno por medio de sus sensores y la información sobre el avance de las ruedas, estiman la posición del robot en sistema de coordenadas del mundo. En algunas ocasiones, estos datos difieren provocando que el robot estime incorrectamente su posición con respecto al sistema de coordenadas de referencia. Esto origina que el robot pierda su ubicación en el plano y calcule sus trayectorias de forma errónea.

Este es el caso que se muestra en la figura 4.1. En este caso se ha pedido al robot que se dirija al punto A señalado en el plano. En la imagen de la izquierda se puede ver una representación del mundo a ojos de la navegación.

Según los datos del plano, el robot ha llegado al cruce de la imagen desde el pasillo 1. Dado que para la navegación, esa es su posición actual, la trayectoria calculada para llegar a su destino es una linea recta. En la imagen de la derecha, se puede ver el caso real donde el robot se encuentra desubicado y en lugar de venir desde el pasillo 1, viene desde el pasillo 2[1]. Como la trayectoria calculada por la navegación es una linea recta, el robot continuara avanzando en esa dirección hasta encontrarse con una pared, la cual intentar esquivar como si se tratase de un obstáculo volviendo a calcular la trayectoria.

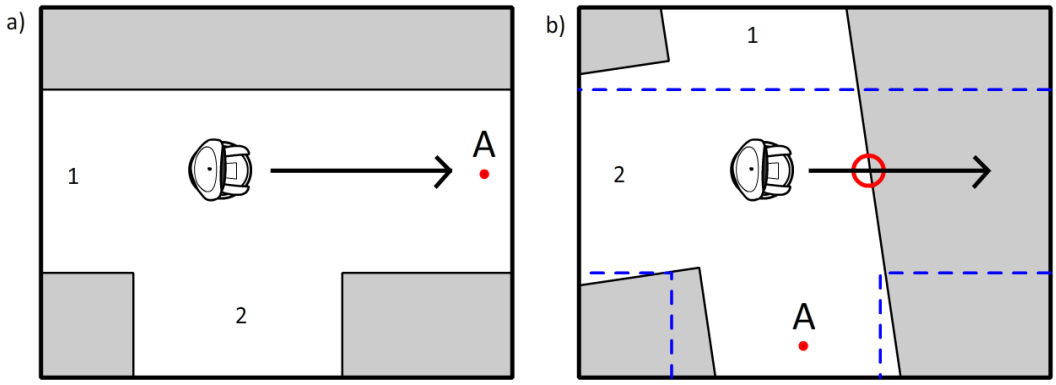


Figura 4.1: a) Orientación según la navegación. b)Orientación real.

El objetivo de la aplicación de relocalización es utilizar la capacidad que tiene el dispositivo Stragazer de obtener la posición exacta del robot con respecto de un marcador y de determinar su posición en el plano para reubicar al robot. Para ello se colocarán marcadores en puntos clave del edificio tales como cruces, puertas o ensanchamientos. Estas son zonas por las que de un modo u otro acabará pasando el robot. En el momento en el que el robot detecte cualquiera de estos marcadores, se corregirá automáticamente su posición en el plano con respecto al sistema de coordenadas del mundo.

Para desarrollar esta aplicación, es necesario saber la posición exacta de

[1]Para mayor claridad se ha representado con linea azul discontinua el cruce desde el punto de vista del robot desorientado.

cada marcador en el mundo. Esta información es enviada a los drivers del Stargazer junto con los demás parámetros de configuración en el fichero correspondiente.

La función de la aplicación es obtener la posición del robot en el mundo. Este dato lo publica el software del Stargazer en un topic llamado */robot_pose*. Una vez obtenida, la aplicación la envía esta información al sistema de navegación por medio del topic */initial_pose* para que éste defina esta posición como su posición actual en el mapa.

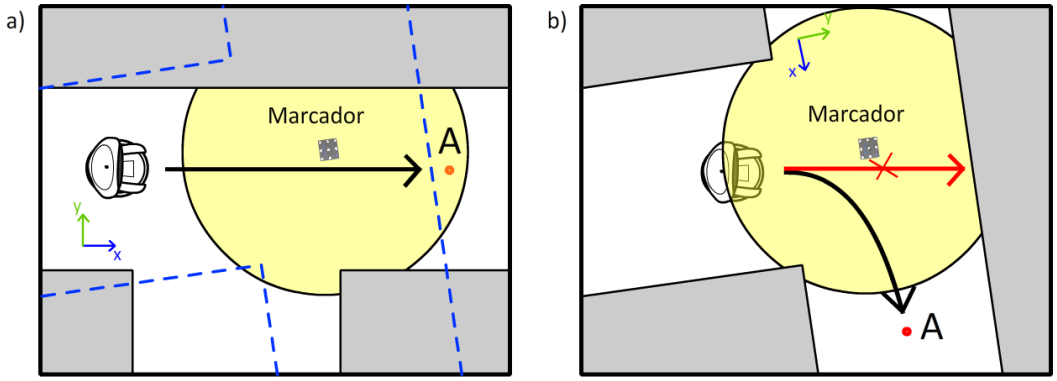


Figura 4.2: Función de Relocalización.

En la figura 4.2 se muestra el mismo caso que en la figura 4.1, pero esta vez utilizando la aplicación de relocalización en combinación con el sistema de navegación. En la imagen de la izquierda se puede ver cómo, al igual que en el caso anterior, la navegación sitúa erróneamente el robot en el plano y calcula su trayectoria hacia el punto A en linea recta. Esta vez se ha colocado un marcador en el cruce de modo que el área de detección abarque toda la intersección. En la imagen de la derecha se ve cómo esto produce que en el momento en el que el Stargazer detecta el marcador y obtiene su posición, la aplicación obtiene dónde está situado el robot en el mundo y pasa esta nueva información a la navegación. Utilizando estos datos la navegación desecha la

trayectoria errónea anterior y traza una nueva que llevará correctamente al robot hasta el punto A.

Capítulo 5

Pruebas

A lo largo del proyecto se han realizado diversas pruebas para comprobar el funcionamiento del dispositivo y las aplicaciones desarrolladas. En este capítulo se muestra el diseño de estas pruebas y los resultados obtenido en ellas.

5.1. Precisión

Las especificaciones técnicas del dispositivo Stargazer muestran que el error esperado en la precisión de éste a la hora de determinar la posición de un marcador es de 2 cm aproximadamente. Para comprobar que ésto se cumple se ha diseñado una prueba de precisión que se ha llevado a cabo sobre dos programas diferentes: una sobre el programa que propone el fabricante para hacer funcionar el dispositivo en Windows y la otra sobre los drivers para ROS en Ubuntu 12.04[1]. Con esta prueba se pretende comprobar el error en la estimación de la posición tanto en el eje vertical (eje z) como en el eje radial horizontal (eje x). De forma adicional se ha comprobado si existe error en la orientación del marcador.

[1]Ésto se a hecho así a fin de comprobar que no se presentan errores debidos al software de los drivers.

La prueba consiste en lo siguiente (figura 5.1):

- Se ha situado el Stargazer en una posición fija. En este caso, para mayor comodidad a la hora de realizar las pruebas, se ha elegido realizar la prueba con el dispositivo apoyado en el suelo en posición horizontal, de modo que el eje z (en dirección saliente de la cámara) se encuentre paralelo al suelo.

- Se han definido distintas medidas a intervalos fijos desde la posición del Stargazer, concretamente una marca cada 60 cm.

- Sobre estas marcas se ha situado un marcador de tipo HLD1-S de forma perpendicular al eje z haciendo coincidir su origen de coordenadas con este eje.

- Para cada una de las posiciones definidas, se han tomado los datos mostrados por pantalla tres veces, realizando la media de éstos para cada marca.

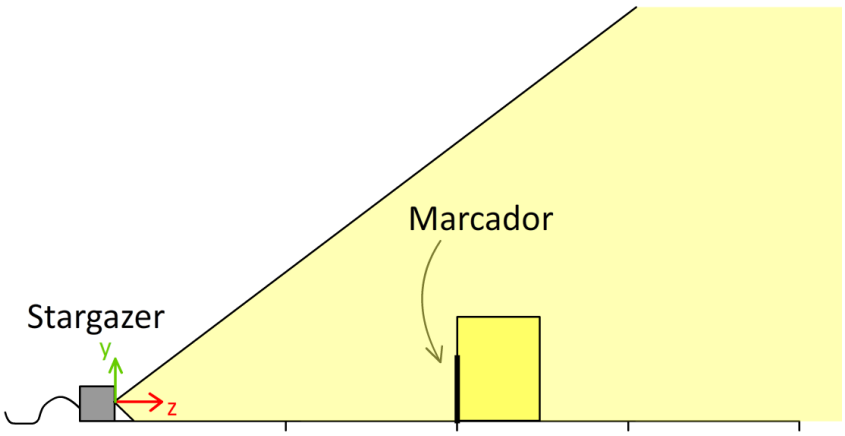


Figura 5.1: Diagrama de la prueba de precisión

Drivers en Ubuntu 12.04				
Desplazamiento en z	**60,00**	**120,00**	**180,00**	**240,00**
Desplazamiento en x	**0,00**	**0,00**	**0,00**	**0,00**
x	1,88	5,66	9,39	12,82
z	70,16	139,59	209,38	277,83
Rotación en z	-1,30	0,15	2,23	0,70

Tabla 5.1: Resultados sobre los drivers en Ubuntu 12.04

Software del fabricante (Windows)				
Desplazamiento en z	**60,00**	**120,00**	**180,00**	**240,00**
Desplazamiento en x	**0,00**	**0,00**	**0,00**	**0,00**
x	1,85	5,56	9,40	12,83
z	70,66	139,57	208,90	276,50
Rotación en z	0,18	1,40	-1,50	0,50

Tabla 5.2: Resultados sobre el software del fabricante (Windows)

Los resultados obtenidos para cada uno de los programas se muestran en las tablas 5.1 y 5.2. Tal y como se muestra en estas tablas, el error obtenido es bastante mayor al esperado. Se ha observado que para el Stargazer disponible en el laboratorio se muestra un error ascendente de aproximadamente 10 cm cada 60 cm para la medida del eje z y un error ascendente de aproximadamente 4 cm para la del el eje x.

