UWB based indoor localization module

Project Title	**Digital Technologies, Advanced Robotics and increased Cyber-security for Agile Production in Future European Manufacturing Ecosystems**
Project Abbreviation	**TRINITY**
Project Funding Scheme	**H2020 Innovation Action (IA)**
Call Identifier	**DT-ICT-02-2018: Robotics - Digital Innovation Hubs (DIH)**
Project Website	**http://www.trinityrobotics.eu/**
Project Start Date	**1.1.2019**
Authors	**Flanders Make, Ali Bin Junaid, Robotics Application Engineer**

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

Contents

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

1. Introduction

In order to operate the UWB localization system it is required to install the appropriate software and hardware components for the intended system configurations, given their application environment.

This document will provide instructions for:

1. Required hardware configurations for placement in the environment and on the mobile robot.
2. Installing localization software running on mobile robot.
3. Installing firmware running on UWB modules.
4. Calibration of the UWB module's antenna delays.
5. Determining optimal placement of fixed hardware in the environment.

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

2. Basic operating principle

UWB based indoor localization consists of beacons, and tags. The beacons, also called anchors, are reference points in the environment. The tags are the floating devices to be localized. By using a minimum of 3 anchors it is possible to find your position in a two-dimensional system.

2.1. Ranging operating principle

The most commonly used method of positioning uses basic geometry to estimate the position. By measuring the distance to a number of anchors with a known position it is possible to obtain your own position. If we measure a certain distance, then we know we will be in a circle of that radius around the anchor. If we make distance measurements with 3 anchors we see that our position is uniquely determined by the intersection of the three circles. This method is called trilateration (or multilateration if more than 3 anchors are used).
The difficulty of this approach lies in the fact that the measurements are not perfect. There will always be some noise on the measurements and because of this, the circles will not intersect at exactly one point. To circumvent this issue, we try to find the point that is closest to all circles.
To obtain the tag's position by measuring the distances to the anchors is done using radio waves. Radio waves are sent from one 'module' to another and measure the time of flight (TOF), or in other words, how long it takes. Because radio waves travel at the speed of light we can simply divide the time of flight by this speed to get the distance.
There are different positioning protocols, e.g. Two-way-ranging (TWR) and Time-difference-of-Arrival. For the remainder of this tutorial the TWR protocol will be the default.
In TWR, distance from a tag to an anchor is obtained by sending packets back-and-forth. By measuring how long it took for the packet response to return, the tag can estimate the distance to that anchor. For positioning, the tag initiates communication with the anchors to obtain ranges towards the anchors, one by one. This process is further on referred to as ranging. Once the tag has ranged with at least three, ideally four or more anchors, range data can be used to compute position.

In order to raise ranging accuracy further, and to obtain also orientation information a minimum of two tags is required. Although the position is instantaneous available, orientation will become available after displacement of the mobile robot.

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

3. System overview

3.1. System concept

The ranging hardware used is the Decawave TREK1000 evaluation kit [**Error! Reference source not found.**] containing four evaluation boards and antennas. For two-dimensional localization the minimum number of anchors is theoretical three, although four is preferred to boost performance and area coverage. Below figure shows the system concept.

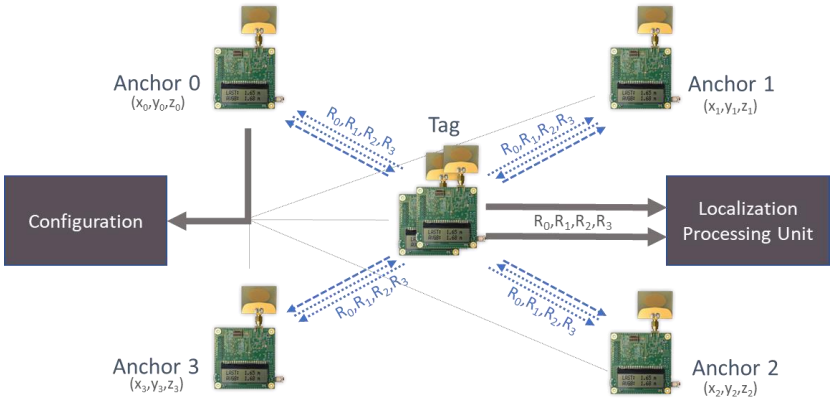


Figure 1 – System concept.

To obtain orientation information the ranges of two tags will be fused in the Localization Processing Unit, which can be e.g. a PC or single board computer like Raspberry PI-3.

3.2. Hardware architecture

Since for localization it is evident that a processing unit is required solving both the computational requirements as power management. However for anchors the configuration options are dependent on the power management structure. Figure 2 gives two possible configurations.

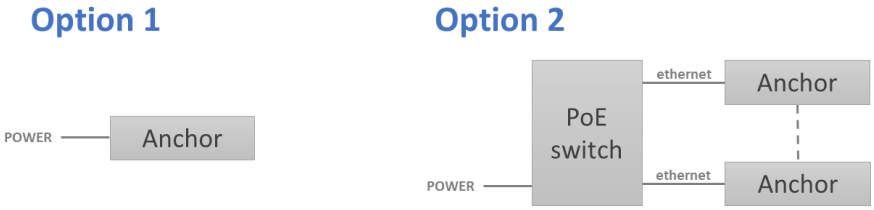


Figure 2 – Power management examples for anchors.

When installing the anchors into the environment their location need to be known, as well as anchors need to be powered. Anchors are typically continuously powered, thus powering through batteries is non-preferred. Secondly, anchors are potentially hard to reach and should therefore be easily upgradable with new firmware without risk of impacting the exact known position or personal injuries. A solution is to have anchors connected and powered by an Ethernet switch (option-2), providing the ability to access them from a single point and benefit full remote control during system debug, system wide (re)configuration and future maintenance. Any computer can be connected through a DHCP router to the switch.

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

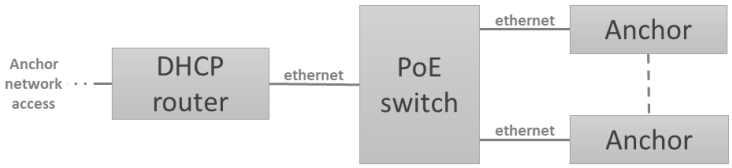


Figure 3 – Single point access and powering to anchors.

Each anchor is not only containing a TREK1000 evaluation board and an ST Link programmer as depicted in Figure 4, but also a PI-3 compute system. This combination of components is called: UWBox.

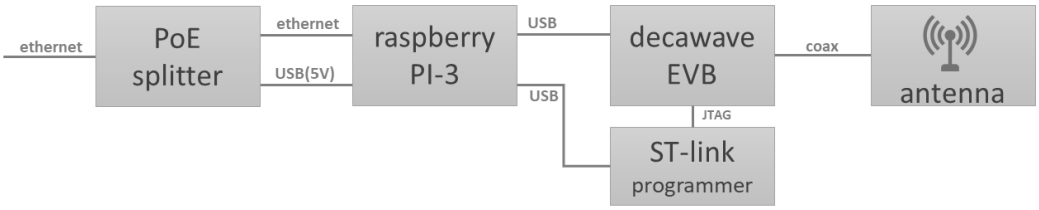


Figure 4 – Hardware architecture of anchors.

Figure 5 – Example UWBox implementation.

All software on UWBoxes will be kept the same using distribution scripts. To support easy linking between anchor ID numbers (see 4.1) and the actual UWBox, each UWBox will be given a static IP-address. This IP-address becomes then an unique identifier also used for other purposes, e.g. antenna calibration (see chapter 7).

