Guía UWB con MDEK1001

Diciembre, 2021

# Resumen

El objetivo de este documente es describir el paquete de ROS dwm1001\_drivers. Este paquete ha sido desarrollado con la finalidad de poder realizar experimentos de autocalibración y localización con balizas UWB. Dichas balizas forman parte del kit de desarrollo MDEK1001 de Decawave https://www.decawave.com/product/mdek1001-deployment-kit

# 1. Experimento de autocalibración

El objetivo de este experimento es poder determinar la posición de los *anchors* en base a lecturas de rangos entre *anchors*. En las siguientes subsecciones se describe el procedimiento que se deberá seguir para realizar el experimento.

## 1.1. Situar los anchors distribuidos en el espacio

Será necesario definir un **sistema de coordenadas** (habitualmente /world) sobre el cual se definirán las posiciones de los *anchors*, además de **anotar las posiciones** **de los *anchors*** respecto de este sistema de coordenadas. Dichas posiciones serán consideradas como el ***ground truth***a la hora de evaluar los algoritmos de autocalibración. Para anotar estas posiciones y definir el resto de los parámetros necesarios (para este experimento y para el experimento de localización) será necesario modificar/crear un archivo /params/nodes\_cfg/<nodes\_cfg\_label>.yaml (se puede tomar el fichero default.yaml como plantilla).

## 1.2. Tomar lecturas de rangos desde cada anchor al resto de anchors

Para tal fin, dentro del paquete, será necesario ejecutar un script de Python que convierte al *anchor* en un ***tag***, toma **lecturas** de rangos en cada una de las redes definidas en el fichero de configuración /params/nodes\_cfg/<nodes\_cfg\_label>.yaml, **guarda** las lecturas en un fichero de texto y vuelve a configurar la baliza como un ***anchor***. Dicho script se encuentra en la carpeta /scripts y los ficheros de texto se guardarán en la carpeta desde la cual se ejecute el siguiente comando:

python scripts/autocalibration\_sample\_uart.py <n\_samples> <nodes\_cfg\_label> <initiator\_enabler> <dwm\_module>

Donde n\_samples es el número de lecturas de rangos entre el *anchor* con id dwm\_module y el resto de *anchors*; nodes\_cfg\_label indica la configuración de las redes de los anchors; e initiator\_enabler indica si dwm\_module se configurará como initiator una vez finalizadas las lecturas.

## 1.3. Determinar la posición de los anchors en base a las lecturas tomadas (EN DESARROLLO)

Los /scripts de python para poder realizar la autocalibración de posición de los anchors se encuentran en el submódulo /dwm\_ble\_cfg debido a los requerimientos de librerías como pandas. Por tanto, para poder ejecutar estos scripts será necesario instalar estas librerías en un entorno virtual mediante el fichero /dwm\_ble\_cfg/requirements.txt o bien instalarlas en el entorno global de python. Para resolver la autocalibración se deberá ejecutar el siguiente /script que implementa un procedimiento multi-stage basado en una reestimación iterativa y en una etapa final de optimización:

python scripts/autocalibration\_solver.py <nodes\_cfg\_label> <n\_samples>

Donde nodes\_cfg\_label se corresponde con el fichero de configuración utilizado durante las lecturas de rangos entre anchors y n\_samples es el número de lecturas realizadas.

**Futuras ampliaciones**: será necesario guardar las posiciones de lo *anchors* resultantes de la autocalibración en el fichero de configuración <nodes\_cfg\_label>.yaml para posteriormente utilizarlas en los experimentos de localización.

# 2. Experimento de localización

El objetivo del experimento de localización será determinar la posición de los *tags* acoplados al robot que, en las siguientes secciones, se asume que es un kobuki. En las siguientes subsecciones se describe el procedimiento que se deberá seguir para realizar el experimento.

## 2.1. Situar los anchors en el espacio

Si no se ha realizado previamente el experimento de autocalibración será necesario situar los anchors en el espacio, definir un sistema de coordenadas, anotar las posiciones de los anchors respecto de este sistema de coordenadas y modificar/crear un fichero de configuración en /params/nodes\_cfg/<nodes\_cfg\_label>.yaml (se puede tomar el fichero default.yaml como plantilla).

## 2.2. Poner en marcha los drivers del kobuki y su sensor

Para poner en marcha los drivers del kobuki y su sensor será necesario seguir los apartados **Conexión con la NUC montada sobre un kobuki** y **Puesta en marcha de un kobuk**i del documento **Guía kobuki\_multi\_robot** que puede encontrarse en https://github.com/Taucrates/kobuki\_multi\_robot/blob/master/documentation/guia\_kobuki\_multi\_robot.docx

## 2.3. Crear un mapa del entorno

Para este paso será necesario consultar el apartado de **Crear un mapa del entorno** del documento **Guía kobuki\_multi\_robot**. El mapa deberá guardarse en la carpeta /dwm1001\_drivers/maps.

## 2.4. Poner en marcha el sistema de localización con UWB

Para poner en marcha el sistema de localización con UWB será necesario poner en marcha los drivers ROS para módulos DWM1001 configurados como *tags*. Para ello, será necesario ejecutar el siguiente comando para cada uno de los *tags* acoplados al robot:

roslaunch dwm1001\_drivers dwm1001.launch <nodes\_cfg\_label> <tag\_id>

Nota: El funcionamiento del driver consiste en leer la información proporcionada por los tags (rangos entre *tag* y los *anchors* que se encuentren en su misma red) transmitida por puerto serie, elaborar mensajes de ROS con esta información y publicarla en los tópicos que más adelante se utilizarán en un nodo que implementa el algoritmo de localización.