Se ha observado que el Stargazer utilizado para estas pruebas presenta un problema en el funcionamiento del anillo de LED's. Como se muestra en la figura 5.2, un tercio del anillo de LED's no se enciende. El fabricante ha informado que este hecho podría ser el causante del error.

Pese a que la diferencia en el error es tan grande con respecto a las especificaciones del dispositivo, se ha mantenido el uso del sensor en el desarrollo del proyecto. Esto se ha podido llevar a cabo dado que el error en la altura no es relevante en la aplicación que se le quiere dar al dispositivo. Esto es así por que el sistema de navegación sólo tiene en cuenta el plano sobre el que se

Figura 5.2: Mal funcionamiento de los LEDS del Stargazer.

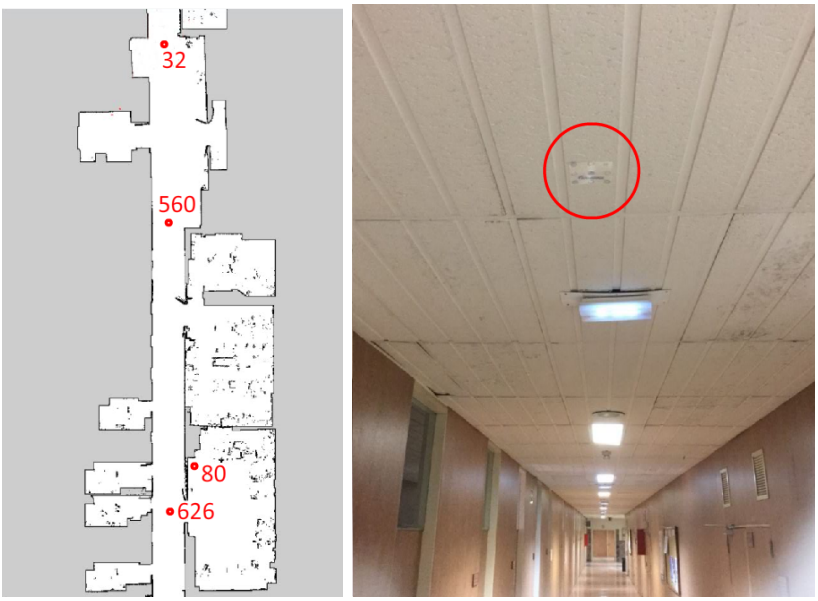
desplaza el robot. Por otro lado, el error angular es mínimo y el error en el eje x no es tal como para suponer un problema en la localización del robot.

5.2. Funcionamiento de las aplicaciones

Para comprobar que la aplicación realizada funciona correctamente y que el dispositivo responde como se espera, se han realizado pruebas de funcionamiento haciendo circular al robot por las instalaciones de la universidad. Para ello se han situado marcadores en los pasillos tal y como se muestra en la figura 5.3.b. En total se han situado tres marcadores en el pasillo principal y un cuarto sobre la base de carga del robot en el interior del laboratorio. En la figura 5.3.a se muestra la posición y la ID de estos marcadores sobre el plano que utiliza la navegación.

El marcador 80 es el que se encuentra sobre el cargador del Mbot. Los marcadores 32 y 560 se han posicionado en ensanchamientos del pasillo y el 626 frente a la puerta del laboratorio.

Para monitorizar el proceso y dar las ordenes al robot se ha utilizado la interfaz web que posee el sistema de navegación. En esta interfaz se puede



(a) Situación de los marcadores sobre el plano.

(b) Marcador en el pasillo.

Figura 5.3: Colocación de los marcadores.

observar el plano de la navegación y la posición del robot en este. A su vez también se muestra de manera superpuesta la silueta del entorno que detecta el robot por medio de los sensores láser. Cuando el robot se encuentra correctamente localizado, la silueta del plano coincide con la obtenida por los sensores.

Si el robot se encuentra bajo la influencia de algún marcador, el Stargazer envía de manera continua su posición respecto a él. Durante este tiempo, en la interfaz web, el robot se ve representado por una nube de puntos azules. En el momento que el robot abandona el radio de detección de las marcas, la información de su posición la obtiene únicamente de los encoders de sus ruedas y de la percepción del láser y su representación pasa a ser mas compacta.

5.2.1. Prueba de relocalización con un solo marcador

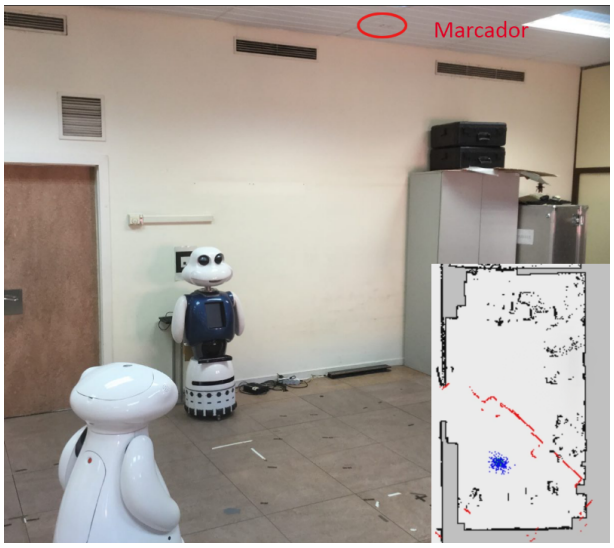
La primera prueba se ha realizado en el interior del laboratorio por lo que solo se ha utilizando el marcador 80 en el puesto de carga del robot. La prueba ha consistido en mover al robot hasta una posición alejada del radio de detección del marcador y forzar su deslocalización mediante la interfaz web. A continuación se ha controlado manualmente al robot para hacerlo avanzar en linea recta hacia el marcador con el objetivo de comprobar si en el momento de la detección el robot se relocalizaba correctamente.

El resultado de esta prueba ha sido satisfactorio. Cuando se deslocaliza al robot, la silueta del láser aparece girada y desplazada con respecto al plano (figura 5.4.a). Tal y como se esperaba, en el momento en el que el robot ha entrado bajo la influencia del marcador, ha reconocido su ID y la silueta del láser ha pasado a coincidir con el plano, indicando así que el dispositivo había calculado correctamente su posición con respecto al mundo (Figura 5.4.b).

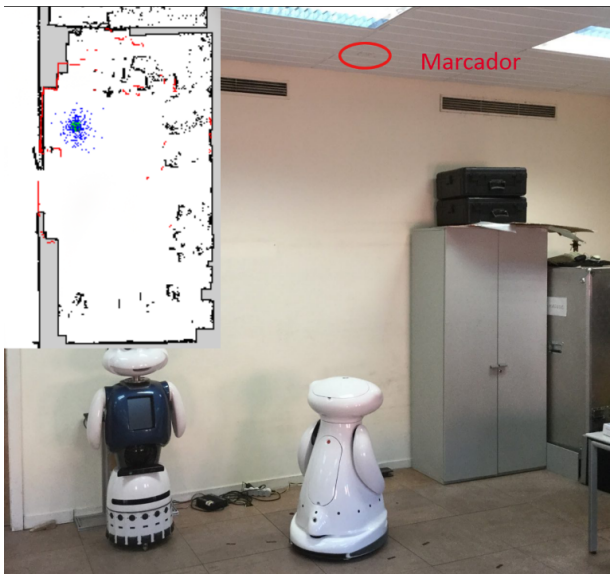
5.2.2. Prueba de relocalización con varios marcadores

La segunda prueba se ha realizado utilizando los tres marcadores del pasillo (626, 560 y 32). En este caso se ha situado al robot en un extremo del pasillo y se le ha ordenado que se desplace hasta el otro de manera autónoma utilizando la interfaz web para marcar el punto de destino. El objetivo de esta prueba es comprobar que la relocalización funciona correctamente al utilizar más de un marcador.

La detección de los marcadores, y la consiguiente relocalización del robot, funcionan correctamente. No obstante, durante el transcurso de esta prueba se han observado ciertos fallos de detección. En algunos casos cuando el robot



(a) Mbot deslocalizado.



(b) Mbot bajo la influencia del marcador.

Figura 5.4: Prueba de relocalización con un solo marcador.

pasa bajo una fuente de luz, se produce un falso positivo en la detección de marcadores. El dispositivo asocia la información que percibe con la de algún marcador y, en caso de asociar dicha información a alguno de los marcadores configurados en el fichero .yaml, puede llegar a localizarse erróneamente en el mapa.