3.3. Software architecture

The software will be split into next main sections:

1. Firmware for the TREK1000 evaluation boards.
2. ROS Kinetic based localization running on a Localization Processing Unit.

The TREK1000 evaluation boards are flashed with optimized firmware compared to Decawave COTS firmware. The DIP Switch on these boards determine the initial configuration e.g., tag or anchor.

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

Within the Localization Processing Unit the actual ranging data is read over USB from each tag's firmware API and published as self-defined message on a ROS topic by *USB2topic*[1] submodule. The *USB2topic* submodule publishes its ranges stemming from each as tag configured TREK1000 evaluation board. The *Localization* submodule subscribes to these ROS topics and publishes the two-dimensional position (X,Y), orientation and linear and angular velocities.

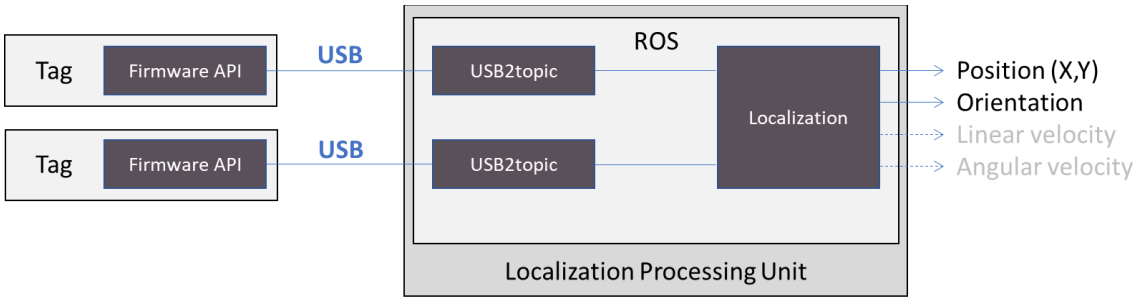


Figure 6 – Dual-tag based Localization Processing Unit software architecture.

The *Localization* submodule performs Kalman filter based sensor fusion.

[1] Arbitrary name to indicate primary function.

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

4. System components

This chapter does describe the installation of the components introduced in the previous chapter. First hardware related is outlines

4.1. Hardware

The minimum component required is a TREK1000 evaluation board with provided antenna attached. With the default Decawave firmware the role, tag or anchor, can be easily set via a DIP Switch. This approach is maintained, however our applications requires access to other UWB settings than the original TREK1000 firmware provides. Therefore our firmware uses a different DIP Switch configuration (see Figure 7and Table 1) than shown on the sticker placed on the TREK1000 evaluation board.
The DIP Switch is located at the LCD panel side of the evaluation board.

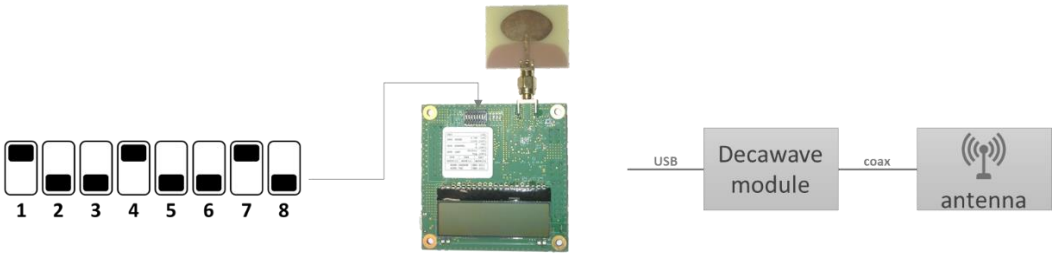


Figure 7 – TREK1000 evaluation board with DIP Switch highlight and schematic representation.

Table 1, DIP Switch configurability.

Switch #	Description	Up	Down
1	ON	On	Off
2	Role	Anchor	Tag
3	Channel select	5	2
4	Preamble length	128	64
5	PRF	64	16
6-8	ID number	down = 0 → x000… x111 = 0 … 7	

All TREK1000 evaluation boards shall be unique, which can be configured through the ID 'field'. For a minimum system configuration of 4 anchors and 2 tags example settings are shown in Table 2.

Table 2, Example settings for a minimum system configuration.

Switch #	Description	State
1	ON	On
2	Role	Anchor 4x / Tag 2x
3	Channel select	2
4	Preamble length	128
5	PRF	16
6-8	ID number	Anchor: 0 .. 4 Tag: 0, 4

When using multiple tags it is most optimal to have range measurement with equidistant time intervals.

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

Since there is an update rate per tag, and a maximum of 8 tags partake per ranging cycle, an equidistant timing can be achieved using tag ID *x* and *x+4*.

4.2. Software

4.2.1. Database directory structure

The approach taken is from an application view, typically referenced as appl_uwb_<name>.
The directory structure is given in Figure 8, containing python based ROS packages for the application and its sub modules as well as documentation, supporting tools and firmware. The numbering shown is for later reference purposes only.

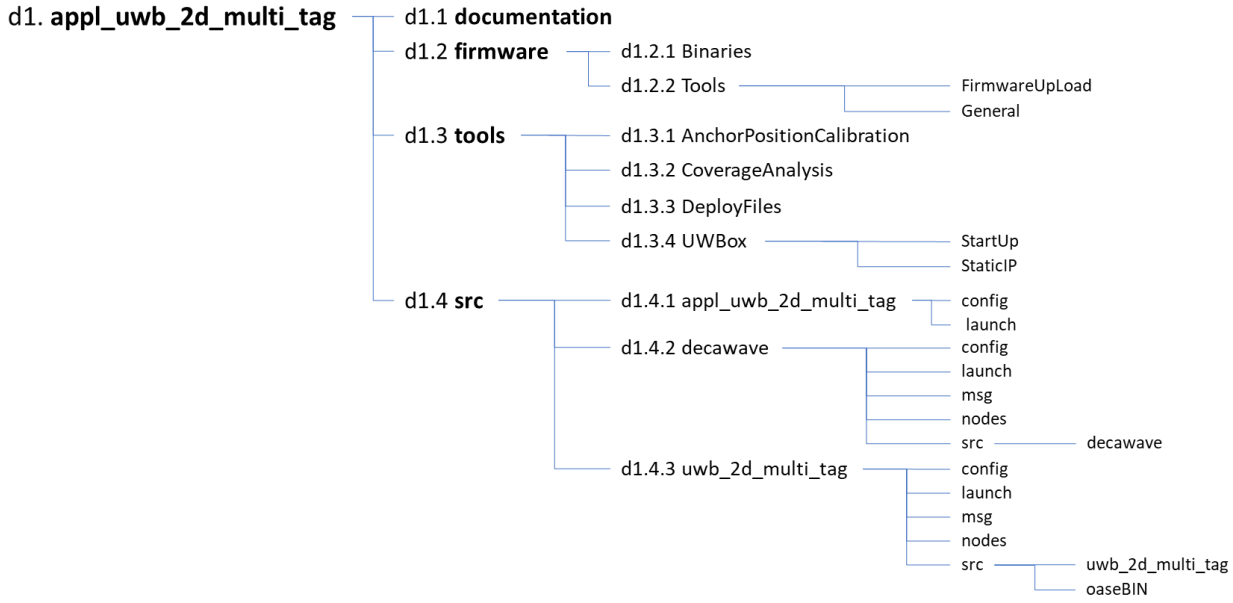


Figure 8 – Database directory structure.

4.2.2. Firmware

In order to flash firmware an ST Link programmer need to be attached to the JTAG interface of the evaluation board (Figure 9).