Si en el archivo /params/nodes\_cfg/<nodes\_cfg\_label>.yaml se han configurado correctamente las posiciones de los anchors es posible poner en marcha el nodo de localización que leerá la información publicada por los nodos que implementan los drivers ROS para los módulos DWM1001.

roslaunch dwm1001\_drivers localization.launch <nodes\_cfg\_label> <using\_ekf>

Donde using\_ekf indica si el método de localización utilizado es un *Extendend Kalman Filter* (EKF) o un método basado en *Least Squares* en caso contrario.

## 2.5. Preparaciones adicionales

Antes de poder hacer un recorrido con el robot y poder estimar su posición serán necesarios algunas preparaciones adicionales expuestas a continuación.

1. **Definir la TF entre world y map**

Asumiendo que el sistema de coordenadas sobre el cual se han situado los *anchors* es /world y el sistema de coordenadas sobre el cual se encuentra el kobuki es /map, será necesario definir la TF entre ambos sistemas de coordenadas. Para ello será necesario definir la publicación de una TF dentro del archivo /dwm1001\_drivers/launch/nodes\_viz.launch.

1. **Lanzar la visualización/monitorización del recorrido**

Para facilitar la visualización del recorrido tanto un archivo *launch* como un fichero de configuración .viz están disponibles en el paquete.

roslaunch dwm1001\_drivers nodes\_viz.launch <map\_name>

Donde map\_name indica un mapa dentro de la carpeta /dwm1001\_drivers/maps.

1. **Lanzar los nodos para tener una referencia de la posición del robot (*ground truth)***

Es posible obtener una referencia de la posición verdadera del robot mediante un nodo que implementa un AMCL o bien mediante un cliente VRPN que publica la posición del robot obtenida con *Optitrack*.

Para poner en marcha el **nodo ACML** será necesario acceder al paquete kobuki\_multi\_robot:

roscd kobuki\_multi\_robot

Modificar la posición inicial del kobuki respecto del sistema de coordenadas /map:

nano /param/missions/<mission\_label>/<robot\_name>.yaml

Y, por último, poner en marcha el nodo ACML:

roslaunch kobuki\_multi\_robot amcl.launch <robot\_name> <mission\_label>

Si, en lugar de obtener la posición de referencia del kobuki con AMCL, se selecciona el sistema ***Optitrack*** para tal fin será necesario seguir los siguientes pasos:

1. Encender las cámaras (enchufe *switch*)
2. Encender el PC (Windows 8, contraseña: srvpass)
3. Abrir motive, seleccionar proyecto de la carpeta Documentos/kobuki
4. Abrir el *layout* con nombre *capture*
5. Por último, poner en marcha el cliente VRPN (tanto el PC como la máquina sobre la que se ejecute el cliente deben de estar en la misma red)

roslaunch dwm1001\_drivers vrpn\_client.launch

Es importante corregir la TF entre /optitrack y /world definida en este archivo *launch* si es necesario.

## 2.6. Recorrido con el kobuki

Para poder hacer un recorrido con el kobuki será necesario poner en marcha dos nodos mediante dos archivos launch ubicados dentro del paquete kobuki\_multi\_robot.

roslaunch kobuki\_multi\_robot velocity\_smoother.launch <robot\_name> (en NUC kobuki)

roslaunch kobuki\_multi\_robot keyop.launch <robot\_name> (en *base station*)

## 2.7. Grabación del experimento

Para poder grabar el recorrido realizado con el kobuki mientras se obtienen las posiciones de referencia (AMCL u *Optitrack*) y estimadas (EKF o *Least Squares*) será necesario grabar archivos *.bag*. Para ello, accederemos a aquella carpeta donde queramos guardar dichos archivos:

cd <carpeta>

Y, posteriormente, grabaremos todos los tópicos publicados

rosbag record -a -O <file\_name>

En lugar de grabar todos los tópicos (-a) es posible indicar únicamente los tópicos que queremos grabar. Esta y otras funcionalidades de rosbag pueden encontrarse en http://wiki.ros.org/rosbag/Commandline

# 3. Procesamiento de datos

Para poder analizar las posiciones estimadas y compararlas con las posiciones de referencia será necesario crear ficheros de texto con las posiciones de referencia y las lecturas de rangos entre *tags* y *anchors*. Para ello será necesario:

1. Modificar el archivo que describe la distribución de los nodos en /dwm1001\_drivers/params/nodes\_cfg/<nodes\_cfg\_label>.yaml, si no se ha hecho anteriormente durante el experimento.

2. Filtrar el bag grabado si es necesario (ver http://wiki.ros.org/rosbag/Commandline)

rosbag filter grabacion\_original.bag grabacion\_filtrada.bag "topic not in <lista de topicos>"

3. Reproducir la grabación filtrada del experimento y poner en marcha el nodo de localización

roslaunch dwm1001\_drivers localization\_sim\_time.launch <nodes\_cfg\_label> <duration>

* 1. Para poder guardar los ficheros de texto con las posiciones de referencia y las lecturas de rangos entre *tags* y *anchors* será necesario descomentar las líneas correspondientes del nodo de localización dwm1001\_localization.py

# 5. Otra información relevante

## 5.1. Documentación adicional

* Página oficial del kit MDEK1001 de Decawave: https://www.decawave.com/product/mdek1001-deployment-kit/
* App Android oficial:

https://www.decawave.com/wp-content/uploads/2019/03/DRTLS\_Manager\_R2.apk

* Zip con toda la información referente del kit MDEK1001 incluyendo la guía API:

https://www.decawave.com/1001-license/