Se ha repetido un mismo recorrido varias veces a fin de comprobar si estos errores ocurrían de forma aleatoria o se trataba de casos aislados. Gracias a ésto se ha podido comprobar que aquellos lugares donde aparece el error están asociados a fuentes de luz colocadas en el techo y además, el dispositivo siempre los detectaba bajo el mismo ID haciendo imposible corregir este comportamiento erróneo mediante métodos estadísticos.

Una posible solución planteada es la utilización de marcadores del tipo HLD1-L, ya que, al ser matrices de $4x4$ puntos, aumenta su complejidad y numero de ID's, haciendo así menos probable que sus patrones puedan confundirse con algún objeto luminoso.

Capítulo 6

Conclusiones

El objetivo principal del proyecto, integrar el dispositivo Stragazer en un robot social para llevar a cabo la detección de marcas en el entorno, se ha cumplido con éxito. Para ello se han obtenido *drivers* del dispositivo compatibles con el *framework* ROS (Robot Operating System) y se ha procedido a su estudio a fin de comprender cómo funcionan.

Una vez comprendido el funcionamiento de los *drivers*, ha sido posible instalar físicamente el dispositivo en el robot Mbot y desarrollar una aplicación que permite a éste relocalizarse al situarse bajo un marcador pasivo.

A su vez se han llevado a cabo pruebas de funcionamiento tanto del software del dispositivo como de la aplicación desarrollada para la relocalización. Los resultados de estas pruebas han demostrado pequeños errores en la precisión del dispositivo utilizado, pero dada su magnitud no han afectado al correcto funcionamiento de la relocalización.

La instalación del dispositivo y su nueva funcionalidad en el robot, han contribuido a mejorar la percepción que tiene éste de su entorno lo cual ha supuesto una ayuda considerable para el sistema de navegación a la hora de ubicar al robot en el mundo. De manera adicional, la integración del disposi-

tivo también ha hecho que surjan nuevas aplicaciones de gran utilidad para el correcto funcionamiento del robot, como es la delimitación de zonas prohibidas donde un robot móvil no debe entrar (por ejemplo, escaleras o zonas de acceso restringido)

6.1. Trabajos futuros. Zonas prohibidas

Una situación posible en la vida útil del robot es que por razones ajenas a él tenga acceso restringido a alguna zona por la que normalmente puede transitar, por ejemplo, cuando se están realizando labores de mantenimiento en algún pasillo.

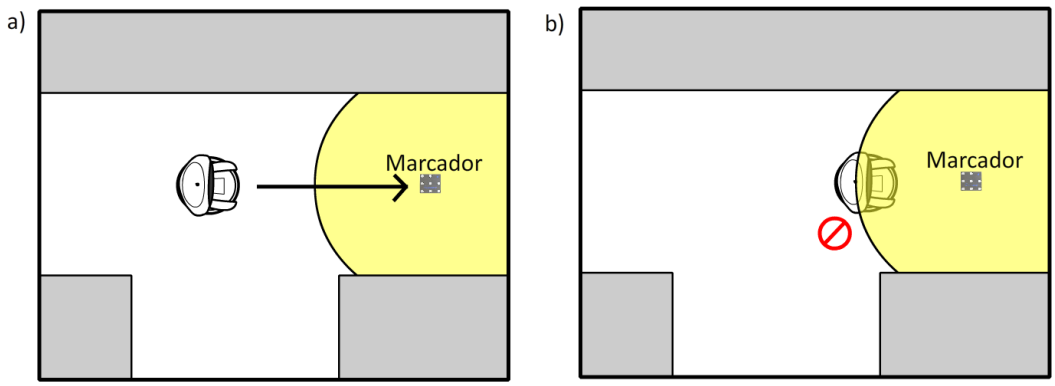


Figura 6.1: Función de Zonas prohibidas.

Para evitar que el robot acceda a éstas zonas, se ha planteado utilizar los marcadores del Stargazer para definir zonas de paso restringido. Estos marcadores se situarían en localizaciones a partir de las cuales el robot no puede transitar. De este modo, si por alguna razón (fallos en la navegación, desorientación, ordenes incorrectas, etc.) el robot se dirigiera en dirección a la zona prohibida (figura 6.1.a), se detendría en el momento en el que el Stargazer detectara el marcador que indica el comienzo de dicha zona (figura 6.1.b).

Bibliografía

[1] Ali A Abdulla, Hui Liu, Norbert Stoll, and Kerstin Thurow. Multi-floor navigation method for mobile robot transportation based on stargazer sensors in life science automation. In *2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*, pages 428–433. IEEE, 2015.

[2] Kevin Curran, Eoghan Furey, Tom Lunney, Jose Santos, Derek Woods, and Aiden McCaughey. An evaluation of indoor location determination technologies. *Journal of Location Based Services*, 5(2):61–78, 2011.

[3] Guilherme N DeSouza and Avinash C Kak. Vision for mobile robot navigation: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 24(2):237–267, 2002.

[4] David Feil-Seifer and Maja J Mataric. Defining socially assistive robotics. In *9th International Conference on Rehabilitation Robotics, 2005. ICORR 2005.*, pages 465–468. IEEE, 2005.

[5] Jodi Forlizzi and Carl DiSalvo. Service robots in the domestic environment: a study of the roomba vacuum in the home. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 258–265. ACM, 2006.

[6] Víctor González-Pacheco, Álvaro Castro-González, María Malfaz, and Miguel A Salichs. Human-robot interaction in the monarch project. Entregable DX.Y.Z. http://monarch-fp7.eu.

[7] Hagisonic. *User's guide: Localization system StarGazer for Intelligent Robots.*

[8] Junichi Ido, Yoshinao Shimizu, Yoshio Matsumoto, and Tsukasa Ogasawara. Indoor navigation for a humanoid robot using a view sequence. *The International Journal of Robotics Research*, 28(2):315–325, 2009.

[9] Barbara Klein and Glenda Cook. Emotional robotics in elder care–a comparison of findings in the uk and germany. In *International Conference on Social Robotics*, pages 108–117. Springer, 2012.

[10] Joaquín López Fernández, Christopher Watkins, Diego Pérez Losada, and Miguel Díaz-Cacho Medina. Evaluating different landmark positioning systems within the ride architecture. *Journal of Physical Agents*, 7(1):3–11, 2013.

[11] Mihai V Micea, Andrei Stancovici, and Sînziana Indreica. *Distance Measurement for Indoor Robotic Collectives*. INTECH Open Access Publisher, 2011.

[12] Yoshihiko Nakamura and Ranjan Mukherjee. Nonholonomic path planning of space robots via a bidirectional approach. *IEEE Transactions on robotics and automation*, 7(4):500–514, 1991.

[13] International Federation of Robotics. Definition of service robots. http://www.ifr.org/service-robots/. Consultado el 20 de Septiembre de 2016.

[14] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*. Kobe, Japan, 2009.

[15] Morgan Quigley, Brian Gerkey, and William D Smart. *Programming Robots with ROS: A Practical Introduction to the Robot Operating System*. " O'Reilly Media, Inc.", 2015.

[16] Robert Edward Ruskin. *Research use of instrumented drones in cloud physics and meteorology*. US Naval Research Laboratory, 1963.

[17] J Sequeira, P Lima, A Saffiotti, V Gonzalez-Pacheco, and MA Salichs. Monarch: Multi-robot cognitive systems operating in hospitals. In *ICRA 2013 workshop on many robot systems*, 2013.

[18] Henrik Stewénius. *Gröbner basis methods for minimal problems in computer vision*. Citeseer, 2005.

[19] Bob Struijk. New design philosophy in military robotics 23. 2012.

[20] S Tilch and R Mautz. Development of a new laser-based, optical indoor positioning system. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences Commission*, 1501:575–580, 2010.

[21] Guangyu Xia, Junyun Tay, Roger Dannenberg, and Manuela Veloso. Autonomous robot dancing driven by beats and emotions of music. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 205–212. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

Apéndice A

Planificación del proyecto

Este apéndice muestra un listado de todos aquellos hitos que es necesario ir cumpliendo para la realización del proyecto, así como una breve descripción de cada uno de ellos. A su vez se presenta la distribución de estas tareas en el tiempo y su tiempo de realización estimado en un diagrama de Gantt.

Las tareas que en las que se divide el proyecto son las siguientes:

A **Aprendizaje de lenguajes de programación (C++ y Python) y bases del entorno ROS.** Antes de comenzar a trabajar directamente sobre el dispositivo o el robot, es necesario familiarizarse con los distintos lenguajes de programación utilizados en el desarrollo del software. A su

Tareas	Fecha de inicio	Duración	Fecha de fin
A	15/01/2016	144	07/06/2016
B	08/06/2016	10	18/06/2016
C	11/06/2016	15	26/06/2016
D	30/06/2016	19	19/07/2016
E	01/09/2016	14	15/09/2016
F	20/07/2016	61	19/09/2016
G	20/06/2016	10	30/09/2016

Tabla A.1: Duración y fechas de las tareas

vez también es necesario familiarizarse con la plataforma ROS (Robot Operating System) que se trata del marco de trabajo sobre el que se sustenta el funcionamiento del robot.