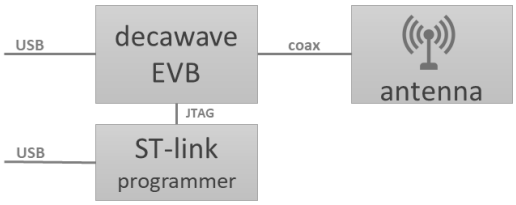


Figure 9 – TREK1000 evaluation board with ST Link connected for firmware updates.

To update firmware, script [F1] can be run with the appropriate firmware binary.

#	**Shell** script name	Short description
F1	flashmcu <binary.elf>	Updates the firmware of the connected TREK1000 module with the <binary.elf> code

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

Both script and binary can be found at *d1.2.2/FirmwareUpLoad* and *d1.1*.

Table 3 list the available firmware release with indication of the maximum amount of tags and anchors supported in a system, as well as the ranging mode and update rate.

Table 3, Firmware releases.

Version	# tags	# anchors	Mode(s)	Update rate	Remark
2.1.0	8	8	TWR	80ms/12.5 Hz	Per tag, 8 anchor ranges are provided every 10ms

4.2.3.Application specific localization software

The application specific localization software is provided as a ROS package.

Following applications are supported:

Table 4, Supported applications.

Nr.	Application	Package name	Topics published
A1	Multi tag 2D localization without fusion or vehicle model	appl_uwb_2d_multi_tag	UWBPose Message type: PoseWithCovarianceStamped

Note: launch file name is package name of with .launch extension.

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

5. Software installation

5.1. Prerequisites

5.1.1.Operating systems

Device	Operating system type
Anchor	Raspbian OS - headless
Tag	
Location Processing Unit	Ubuntu 16.04

5.1.2.Packages

Package	Installation
python2.7	-
python3	-
pyyaml	pip3 install pyyaml
regex	pip install regex
ipdb	pip install ipdb
pyserial	pip install pyserial
argparse	pip install argparse
sockets	pip install sockets
numpy	pip install numpy
python-math	pip install python-math
matplotlib	pip install matplotlib
casadi	pip install casadi
python-rospy	sudo apt install python-rospy
pickle-mixin	pip install pickle-mixin
catkin	sudo apt install ros-kinetic-catkin
unzip	Sudo apt install unzip

5.1.3.ROS workspace

First installation of ROS kinetic is required; see *http://wiki.ros.org/kinetic/Installation/Ubuntu*.

Than build a catkin workspace as following (*http://wiki.ros.org/catkin/Tutorials/create_a_workspace*).

```
$ source /opt/ros/kinetic/setup.bash
$ mkdir -p ~/workspace/src
$ cd ~/workspace
$ catkin_make
$ source devel/setup.bash
$ echo $ROS_PACKAGE_PATH
/home/<user>/workspace/src:/opt/ros/kinetic/share
```

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

5.2. Build localization application

In this catkin workspace the database can be unzipped and the application can be build.

```
$ cd workspace/src
$ unzip appl_uwb_2d_multi_tag
$ cd ..
$ catkin_make
$ source devel/setup.bash
```

5.3. Userspace device manager setup

Unfortunately TREK1000 evaluation boards and firmware do not have a means of providing an unique identifier. Therefor the physical USB port to which an evaluation board is connected will be used to setup a symbolic Linux device. In this approach the physical USB socket determines which evaluation board connects to which ROS Decawave reader node (earlier referred to as USB2topic in 3.3).
This is done through *udev* (userspace */dev*), which is a device manager for the Linux kernel and primarily manages device nodes in the */dev* directory. At the same time, *udev* also handles all user space events raised when hardware devices are added into the system or removed from it, making the plugin order irrelevant.

To install these symbolic links the following file and content need to be added:

File	**/etc/udev/rules.d/99-usb-serial.rules**
Content	KERNEL=="ttyUSB*", MODE:="0777" KERNEL=="ttyACM*", MODE:="0777" SUBSYSTEM=="tty", SUBSYSTEMS=="usb", KERNELS=="**<port1>**", SYMLINK+="decawaveT0" SUBSYSTEM=="tty", SUBSYSTEMS=="usb", KERNELS=="**<port2>**", SYMLINK+="decawaveT4"

The physical port definitions for <port1> and <port2> can be found following next steps

1. Run command
   ```
   $ dmesg
   ```
2. Plugin a single USB plug in the intended physical port
3. Run command
   ```
   $ dmesg
   ```
4. The delta between the output of step-1 and step-2 typically is something like
   ```
   [123456.123456]usb 1-1: new …
   ```
   The sequence after 'usb', in this case '1-1' is to be filled in at <port*X*> in the *udev* rules file.

To effectuate these rules a system reboot is required (normal and future operational way of working) or execution of next commands if immediate effectuation without reboot is preferred:

```
$ udevadm control --reload-rules; udevadm trigger
```

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

Finally check of the links do exist

```
$ ls /dev/decawaveT*
```

5.4. Configuration files

In order to describe the UWB localization system, a set of configuration files in yaml format are provided for completion in the application's *d1.4.1/config* directory.
The following is an example launch file referring a set of configurations files. It has been an deliberate choice to capture related parameters together into separate files. E.g. environment specific parameters can be easily swapped/redefined when moving to a different physical location.

File	**../appl_uwb_agv_1.launch**
Content	<launch> <group ns="appl_uwb_2d_multi_tag" > <rosparam command="load" file="$(find appl_uwb_2d_multi_tag)/config/config_decawave.yaml" /> <rosparam command="load" file="$(find appl_uwb_2d_multi_tag)/config/config_anchor_placement.yaml" /> <rosparam command="load" file="$(find appl_uwb_2d_multi_tag)/config/config_application.yaml" /> <rosparam command="load" file="$(find appl_uwb_2d_multi_tag)/config/config_system_runTime.yaml" /> <include file="$(find decawave)/launch/publish_dw_uwb_ranges.launch" > <arg name="node_name" value="decawave_uwb_range_T0" /> <arg name="tag_label" value="T0" /> </include> <include file="$(find decawave)/launch/publish_dw_uwb_ranges.launch" > <arg name="node_name" value="decawave_uwb_range_T4" /> <arg name="tag_label" value="T4" /> </include> <include file="$(find uwb_2d_multi_tag)/launch/uwb_2d_multi_tag.launch" > </include> </group> </launch>

It shall be noted that component, environment and application specific parameters are static. While the runtime parameters are (slightly) more dynamic.

5.4.1.Component specific configuration file

Following parameters are defined for the ROS Decawave reader (earlier referred to as USB2topic) node.

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

TagInterface:

usb_port: <port_name>
baud_rate: <baud_rate>
ros_usb_rate: <ros_usb_rate>
reader_name: <reader_name>

File	**Reader_config_decawave.yaml**	
Parameters	<port_name>	Symbolic port name of the USB connection to TREK1000 module, which will be extended with the <tag_id> (see 5.4.3). The combination should match the symbolic name used by the device manager (see 0)
	<baud_rate>	Baud rate of the USB connection to TREK1000 module
	<ros_usb_rate>	The polling rate of ROS on the USB connection to TREK1000 module
	<reader_name>	Used at topic /<reader_name>/ranges/<tag_id> See for <tag id> chapter

5.4.2.Environment specific configuration file

The exact absolute position of anchors are to be defined with following parameters

AnchorsPosition:
anchor1:
ID: <anchor_id>
coordinates:
x : <x>
y : <y>
z : <z>
anchor2:
ID: <anchor_id>
coordinates:
x : <x>
y : <y>
z : <z>
anchor …

File	**system_config_anchor_setup.yaml**	
Parameters	<anchor_id>	The id should match the physical anchor numbering. Advised numbering: A0…n

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

	<x>	X coordinate in world frame
	<y>	Y coordinate in world frame
	<z>	Z coordinate in world frame

5.4.3.Application specific configuration file

The exact relative position of tag(s) to the 'reference spot' to localize e.g. on a vehicle are to be defined with following parameters, similar to the anchor definition.

MaxTags: <max_tags>
TagsPosition:
tag1:
ID: <tag_id>
coordinates:
x : <x>
y : <y>
z : <z>
tag …

The *MaxTags* parameter is dependent of the firmware configuration and the choice of equidistance timing setup (see 0 Note that the earlier given access file name is expected by the deployment scripts.

).

The next parameters are an example, applicable for the [A2] application, and are therefore likely to change between applications or even are non-existent.

VehicleParams:
driving_wheel_offsset: <wheel_offset>
distance_driving_front_wheel: <wheel_distance>
vehicle_frame_ID: <frame_id>

File	**system_config_application.yaml**	
Parameters	<max_tags>	The number to tags supported by the firmware (e.g. firmware 2.1 supports 8 tags)
	<tag_id>	The id should match the physical tags numbering.

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

		Advised numbering: T0…n Within an application matches also the 'tag_label' required to launch the ROS reader node.
	<x>	X coordinate in application/vehicle frame
	<y>	Y coordinate in application/vehicle frame
	<z>	Z coordinate in application/vehicle frame
	<wheel_offset>	Driving wheel offset compared to the centerline
	<wheel_distance>	Driving wheel distance from the 'reference spot'
	<frame_id>	Name of the application/vehicle frame

5.4.4.Runtime specific configuration file

InitState:
position:
x : <x>
y : <y>
z : <z>
orientation: <orient>
log_to_file: <log>
log_to_file_covar: <covar_min_value>
print _running_average: <run_avg>

File	**system_config_runtime.yaml**	
Parameters	<x>	Initial X coordinate of the 'reference point' in meters
	<y>	Initial Y coordinate of the 'reference point' in meters
	<x>	Initial Z coordinate of the 'reference point' in meters
	<orient>	Initial orientation of the 'reference point' in radians
	<log>	Debug option: enables logging to a file for offline processing; 0= no logging, 1= logging
	<covar_min_value>	When logging is enabled, values will only be written to file when lower than specified value
	<run_avg>	Debug option: print the running average of incoming ranges

Note: debug options are not guaranteed to work for all applications.

5.5. Setting up a static IP-address for an UWBox

The static IP configuration as given by Table 5 can be used.

Table 5, Router IP address configuration.

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

Gateway	192.168.10.1
DHCP	192.168.10.50-99
Static – Anchors	192.168.10.100-199
Static – Tags	192.168.10.200-249

Make sure the router is setup accordingly, and that static IP addresses are given to each of the UWBoxes. This can be done with the following script executed at the PI-3 inside the UWBox:

#	**Shell** script name	Short description
S1	StaticIP <device> <id>	Script to set a static IP address <device> determines the role either: anchor or tag <id> ID number as per DIP Switch (see Table 1)

In addition it is strongly advised to change the *hostname*. Next time a remote shell accesses an UWBox will show the *hostname* immediately as confirmation of reaching the correct UWBox. The *hostname* can be update with next command:

```
$ hostname pi-<device>-<id>
```

Where <device> and <id> are the same as used during execution of script [S1].

Note that the StaticIP.sh script [S1] does update the */etc/dhcpcd.conf* file. There is a check included on possible existing static IP address configurations. If so, manual intervention on this file is suggested. This typically may occur when the StaticIP.sh script [S1] is ran for multiple times.

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

6. Tools

6.1. Distribution of software to all anchors

Once all UWBoxes are given unique IP-addresses the required soft/firmware can be distributed to all UWBoxes with a single command [S2] to be found in *d1.3.2 DeployFiles*.

#	**Shell** script name	Short description
S2	DeployFilesToAnchors	Script to copy from the local repository the required soft/firmware for UWBoxes

In addition an access file is required. It is assumed that the user/login ID remains 'pi', while the IP-address and associate password is given in the access file.

The access file <anchors_access.txt> has following format (space separated):

<*IP-address*> <*password*> # general format
192.168.10.100 raspberry # example format
…

Figure 10, Anchor access file format.

Note that the earlier given access file name is expected by the deployment scripts.

6.2. Anchor placement analysis

6.2.1. Optimal position in the environment

Given a space one should analyse the optimal position given the available space and placement possibilities within (e.g. doors not allowing to place anchors). A measurement technique from GPS positioning, GDOP (Geometric Dilution of Precision), will be used.

HDOP stands for the Horizontal Dilution Of Precision and is a part of the DOP measurement describing the geometric strength of the visible anchor configuration on the UWB localization accuracy in a horizontal plane (2D). Another one is PDOP (Position Dilution Of Precision), which is a measure of precision in horizontal and vertical (3D)

A low DOP value represents a better positional precision due to the wider angular separation between the satellites used to calculate a unit's position. Other factors that can increase the effective DOP are obstructions such as nearby mountains or buildings. HDOP values below 2 are considered excellent to ideal. See also Wikipedia.

Procedure to run the tool:

1. Open MATLAB
2. Open *plot_gdop_tof.m* in Matlab editor (location *d.1.3.2 CoverageAnalysis*).
3. Make sure the other file: *calculate_dop_tof.m* is in the path.
4. Change the part "anchors coordinate" with the actual anchors coordinate to be evaluated.
5. Change the maximum area to be evaluated if necessary.
6. Change the height of tag to which the DOP will be analysed.
7. Run the script.

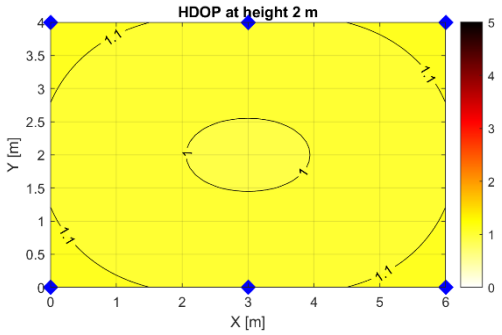
Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

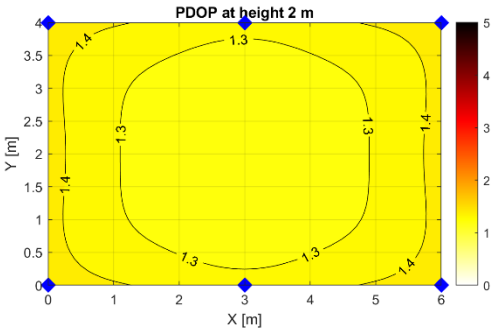
Next is the configuration part to be filled out:

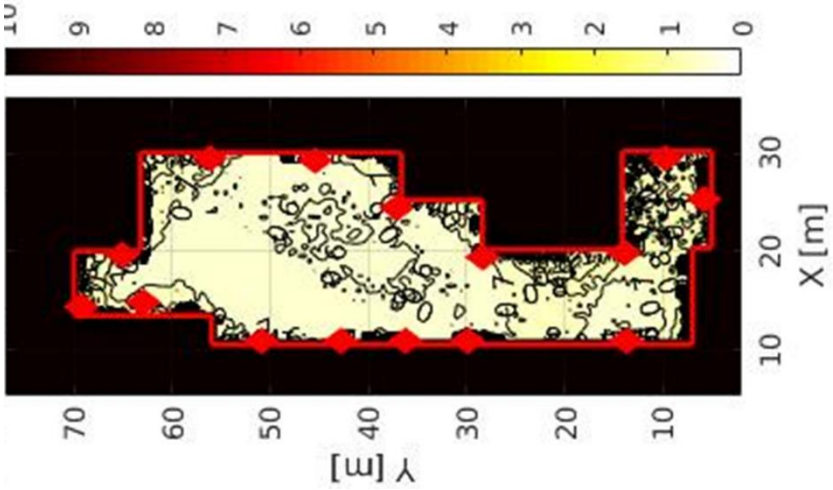
```
%% anchors coordinates
anchors = [
            0 0 4;
            0 4 4;
            3 0 4;
            3 4 4;
            6 0 4;
            6 4 4;
            ];

%% area to be evaluated
max_x = 6;
max_y = 4;
steps = 0.1;
x_true = [0:steps:max_x];
y_true = [0:steps:max_y];
z_true = 2;
```

After the analysis an optimal anchor placement within a given space is known. The output can per example look like:







Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

Figure 11 – Example HDOP analysis output.