B **Familiarización con el dispositivo Stargazer.** Para poder comenzar a trabajar con el dispositivo Stargazer, es necesario aprender cómo es el dispositivo, cómo funciona y qué es capaz de hacer. Para ello se deben consultar las especificaciones técnicas del fabricante tanto a nivel de software como de hardware.

C **Puesta en marcha del dispositivo en Ubuntu 12.04.** Un vez que se conoce bien el funcionamiento del dispositivo Stargazer, el siguiente paso es adecuarlo al sistema operativo utilizado por el robot. En este caso se trata de Ubuntu 12.04. Para ello es necesario conseguir o desarrollar *drivers* que adapten el funcionamiento del dispositivo Stargazer a ROS de modo que pueda integrarse fácilmente en el robot.

D **Pruebas de precisión y desarrollo de pequeños programas para comprobar el correcto funcionamiento del dispositivo en Ubuntu 12.04.** Para comprobar que el dispositivo y sus drivers operan correctamente, es necesario realizar una serie de pruebas de precisión en busca de posibles errores en la estimación de la posición de los marcadores que utiliza el dispositivo. En caso de encontrar errores de precisión, se procederá a su evaluación. A su vez, es conveniente realizar algún ejemplo de programa que incluya el dispositivo en su funcionamiento a fin de comprobar su rendimiento.

E **Integración del dispositivo y los drivers en el robot.** Una vez que el dispositivo se encuentra operativo en un ordenador externo, se ha de proceder a su instalación física en el robot y hay que adaptar los

programas de prueba a los parámetros utilizados por éste, comprobando así si el rendimiento en el robot es el mismo que el obtenido en las pruebas previas.

F **Integración del dispositivo y los drivers en la navegación.** Con el dispositivo ya instalado en el robot, es el momento de desarrollar distintas aplicaciones basadas en su uso que sirvan como complemento para la navegación. Para ello, es necesario estudiar el funcionamiento de ésta y adaptar el código a sus parámetros de funcionamiento.

G **Solución de problemas y evaluación de los resultados.** Esta es la última tarea del proyecto. Consiste en evaluar el correcto funcionamiento de las nuevas funcionalidades del robot y se proceder a la corrección de problemas en caso de que fuese necesario.

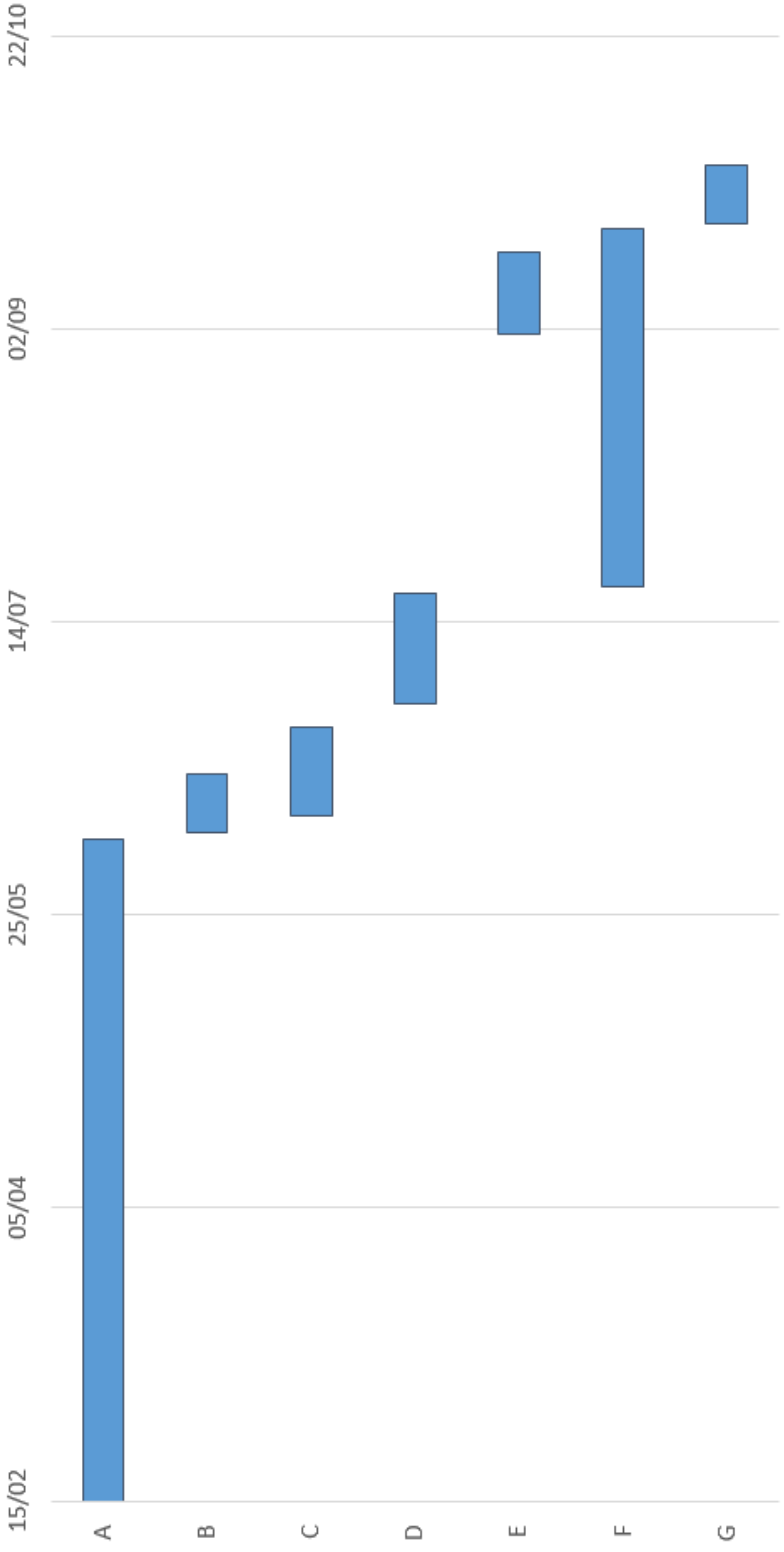


Figura A.1: Diagrama de Gantt del proyecto.

Apéndice B

Presupuesto

En este apéndice se muestra una estimación de los distintos costes que supone el desarrollo del este trabajo a distintos niveles. Para mostrar este presupuesto se ha llevado a cabo la división entre costes materiales , costes de desarrollo y costes indirectos.

- **Costes materiales**[1]**.** En este apartado se encuentran los costes referentes tanto al dispositivo y el robot, como a los equipos informáticos utilizados para el desarrollo del software.
- **Costes de desarrollo.** Aquí se ubican los costes relativos al tiempo empleado en el desarrollo, análisis y documentación del software empleado, así como el invertido en realizar las distintas pruebas sobre el dispositivo y el robot.
- **Costes indirectos.** Son los costes asociados al proyecto que no estan implícitamente relacionados con él. Aqui se incluyen la conexión a Internet, los costes de la luz, teléfono, etc. Se estima que estos gastos corresponden al 5 % de los costes netos del proyecto.

[1]El cálculo de las amortizaciones se lleva a cabo teniendo en cuenta el tiempo que se ha utilizado el activo sobre su vida util estimada.

Costes Materiales	Coste (€)	Amortización	Coste final (€)
Mbot	25000	1/30	833.33
PC portátil	870	7/36	169.17
Stargazer	872	1/8	109
Total Costes Materiales			**1111.5**

Tabla B.1: Costes Materiales.

Costes de desarrollo	Coste (€/hora)	Horas	Coste final (€)
Análisis	20	130	2600
Programación	20	210	4200
Pruebas	20	30	600
Documentación	20	50	1000
Total Costes Materiales			**8400**

Tabla B.2: Costes de Desarrollo.

Presupuesto Final		Coste final (€)
Costes Materiales		1111.5
Costes de Desarrollo		8400
Total neto		**9511.5**
Costes Indirectos	(+5 % del total neto)	475.58
Total del Proyecto		**9987.08**

Tabla B.3: Presupuesto final.

Apéndice C

Ficha técnica del Stargazer

StarGazer™ User's Guide

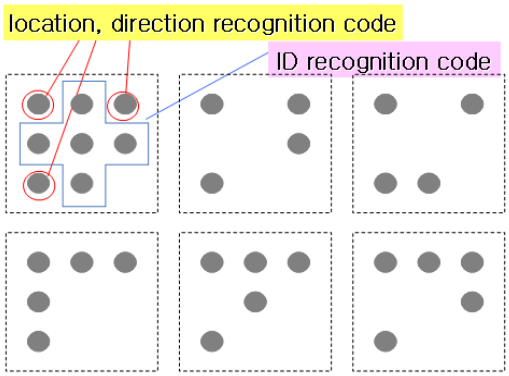
Table of Contents

1. Product Overview

StarGazer™ is a unique sensor system for Indoor localization of intelligent mobile robots. It analyzes infrared ray image which is reflected from a passive landmark with an independent ID. The output of position and heading angle of a robot is given with very precise resolution and high speed. It is seldom affected by surroundings such as an infrared ray, a fluorescent light and sunshine.

An illustration showing how StarGazer™ works.

StarGazer™



Passive Landmarks

2. Parts List

A. StarGazer Module (DSP Module, IRED Module, Support, Lens Hood) - 1set

B. 3pin Connector with cables - 1ea, 7pin Connector with cables - 1ea

C. User's Manual (Download from website www.hagisonic.com.)

D. Sample Program with Visual C++ 6.0(Download from website www.hagisonic.com.)

3. Features and Specification

A. Landmarks

- With IDs i.e. codes given by the combination of circles to reflect infrared rays
- No battery or adapter for Landmark is required.
- Easy extension of application range
- Landmark Types by use

1) HLD1: for *1.1 m ≤ Height ≤ 2.9 m*

HLD1-S: 3 X 3 combination, total 31ea, for normal use

HLD1-L: 4 X 4 combination, total 4,095ea, for larger spaces

2) HLD2: for *2.9 m ≤ Height ≤ 4.5 m*

HLD2-S: 3 X 3 combination, total 31ea, for normal use

HLD2-L: 4 X 4 combination, total 4,095ea, for larger spaces

3) HLD3: for *4.5 m ≤ Height ≤ 6.5 m*

HLD3-S: 3 X 3 combination, total 31ea, for normal use

HLD3-L: 4 X 4 combination, total 4,095ea, for larger spaces

※ 'Height' means the distance between a StarGazer between a landmark which is attached on ceiling.

※ Please refer to [12. Appendix D. for the detailed information about landmark]

B. Specification of StarGazer™

Hardware Interface	UART(TTL 3.3V), 14,400bps ~ 115,200bps
Size	50×50×28mm
Communication Protocol	User protocol based on ASCII code
Measurement Time	10 times/sec
Localization Range (per a Landmark)	2.5~3m in diameter (for ceiling height 2.4m)
Repetitive Precision	2 cm
Heading Angle Resolution	1.0degree
Landmark Types (Classification for height range)	*HLD1: 1.1 ≤ height ≤ 2.9 m* *HLD2: 2.9 ≤ height ≤ 4.5 m* *HLD3: 4.5 ≤ height ≤ 6.0 m*
Landmark Types (Classification for total ID numbers)	HLDnS: 31 ea (for a normal space) HLDnL: 4,095 ea (for a larger space) (n=1,2,3; see the classification for height range)
Power Consumption	5 V: 300 mA, 12 V: 70 mA

C. Features and Performance

- It analyzes the image of the infrared ray which is reflected from a passive landmark with a unique ID.
- It is composed of an IR Projector part and an image processing unit.
- High resolution and high speed localization of position and heading angle.
- Landmark is used by being attached on ceiling.
- No need for any synchronization or communication between a robot and a landmark.
- The area that StarGazer covers is extended by only adding landmarks to ceiling.
- Each room can be distinguished easily each other by using landmarks with different IDs.
- Automatic measurement and calibration of distance between landmarks and ceiling height.
- No battery or power supply for landmark is needed.
- A little extra cost consumes when landmarks are attached additionally.
- Nearly not affected in environment such as lamp and sunlight
- It works excellent localization function at night as well as in the day.
- World's best in resolution, convenience and cost-efficiency

4. Connector Configuration

① Connector configuration for DSP Module

Cable Line Color	White	Black	White	Black	White	Red	Red
Function	Reserved	GND	SDIN	GND	SDOUT	VCC(5V)	VCC(5V)

② Connector configuration for IRED Module

Cable Color	Black	White	Yellow
Function	GND	Reserved	VCC(12V)

5. UART Configuration

The StarGazer supports UART communication as shown in Table 1.

Table. 1. UART configuration

I/O Level	TTL 3.3V Output, 3.3V~5V Input
Baudrate	14400 bps ~ 115200 bps
Data Bit	8bit
Stop Bit	1bit
Paraty Bit	None
Flow Control	None

6. Communication Protocol

StarGazer calculates coordinates and heading angle using parameters in flash memory. The protocols, shown in Table 2 and Table 3, can be used to read or update the parameters.

A. Communication Protocol, Parameters, Commands

Table. 2. Communication Protocols

Read	STX	@	Parameter/Command	ETX	
Write	STX	#	Parameter/Command	| Data	ETX
Return Value	STX	$	Parameter/Command	| Data	ETX
ACK	STX	!	Parameter/Command	| Data	ETX
Message	STX	*	Message	[|Data]	ETX

※ Notice: STX: '~', ETX: '`'

Table. 3. Parameters and commands

Parameters and Command	Version	Firmware Version
	IDNum	Number of ID(1-31, 1-4095)
	RefID	Reference ID(2-626, 2-28662)
	HeightFix	Mark Height Fix(Yes/No)
	MarkHeight	Height of Landmark(mm)
	MarkType	Mark Type(HLD1S/HLD1L/HLD2S/HLD2L/HLD3S/HLD3L)
	MarkMode	Landmark Mode(Alone/Map)
	BaudRate	UART Baudrate(14400~115200bps, default:115200bps)
	HeightCalc	Calculate Height of Landmark
	MapMode	Map Building Mode(Start/Stop)
	CalcStart	Calculation Start
	CalcStop	Calculation Stop
	SetEnd	Parameter Setting End
	Reset	Reset All Parameters

1) Basic Command and Protocol

~ : to mean the start of command sentence; STX(start of text) character.

` : to mean the end of command sentence; ETX(end of text) character.

@ : to mean command to read a following parameter; READ command

! : to follow automatically when READ or WRITE command completely executed; ACK(acknowledge) character. Response symbol sent from a StarGazer.

$: Response symbol to mean that data follow after the following parameter as response of READ command.

* : Symbol to indicate the message from StarGazer

| : Symbol to distinguish a command from data or to distinguish Parameter from data

2) Parameters for data

- Parameters: Version, IDNum, RefID, MarkHeight, BaudRate

Version: Version of Firmware.

IDNum: A total number of landmarks to be assigned under Map Mode. Default value: 4

RefID: The number of reference ID under Map Mode; Default value: 2

MarkHeight: Distance from a StarGazer to a landmark; used when wanting to input manually the height; Default value: 2400 mm

BaudRate: Communication Speed for UART(14400bps~115200bps); Default value: 115200.

3) Parameters for setting up modes

- Parameters: HeightFix, MarkType, MarkMode

HeightFix : Select the option to use fixed ceiling height or automatic measurement (Yes or No).

- Yes : This option will calculate the coordinates by using the manually inputted ceiling height.(fixed ceiling height).
- No : This option will calculate the coordinates automatically all the time.(When StarGazer is used in different ceiling height). This option will reduce the precision slightly. This option can not be used with different landmark types. For example, HLD1-S or HLD1-L has 1.1m~2.9m range and this option only allows the different ceiling height within this range. When using HLD1-S(L) and HLD2-S(L), for instance, this option can not be used. This rule applies to all of our landmark models.

MarkType: To set up landmark type by use(HLD1S, HLD1L, HLD2S, HLD2L, HLD3S, HLD3L).

- HLD1 : 1.1 ≤ height ≤ 2.9 m
- HLD2 : 2.9 ≤ height ≤ 4.5 m
- HLD3: 4.5 ≤ height ≤ 6.0 m

HLDnS means landmark up to 31 IDs and HLDnL means landmark up to 4095 IDs. Default: HLD1S.

MarkMode: To determine whether landmarks are used independently under Alone Mode or not (dependently under Map Mode). There are Alone and Map. Default; Alone (if Map mode, then 'Map'.)

4) Execution Commands

- Commands: HeightCalc, MapMode, CalcStart, CalcStop, SetEnd, Reset

HeightCalc: Command to calculate automatically height between a StarGazer and a landmark. It is enough to execute only once when installing.

MapMode: To determine whether map building mode is executed or not. There are Start and Stop. If action under Map Mode is required, you should set the parameter 'Start' and start Map Building. Default; Stop

CalcStart: Command to start calculation. After executing the command, the output of data including position and angle is obtained continuously.

CalcStop: Command to stop calculating.

SetEnd: Command to mean the completion of a serious of command sentences. Values for a serious of parameters given in preceding several commend sentences are operated only after 'SetEnd' command comes into practice.

Reset: Reset all parameters

Default reset values are as follows:

IDNum = 4

RefID = 2

MarkHeight = 2400

MarkType = HLD1S

MarkMode = Alone

BaudRate = 115200

※ Note: Execution command 'HeightCalc', 'CalcStart', 'CalcStop', 'Reset' are operated without 'SetEnd'.

5) Message

- Message from StarGazer during operation
- Commands: DeadZone, MAPID

DeadZone: The message indicates that there is no landmark detected.

Example: ~*DeadZone`

MAPID: a newly-mapped ID during a map-building process.

Example: ~*MAPID|id`

B. How to write data to parameters and the procedure

① Send a command to stop calculation. Ex. ~#CalcStop`

② Send a command sentence for the change of a parameter.

Ex. ~#MarkHeight|2200`

③ Wait a response message. In the response '#' is changed to '!'.

Ex. ~!MarkHeight|2200`

④ Send another command sentence for the change of another parameter and wait a response.

Send other sentences in the same way.

⑤ Send the completion command 'SetEnd' after sending whole sentences for parameters. Ex. ~#SetEnd`

⑥ StarGazer responds with the message, ~!ParameterUpdate`, after receiving 'SetEnd' and updating the values and strings for parameters to flash memories.