6.3. Placement accuracy

In order to obtain the best possible result, position of anchors need to be known accurately. Errors in the anchor location will become part of the tags localization accuracy.
One need to have a reference plane and obtain the (x,y,z) coordinates of the antenna within that plane. Do note that the most common assumed reference plane is the floor of the environment, however this floor will not be a perfect plane, and as such introduce error.

6.4. Physical constraints

During the placement of the anchors the following constraints should be take into account:

1. The distance from walls to be more than 15 cm.

   When mounting the TREK1000 evaluation units do not place the antennas too close to walls or any other objects as this can interfere with the radioan pattern of the antennas. It is recommended that the antenna be more than 15 cm away from the nearest object.

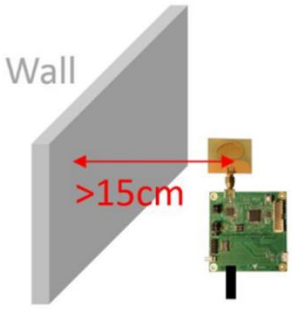


Figure 12 – Placement constraint against objects.

2. Metal mounting material or poles/tripods shall be below the antenna ground plane.

   When mounting the TREK1000 evaluation units on metal constructions, tripods or poles ensure that the highest point is below the ground plane of the antennas. The ground plane of the antenna is considered to be the top of the SMA connector under the antenna.

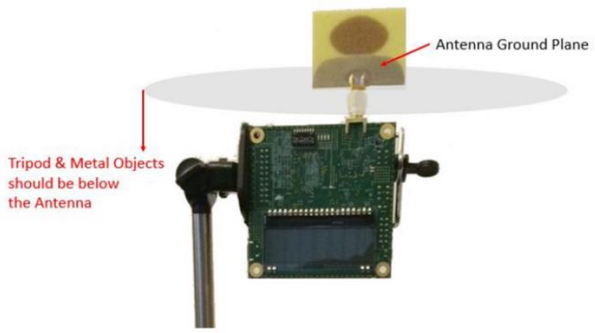


Figure 13 – Placement constraint against antenna ground plane (tripod example).

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

7. Antenna calibration

7.1. Introduction

Calibration of UWB modules is an important step to decrease the systematic error of distance measurements between two devices.

The type of systematic errors covered in this procedure are related to the propagation delay due to different antenna-configurations. With antenna-configuration is meant the combination of an UWB transceiver, PCB, an optional RF cable and chosen antenna.
After calibration the specific combination of calibrated parts shall be kept together and cannot be interchanged without recalibration.

The calibration procedure is compensating the time duration for the RF signal to traverse from UWB transceiver to antenna. This time duration is further referred to as antenna delay.

To calibrate the antenna delay, the measured distance is a known distance between two UWB devices. Antenna delay can be adjusted until the known distance and reported range agree.

Tools are provided for the calibration process itself, but also to ensure calibration values are correctly setup after power up. This assumes the options of an available One Time Programmable memory (OTP) or hardcoding into firmware is not chosen.

The current document is limited to Decawave TREK1000 evaluation kit based systems.

7.2. Calibration approach

The approach taken is a simplified version as documented by Decawave[2]. There are three steps. First a refence-device/tag is created to which secondly all the anchors will be calibrated. Thirdly, any further tag can be calibrated against one of the calibrated anchors, or preferably the tag can momentarily swap role to anchor and be calibrated against reference-device.

It shall be noted that use case dependent information regarding Pulse Repetition Frequency (PRF) and channels are to be taken into account.

Calibration is always performed in Two Way Ranging (TWR) mode, which is the aggregate (sum of) transmitter and receiver antenna delays. All anchors and tags require calibration, to reduce the error compared to the predefined/shipped TREK1000, average antenna delays.

Tools to read and write delay parameters or read ranges do operate both on a tag or anchor, thus the actual mode of operation (tag or anchor) has no influence.

Each device has an additional so called range bias compensation. One can choose to calibrate the antenna delay such that the zero point of the range bias moves towards higher or lower signal levels

[2] APS014 Application note; Antenna calibration of DW1000 based products and systems, version 1.2, Decawave 2018

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

depending on your application and whether you need accuracy at very short ranges or not[3]. However, here we follow the next calibration distances used for the different channels and different PRFs.

Table 6, Calibration distance per channel and PRF for PHY data rate 6.8 Mbps with Smart Power Control is enabled (apply 6 dB gain on transmission with frame duration < 1 ms).

Channel number	**Fc** (MHz)	**Bandwidth** (MHz)	**PRF** (MHz)	**Calibration distance [min - max]**(m)
1	3494.4	499.2	16 64	[17.69 – 19.00] [11.47 – 12.45]
2	3993.6	499.2	16 64	[15.40 – 16.71] [10.16 – 10.81]
3	4492.8	499.2	16 64	[13.76 – 14.74] [8.85 – 9.83]
4	3993.6	900	16 64	[8.50 – 9.00] [5.25 – 5.50]
5	6489.6	499.2	16 64	[10.95 – 11.71] [7.18 – 7.93]
7	6489.6	900	16 64	[5.25 – 5.50] [5.25 – 5.50]

The distances and the corresponding bias values are depicted in Figure 14. The values are for UWB settings: PHY data rate 6.8 Mbps with Smart Power Control enabled. The Smart TX Power Control is applied because when sending short data frames at this rate and providing that the frame transmission rate is < 1 frame per millisecond, it is possible to increase the transmit power and still remain within regulatory power limits which are typically specified as average power per millisecond. This transmit power increase will increase the link budget and communication range but will slightly change the bias value from other settings when additional power gain is not used.

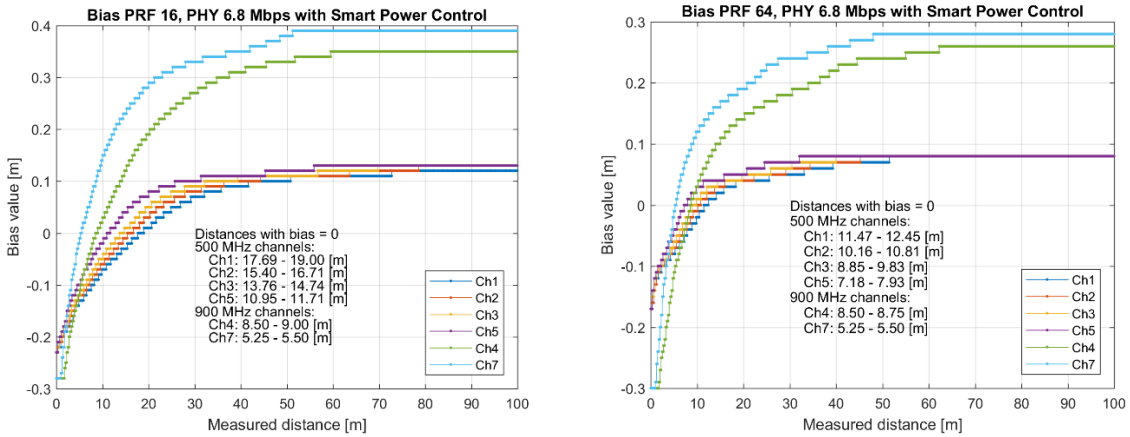


Figure 14, bias values for PRF 16 MHz (left) and 64 MHz (right), PHY data rate 6.8 Mbps, Smart Power Control enabled

[3] APS011 Application note; Sources of error in DW1000 based Two-Way_Ranging (TWR) schemes, version 1.0, Decawave 2014

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

When using lower PHY data rate (110 kbps or 850 kbps), Smart Power Control should be disabled because the frame length are typically more than 1 ms. The calibration distance for this settings are summarized in Table 7. The bias values for different distances are illustrated in Figure 15

Table 7, Calibration distance per channel and PRF for low PHY data rate settings with Smart Power Control disabled.