⑦ Finally, send a command to start calculating. And then, calculated data output is obtained. Ex. ~#CalcStart`

C. How to read data in parameters and the procedure

① Send a command to stop calculation. Ex. ~#CalcStop`

② Send a command sentence to read a parameter. Ex. Read the height of a landmark: ~@MarkHeight`

③ Wait a response message. In the response '@' is changed to '!'. Ex. ~!MarkHeight`

④ Data are immediately received after the response message. If the data is value, the character '$' accompanies the message. Ex. If height is 2200 mm, response is ~$MarkHeight|2200`

⑤ Send the completion command 'SetEnd' after sending whole sentences. Ex. ~#SetEnd`

⑥ Send a command to start calculating. Ex. ~#CalcStart`

D. Examples of Parameter Setting and Map-Building

1) Update a number of IDs to 8, and reference ID to 32

① ~#CalcStop` response message => ~!CalcStop`

② ~#IDNum|8` response message => ~!IDNum|8`

③ ~#RefID|32` response message => ~!RefID|32`

④ ~#SetEnd` response message => ~!SetEnd`
response message => ~!ParameterUpdate`

⑤ ~#CalcStart` response message => ~!CalcStart`

2) Calculate automatically the height of a landmark

① ~#CalcStop` response message => ~!CalcStop`

② ~#HeightCalc` response message => ~!HeightCalc`

③ The data received during calculating
~^Z2|+150.12|-33.12|+12.00|240.00` [Refer to 7. Receiving Data Format]

④ The reponse received after the completion of calculation; ~!ParameterUpdate`

⑤ ~#CalcStart` response message => ~!CalcStart`

3) Start the Map-Building Process and the message

① ~#CalcStop` response message => ~!CalcStop`

② ~#MarkMode|Map` response message => ~!MarkMode|Map`

③ ~#MapMode|Start` response message => ~!MapMode|Start`

④ ~#CalcStart` response message => ~!CalcStart`

⑤ The data received during Map-Building process

~^F2|-165.74|-33.12|+12.00|240.00` [Refer to 7. Receiving Data Format]

⑥ Move StarGazer around landmarks until mapping is completed.

⑦ Receive the Mapped ID Data

~*MAPID|32`

⑧ While all landmark mapping completed the StarGazer move.

⑨ After the completion of mapping the response message is accompanied by

~!ParameterUpdate`

⑩ Then, receiving data is as follows:

~^2I+58.48|-33.12|+12.00|240.00` [Refer to 7. Receiving Data Format]

4) Update the baudrate for UART to 38400

① ~#CalcStop` response message => ~!CalcStop`

② ~#BaudRate|38400` response message => ~!BaudRate|38400`

③ ~#SetEnd` response message => ~!SetEnd`

(response message => ~!ParameterUpdate`)

※ Once a baudrate is changed, the message [ParameterUpdate] cannot be seen. But to get communication back on track, the baudrate of a windows application program or robot CPU should be changed.

④ ~#CalcStop` response message => ~!CalcStop`(for the communication check)

⑤ ~#CalcStart` response message => ~!CalcStart`

E. Notice

① Though it can be possible to send or receive data without the command, 'stop calculation', in that case sometimes the command cannot operate because StarGazer is sending data successively. Therefore, it is strongly recommended to use the command 'stop calculation', prior to other command sentences.

② Sometimes, the command 'stop calculation' can be executed because of the same reason that a command is not executed though a command has been sent, the command should be sent repeatedly.

③ In order to communicate stably with StarGazer, minimum 30~50ms delay is required for every byte.

④ The program should be written to confirm the response message for each command.

⑤ When StarGazer updates a memory, several seconds, typically two or three seconds, is required. So, after a command such as [SetEnd] or [HeightCalc] or after the completion of Map-Building, StarGazer cannot operate. Note that [~!ParameterUpdate`] message is shown after the completion of a memory update.

⑥ StarGazer operates only over 1.1m height.

⑦ Map-Building should be programmed by the procedure to be shown in [6. D. 3) Start the Map-Building Process and the message]

7. Format of Received Data

The format of data received from StarGazer for the command ~#CalcStart` are as follows:

~	^	F I Z	Iiii|±aaaa.aa|±xxxx.xx|±yyyy.yy|zzzz.zz	`

^	Means the result data
F	Indicates the Map-Building Mode
I	Indicates the Map Mode
Z	Indicates the Height Calculation Mode
iiii	The number of an ID
±aaaa.aa	Value of Angle (degrees; -180˚~+180˚)
±xxxx.xx	Position on X axis (cm)
±yyyy.yy	Position on Y axis (cm)
zzzz.zz	Position on Z axis; Height of landmark (cm)

(Angle, X, and Y value are truncation to two decimal point)

Ex. ~^I+150.23|-33.12|+12.00|64`

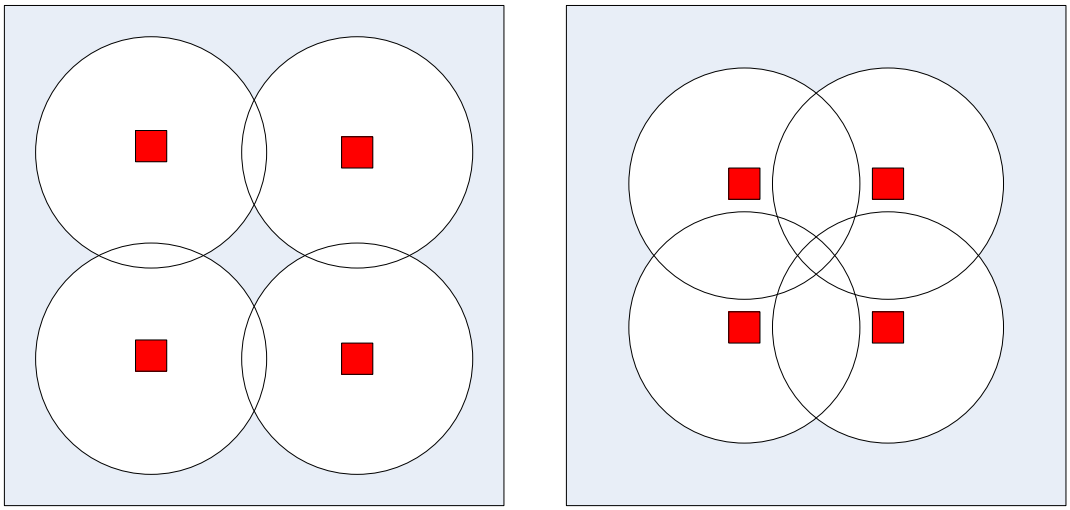
(~^I : Map Mode, +150.23:Angle, -33.12:X, +12,00:Y, 64 : IDNumber)

※ Map-building mode: a mode in the process of making a map by correlating several landmarks under a single coordinate system. In the process StarGazer is moved around under each landmark and correlation between landmarks is calculated.

※ Map mode: the mode to calculate the position and angle of StarGazer and send the data using the Map after the completion of Map-Building.

8. Guidance for Landmark Placement

The landmarks should be placed at 2 m intervals for the height of about 2.5 m in order that any dead zone may not occur.



(a) (b)

The placement of landmarks: (a) with dead zone and (b) without dead zone.

9. Alone Mode and Map Mode

In 'Alone Mode' a system operates under each independent coordinate system corresponding to each landmark.

In 'Map Mode' a system operates under one coordinate system defined by regarding the placement of a reference ID as an origin after map-building process.

10. Map-Building Method

1) Map-building: to make a map under single coordinate system. The placement of a reference ID becomes an origin.
2) Set 'MarkMode' to 'Map' and 'MapMode' to 'start'.
3) If StarGazer detects a landmark, a position data accompanied by '~^F" is responded.
4) As the next step, move toward other nearest landmark and stop for about two seconds near halfway between two landmarks for the time to calculate the relation.
5) And then, move toward other landmark and stop near midway between another landmarks.
6) Proceed by the same way until whole landmarks are detected by StarGazer.
7) If the last landmark is found, the operating is stopped for a short time for data updating to flash memory.

8) Then, a response message, data message accompanied by '~I^' is given and map-building process ends.

9) Mode is automatically changed to 'Map Mode'.