Channel number	**Fc** (MHz)	**Bandwidth** (MHz)	**PRF** (MHz)	**Calibration distance [min - max]** (m)
1	3494.4	499.2	16 64	[13.50 – 14.50] [8.75 – 9.50]
2	3993.6	499.2	16 64	[11.75 – 12.75] [7.75 – 8.25]
3	4492.8	499.2	16 64	[10.50 – 11.25] [6.75 – 7.50]
4	3993.6	900	16 64	[8.50 – 8.75] [5.26 – 5.50]
5	6489.6	499.2	16 64	[7.25 – 7.75] [4.75 – 5.25]
7	6489.6	900	16 64	[5.25 – 5.50] [5.25 – 5.50]

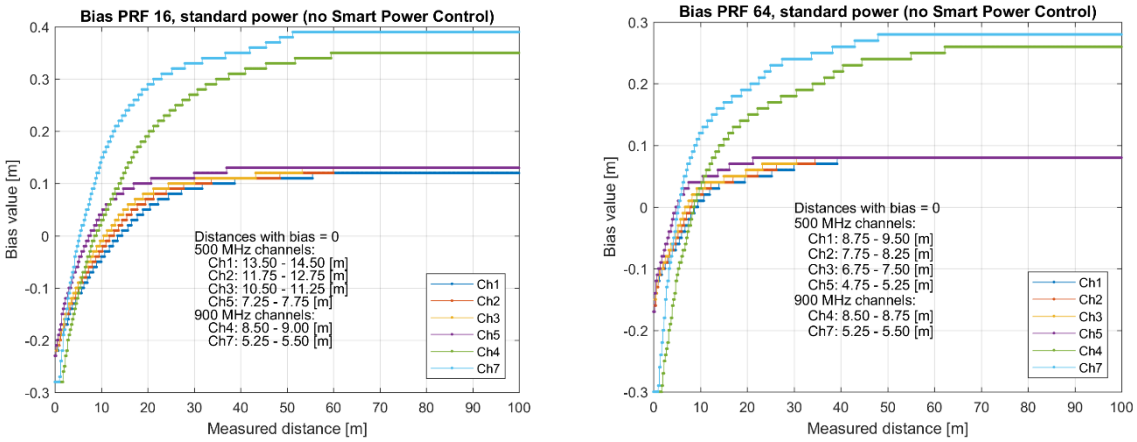


Figure 15, bias values for PRF 16 MHz (left) and 64 MHz (right), when Smart Power Control is disabled

7.2.1.Step 1 – Create reference device

The reference device is to be configured as tag and has the intended antenna-configuration. As counterpart, an out of the box/'average' UWB module can be used (see Figure 16), which is to be configured as anchor (see 4.1).

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

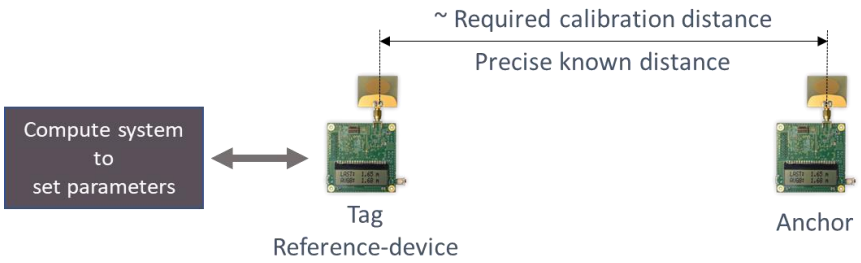


Figure 16, Example reference-device calibration setup.

The distance between the centre points of the antennas need to measured: the known distance. The known distance should be as close as possible to Table 6 and both modules shall have the same parameter setting.

For each intended parameter, mainly channel and PRF, the calibration can be performed by updating the respective values using the scripts and configuration file described later on.

The delay values are updated until the running average of range measurements is in agreement with the known distance.
Note that in TWR mode, the receive (Rx) and transmit (Tx) delays are set to the same values.

7.2.2.Step 2 – Calibration of anchors

Anchors shall be calibrated against the reference-device, configured as tag.

The known distance between any of the anchors and the reference-device should be as close as possible to the calibration distances of Table 6 and both modules shall have the same parameter setting.

The delay values are to be updated until the running average of range measurements is in agreement with the known distance. At this stage the reference-device is used to read the ranges, while each anchor is calibrated.

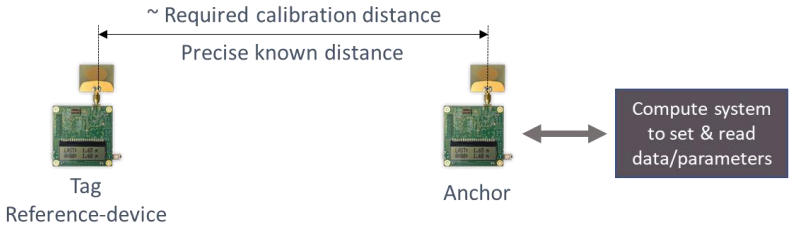


Figure 17, Example anchor calibration setup.

7.2.3.Step 3 – Calibration of tags

After performing step-2 the calibrated anchors can be used to calibrate tags.

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

The known distance between any of the anchors and the tag to calibrate should be as close as possible to the calibration distances of Table 6, where both modules shall have the same parameter setting. See next example setup.

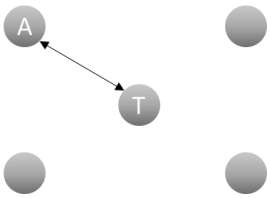


Figure 18, Further tags (T) calibration setup against a target-anchor (A).

Alternatively one can momentarily swap role from tag to anchor and follow step-2.

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

7.3. How to create … – Introduction

The following chapters describe how to actually create a reference-device and calibrate anchors and tags against this reference-device.

7.3.1. How to calibrate UWB localization system configurations

Although an OTP is foreseen inside the TREK1000 chipset, disadvantage of using is that OTPs are (as the name suggests) only programmable once. Therefore calibrated values can be uploaded each time devices start up, manually or use hard programming in firmware.

This chapter describes UWB localization system configurations and their related calibration values update approach.

There are at least two distinct UWB localization system configurations:

1. UWBox, typically used for anchors.
2. Integrated, direct connection to the TREK1000 evaluation board.

   Note: dedicated firmware updates are not further elaborated in this document, although feasible when the antenna delay values are known.