※ For example and some details, refer to [6. D. 3) Start the Map-Building Process and the message]

11. Inquiries for Technical Support and Customer Service

HAGISONIC Co., LTD
TEL: +82-42-936-7740
FAX: +82-42-936-7742
Address: 535 Yongsan-dong, Yuseong-gu, Daejeon 305-500, Korea (South)
Website: www.hagisonic.com
Email : hagisonic@hagisonic.com

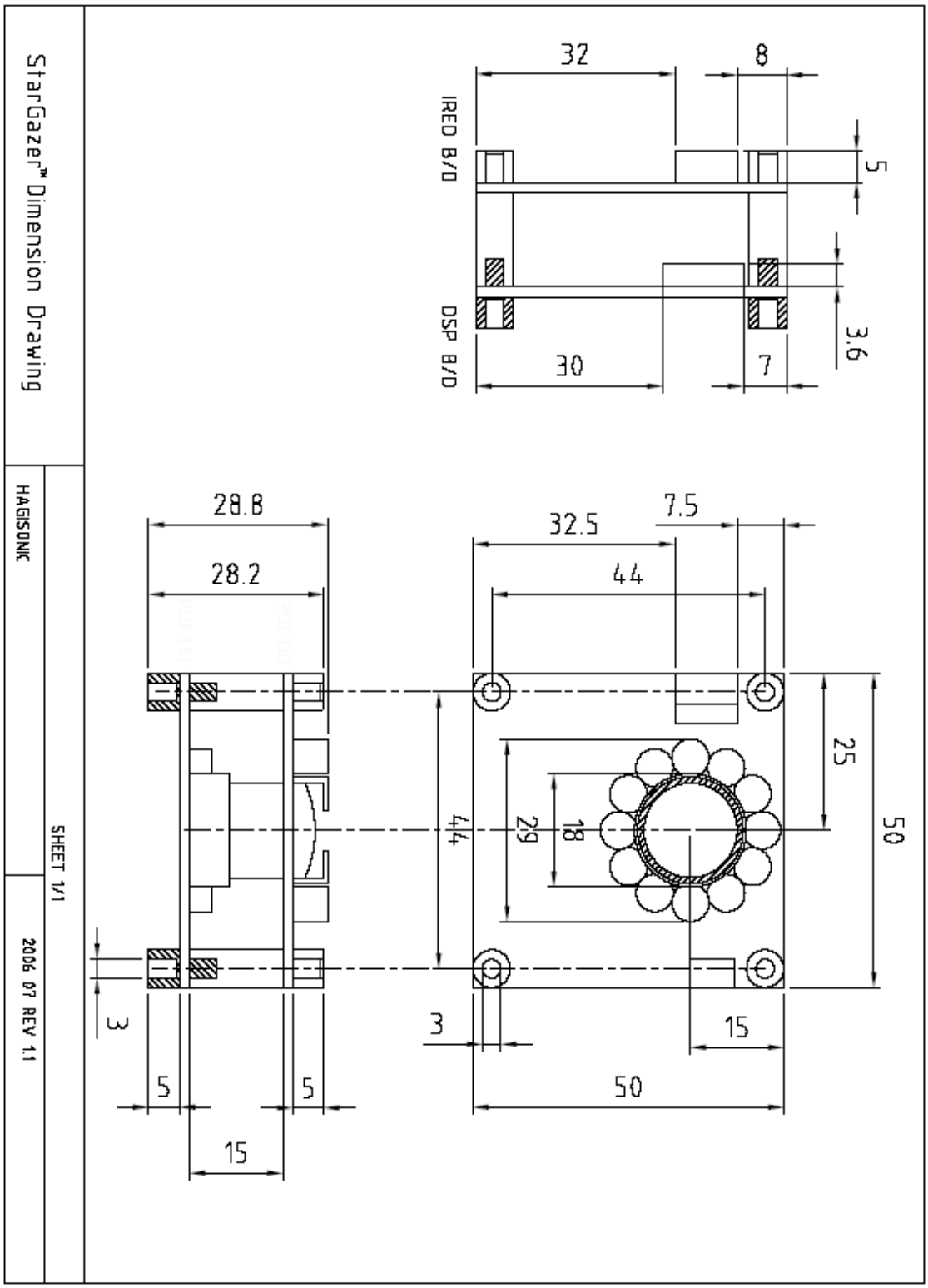
12. Appendix

A. Product Dimension

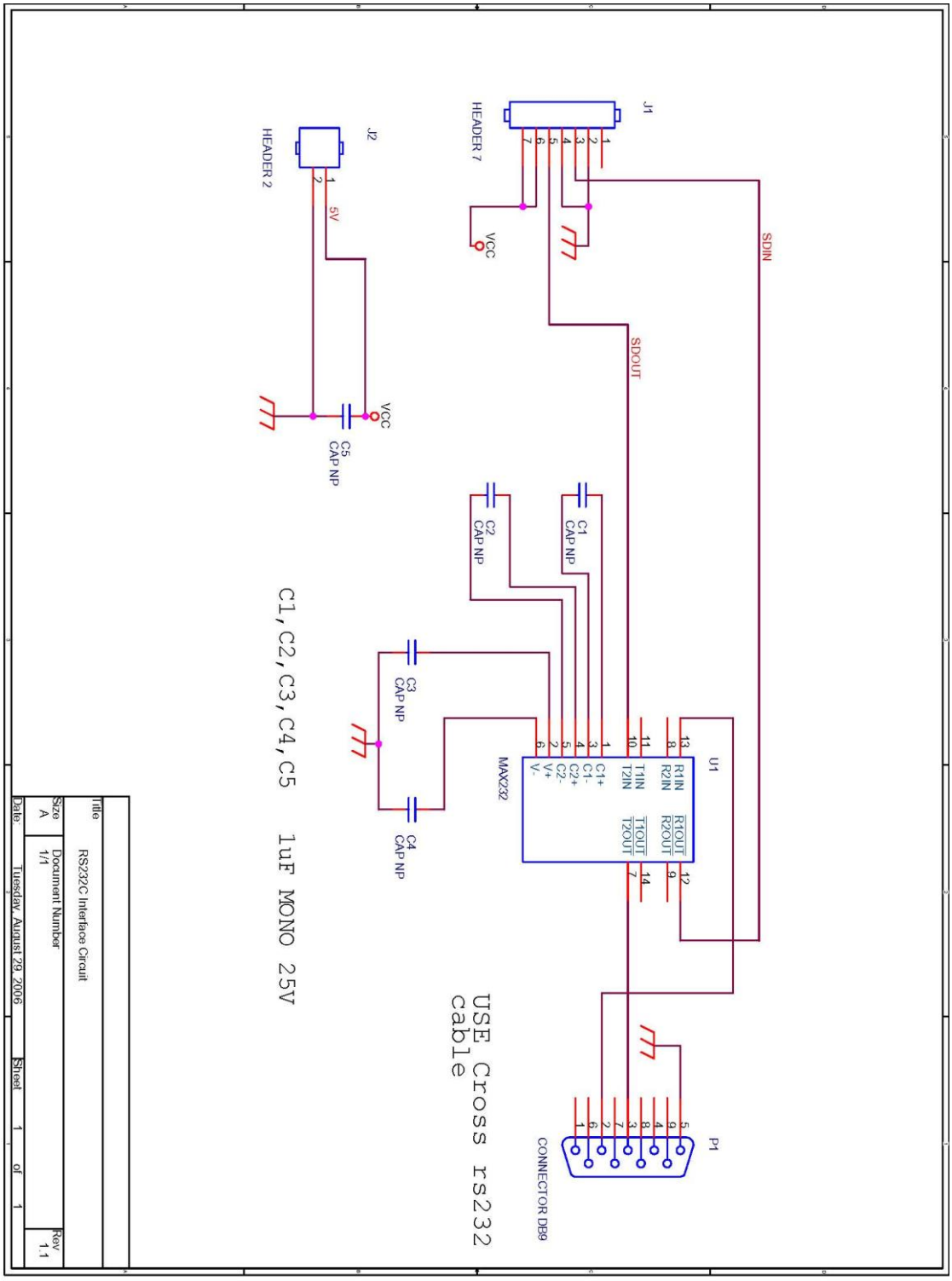
B. RS232 Interface Circuit (for the communication between StarGazer and PC)

C. The types of landmarks and how to generate the number of ID.

A. Dimension of StarGazer™



B. RS232 Interface Circuit
(for the communication between StarGazer™ and PC)



C. Types of landmarks and how to generate the number of ID

(1) HLD1 landmarks are composed of the 3 X 3 combination of small circles. The total number is 31. The landmarks are used for general application such as at home.

(2) HLD2 landmarks are composed of the 4 X 4 combination of small circles. The total number is 4095. The landmarks are used for the application to very large area with several offices.

(3) Each line corresponds to an identified hexadecimal value.

(4) Fig.16-C-3 shows HL1 landmarks and corresponding decimal values.

※ '0x'in figures is the notation to mean hexadecimal.

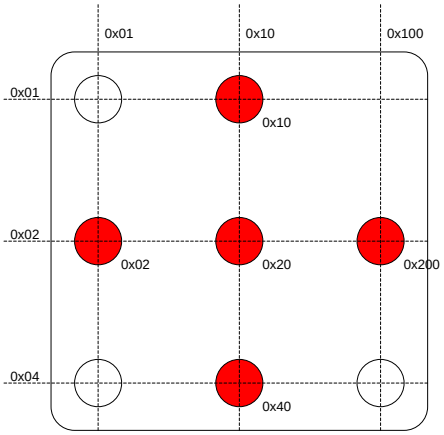


Fig.16-C-1. HL1 landmark with hexadecimal values corresponding to each line.