Following the earlier selected approach of anchors having a fixed hardware configuration, called UWBox, connected and powered by an Ethernet switch. This allows for evolution and maintenance of hardware and software configurations easily accessible from a single point.
Each UWBox consists of a TREK1000 evaluation board, a PI3 compute system and STlink in-circuit debugger/programmer (see 3.2). Static IP-addresses will be used as the unique identifier for loading parameters from a single configurations file. With this approach all software on all UWBoxes will be the same, including a single antenna delay configuration file, for ease of maintenance

UWBox approach – Example command line to update antenna delays:

```
configureAntennaDelay -config myconfig.yml
```

Since (mobile) tags do typically not allow for hardware ‘overhead’ as used in UWBoxes, they follow the integrated approach. In this case the unique identifier has to be supplied through the command line. Therefor next to the UWBox category there is also a tag category (details see next paragraph).

Integrated approach – Example command line to update antenna delays:

```
configureAntennaDelay -dev TAG -tagName myAGVtag -config myconfig.yml
```

Trinity project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 825196.

7.3.2.How to create an antenna delay configuration file

An antenna delay configuration file in YAML format is used to register any antenna delay value. The uniqueness of these values are related to either being an:

- UWBox, with a static IP address as unique identifier, or an
- Integrated module, with user defined unique identifiers.

Since mobile tags do typically not allow for hardware 'overhead' as used in UWBoxes, they follow the integrated approach. In this case the unique identifier has to be supplied through the command line. Therefor next to the UWBox category there is also a tag category.

Per unique identifier, different so called antenna configurations are possible. Per example calibration of a single UWBox was done for a TREK1000 antenna and for a dedicated antenna. All configurations can be registered in the configuration file and the one actually used can be indicated.

Do note that only antenna delay values given in a configuration file will be updated in the UWB module, thus other/existing values remain the same.

7.3.2.1. Programmable parameters

Parameter	Value	Description
biasCorrectionEnable	0 or 1	Disable '0' and enable '1'
ActualConfig	<name>_<num>	<name> will refer to the specific parameters selected to be reprogrammed for the specific UWBox. <num> will be extracted and show as sub-number on the display (format major.minor.<num>)
CH	<num>	The prefix 'CH' is nonstandard, however the number is required since this is used to set the physical channel delays
PRF16/64	<value>	The actual delay value for PRF16/64 for the channel

7.3.2.2. Configuration file structure

The antenna delay configuration file is using the *yaml* format:

```
UWBox:
    <IP address>:
        biasCorrectionEnable: 0 or 1
            ActualConfig:     <name> _<num>   # configuration to be used
            <name>_<num>:                     # configuration example 1
                DelayValues:
                    CH<n>:
                        PRF16: <value>
                        PRF64: <value>
                    CH<n>:
                        …
```

<*name*>_<*num*>: # configuration example 2
…
<*IP address*>: …

TAG:
<*tag name*>:
biasCorrectionEnable: *0* or *1*
ActualConfig: <*name*> _<*num*>
<*name*>_<*num*>:
DelayValues:
CH<*n*>:
PRF16: <*value*>
PRF64: <*value*>
CH<*n*>:
…
<*name*>_<*num*>:
…
<*tag name*>:

An example configuration file and resulting output is given in appendix 1.

7.3.3.How to create a reference-device

In case a reference-device is already existing it is strongly advised to use this device !

When a reference-device is not existing, this device can be created using an arbitrary delay value for the given configuration. An antenna delay value of 16500 is a practical approximation of a 'good' value. Best is to hardcode in firmware or use the one-time-programable (OTP) memory for this value. Otherwise updating the calibration values through the UWBox or integrated approach can be used.

Do note that for the reference-device NO physical changes in antenna path, through e.g. changes in PCB, antenna or physical connections, are permitted after a first calibration exercise!!

7.3.4.How to calibrate anchors and tags

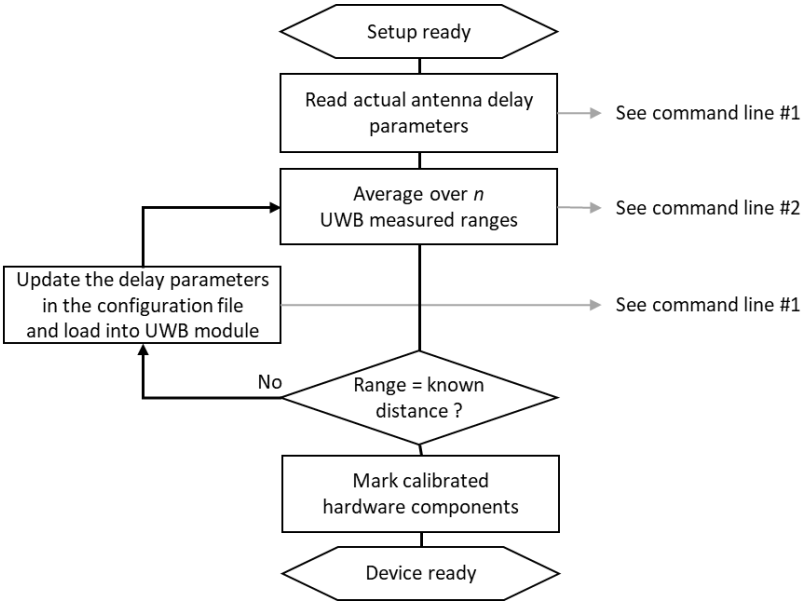
Precondition is to have the tag and anchor modules configured equally (for channel, PRF, …) and spaced with a known distance as shown in Figure 16. The known distance between any of the anchors and the reference-device should be as close as possible to the calibration distances of Table 6.

Note that by increasing antenna delays, the UWB measured range is reducing.

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

7.3.4.1. Manual calibration procedure

Manual calibration can be done by iteratively following next steps:



Flow diagram 1, Manual calibration flow

Use the following commands:

Read and write the delay parameters (based on script [P1]):

UWBox	
	configureAntennaDelay -dev UWBox -port /dev/ttyACM0 -config <your config file>
Integrated	
	configureAntennaDelay -dev TAG -tagName <name> -port /dev/ttyACM0 -config <your config file>

Command line 1, Reading and writing antenna delays.

Note that with Ubuntu 16.04LTS, TREK1000 modules enumerate as /dev/ttyACM*x*, where *x*=0 per default in an UWBox.

Read range values (based on script [P2]):

readFMApp_serialBinary -port /dev/ttyACM0 -save <your logfile> -meas 100

Command line 2, Reading 100 range values into your log file in .csv format.

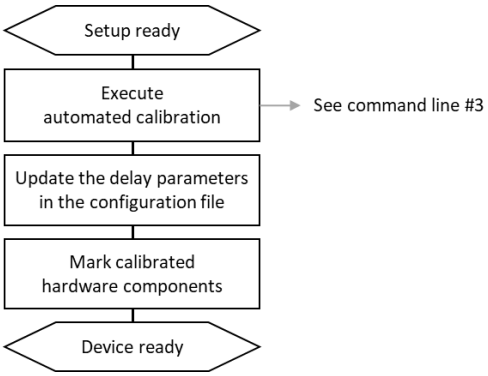
Since the firmware behaves equal during the calibration process for both an anchor or tag, it only depends to which device one connects.

7.3.4.2. Automated antenna delay calibration procedure

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

The automated antenna delay calibration procedure is using a binary search algorithm where the minimum antenna delay value is set to 15000 and maximum value to 18000. These values have proven to be sufficiently spread around the commonly seen 16500 antenna delay value for different TREK1000 modules.
The calibration is considered successful when the known distances matches within ±5mm of the average of 100 measured UWB ranges.

Automated calibration can be done by following next steps:



Flow diagram 2, Automated calibration flow

Start automated calibration (based on script [P3]):

```
calibrate -port /dev/ttyACM0 -anchorNr <your anchor>  -dist <known distance>
```

Command line 3, Reading 100 range values into your log file in .csv format.