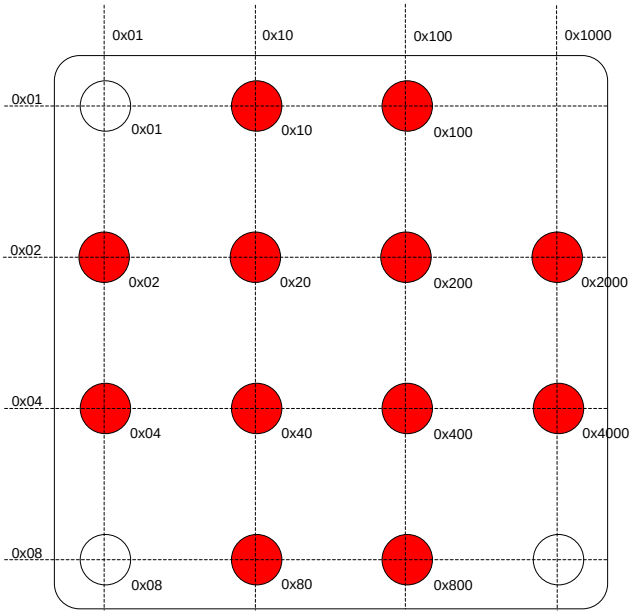
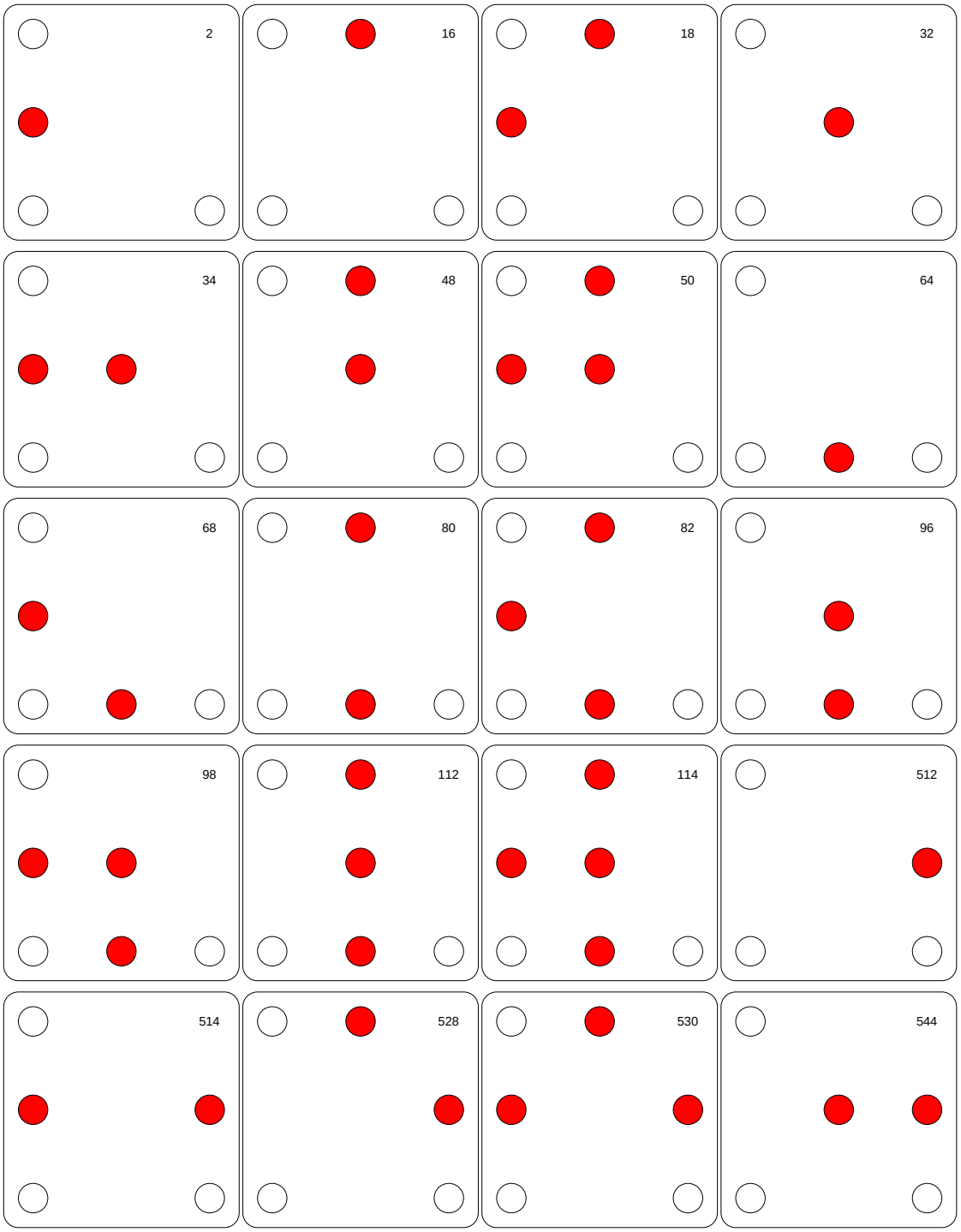


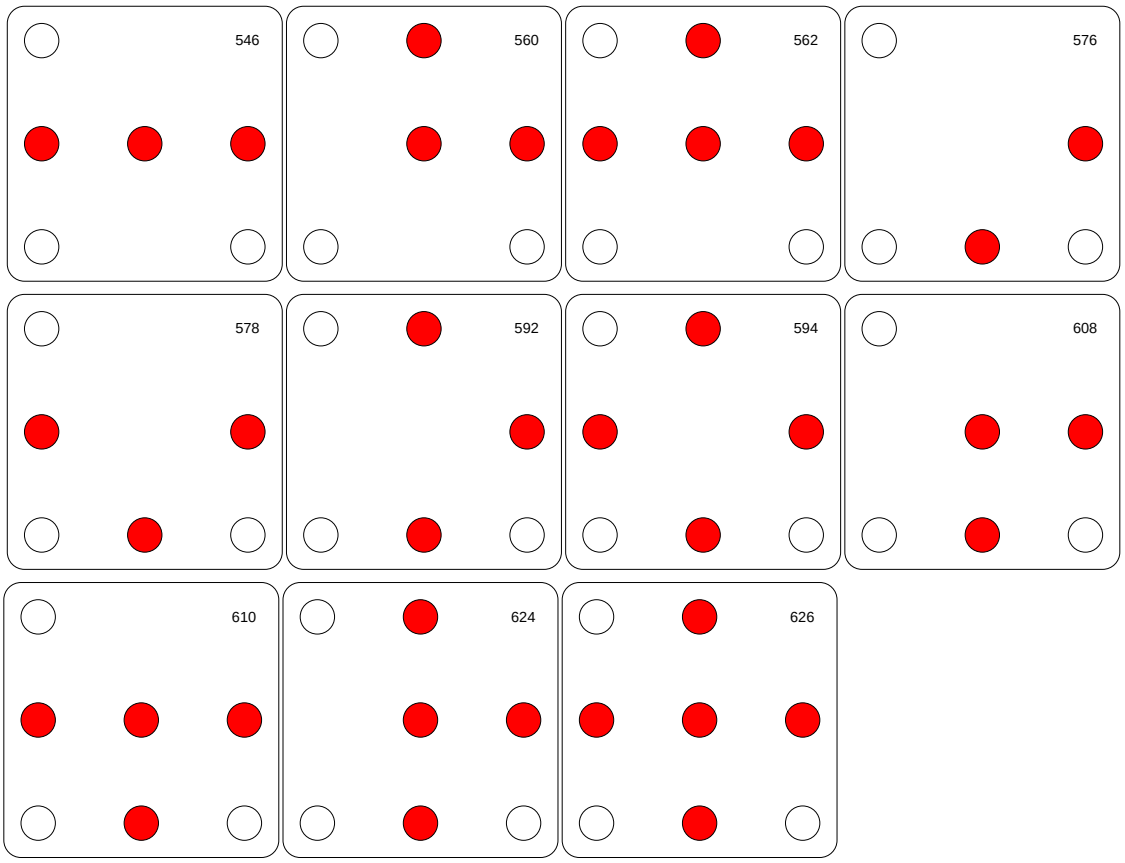
Fig.16-C-2. HL2 landmark with hexadecimal values corresponding to each line.

번호	HEX	DEC
1	0x002	2
2	0x010	16
3	0x012	18
4	0x020	32
5	0x022	34
6	0x030	48
7	0x032	50
8	0x040	64
9	0x042	68
10	0x050	80
11	0x052	82
12	0x060	96
13	0x062	98
14	0x070	112
15	0x072	114
16	0x200	512
17	0x202	514
18	0x210	528
19	0x212	530
20	0x220	544
21	0x222	546
22	0x230	560
23	0x232	562
24	0x240	576
25	0x242	578
26	0x250	592
27	0x252	594
28	0x260	608
29	0x262	610
30	0x270	624
31	0x272	626

Fig.16-C-3. Showing HL1 landmarks and ID numbers.

Fig.16-C-4. Type of the HL1 landmark(decimal number inside the landmark means a ID number)





D. StarGazer RS1.0

(for the communication between PC, Main Process and StarGazer)

(a) StarGazer™ RS 1.0 Kit

1. The user-friendly solution which easily outputs and controls data from StarGazer™, localization sensor, through standard serial wire / wireless communication in PCs, various systems, and robots
2. The optimum localization system which is registered to Microsoft Robotics Studio.
3. Application : Research in laboratory and development of the prototype of robots.

I/O Level	TTL 3.3V Output, 3.3V~5V Input
Size	62x56x50.8mm
Power	DC 12V
Baudrate	115200 bps
Data Bit	8 bit
Stop Bit	1 bit
Parity Bit	None
Flow Control	None

※ Output : ~^I+150.23|-33.12|+12.00|64`