Note that with Ubuntu 16.04LTS, TREK1000 modules enumerate as /dev/ttyACM*x*, where *x*=0 per default in an UWBox.

Since the firmware behaves equal during the calibration process for both an anchor or tag, it only depends to which device one connects.

7.3.4.3. Where to store the antenna delay configuration file

The antenna delay configuration file <config_antenna_delays.yml> shall be stored in the *d1.4.1/config* directory of the database:

appl_uwb_2d_multi_tag
src
appl_uwb_2d_multi_tag
config
config_antenna_delays.yml

Figure 19, Directory to store the antenna delay configuration file.

Note that the earlier given configuration file name is expected by the deployment scripts.

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

7.3.5.How to setup for correct system start-up

7.3.5.1. UWBox start-up

Since UWBoxes operate standalone, a start-up script is therefor to be executed during the boot process. This can be done to insert the next line into the */etc/rc.local* file of PI3:

Line to be added	Description
sudo -H –u <*user*> <*path*>/StartUp_UWBox.sh	During boot the shell script will be called with the permission of the indicated <*user*>

This shell script (example code in script [S3]) will first sleep a given time to allow the system to start-up its ethernet network. Then a python script [P4] is called to check if required USB port (*/tty/ACM0* on Linux Ubuntu 16.04 and up) is available as TREK1000 device connection. Finally python script [P1] is called to upload the antenna delay calibration values.
Please edit the shell script (in *d1.3.4 UWBox*) to match your specifics.

To check if the start-up completed successfully, two methods can be used:

1. The LCD screen on the TREK1000 device will show an update of the last digit of the version number, specified by antenna delay configuration number, defined in the configuration file. Format: <*major*>.<*minor*>.<*configuration*>
2. File */var/log/boot.log* provides an overview of the boot process and can be used to check the previous start-up process in detail.

7.3.5.2. Integrated start-up

From an integrated perspective the start-up procedure is very similar to that of the UWBoxes (see 7.3.5.1). Instead of calling the start-up script during boot, the script can be integrated in the application start-up procedure. Note that there is NO initial sleep time. Script [S2] provides example code.

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

7.4. Supporting scripts

#	**Shell** script name	Short description
S3	StartUp_UWBox.sh	Updates the antenna delay parameters in the UWBox UWB module as defined in the configuration file
S4	StartUp_Integrated.sh	Updates the antenna delay parameters UWB module as defined in the configuration file

#	**Python(3)** script name	Short description	
P1	configureAntennaDelay	Read current and write new antenna configuration: en/disable bias compensation, channel delays for Rx, Tx for PRF 16 and 64	
		-dev	Type of device, either *UWBox* or *TAG* Default: *UWBox*
		-tagName	If <*dev*> is *TAG* this *tagName* will be used
		-port	USB port where the TREK1000 is connected to Default: */dev/ttyACM0*
		-config	Configuration file name and path (see **Error! Reference source not found.**) Default: *config.yml*
P2	readFMApp_serialBinary	Read current and write new antenna configuration: en/disable bias compensation, channel delays for Rx, Tx for PRF 16 and 64	
		-port	USB port where the TREK1000 is connected to Default: */dev/ttyACM0*
		-file	Output filename to save to Default: file with name – *log<timestamp>.csv*
		-mode	Mode: 0 for TWR, 1 for ToA/TDoA Default: 0 (TWR)
		-role	Role: 0 for tag, 1 for anchor. Default: 0 (tag)
		-twma	Max anchors in 1 cell TWR mode Default : 8
		-twmt	Max tags in 1 cell TWR mode Default : 8
		-save	If set to 1, then measurement is saved to file indicated in argument -file If set to 0, measurement will not be saved to file Default: 1
		-meas	Number of measurements Default: unlimited
P3	calibrate	Read current and write new antenna configuration: en/disable bias compensation, channel delays for Rx, Tx for PRF 16 and 64	
		-dist	Known distance between UWB modules (meters) Default: *8.0*
		-anchorNr	Anchor number of the module as setup per dipswitch Default: *0*
		-port	USB port where the TREK1000 is connected to

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

			Default: */dev/ttyACM0*
P4	waitForUSBportOfUWB	Wait sufficient time to ensure UWB connection to TREK1000 is available and ready to use	
		-port	USB port where the TREK1000 is connected to Default: */dev/ttyACM0*
		-time	Waiting time for udev after physical port detection Default: *5* sec

7.5. Expected directory structures

7.5.1. UWBox approach

For the UWBox approach a dedicated directory structure shall be built to support antenna calibration (updates). Following structure (amongst others) is assumed due to the use of relative references:

/home/pi
/*PythonDecawave*
/*ShellScripts*

Directory *ShellScripts* should contain the following scripts, manually copied from their repositories:

#	**Shell** script name	Short description
S3	StartUp_UWBox.sh	Updates the antenna delay parameters UWB module as defined in the configuration file

Directory *PythonDecawave* should contain the following scripts, manually copied from their repositories:

#	**Python(3)** script name	Short description
P1	configureAntennaDelay	Read current and write new antenna configuration: en/disable bias compensation, channel delays for Rx, Tx for PRF 16 and 64
P4	waitForUSBportOfUWB	Wait sufficient time to ensure UWB connection to TREK1000 is available and ready to use
P5	configureUWBsettingsToaTwr	Read / Write settings. Only used to read in order to update the LCD screen data after execution of [P1]

The scripts [S3] and [P4] are located in the application's *tools/UWBox* repository, while scripts [P1] and [P5] are located in the *firmware/tools* repository.

7.5.2. Integrated approach

For the integrated approach a dedicated directory shall be build. The current assumption is to name this directory *UWBstartup.dir* placed in the Linux system's home directory due to the use of relative references. This directory should contain the following scripts, manually copied from their repositories:

#	**Shell** script name	Short description

S4	StartUp_Integrated.sh	Updates the antenna delay parameters UWB module as defined in the configuration file
S5	waitForUSBportOfUWB	Wait sufficient time to ensure UWB connection to TREK1000 is available and ready to use

#	**Python(3)** script name	Short description
P1	configureAntennaDelay	Read current and write new antenna configuration: en/disable bias compensation, channel delays for Rx, Tx for PRF 16 and 64
P4	waitForUSBportOfUWB	Wait sufficient time to ensure UWB connection to TREK1000 is available and ready to use
P5	configureUWBsettingsToaTwr	Read / Write settings. Only used to read in order to update the LCD screen data after execution of [P1]

The scripts [S4], [S5] and [P4] are located in the application's *tools* repository, while scripts [P1] and [P5] are located in the *firmware*/*tools* repository.

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

8. Bringing up the system

In order to have the maximum accuracy out of the system earlier described steps are assumed to have been taken into consideration. The more accurate these preparations are executed, the better the localization result will be.

Following steps have to be taken:

1. Power the anchors so they get functional.
2. Run next command on localization processing unit:

```
$ roslaunch appl_2d_multi_tag appl_uwb_multi_tag.launch
```

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

9. Terminology & Abbreviations

Anchor	UWB device on a fixed location
Antenna-configuration	combination UWB transceiver, PCB, RF cable and antenna
Antenna delay	duration for the RF signal to traverse from UWB transceiver to antenna
COTS	Commercially Of The Shelf
DHCP	Dynamic Host Configuration Protocol (DHCP) is a protocol for assigning dynamic IP addresses to devices on a network
DIP Switch	Dual Inline Package (physical layout) of a set of on/off switches
(H)DOP	(Horizontal) Dilution Of Precision for geometric strength of anchor placement on position accuracy
Fc	Centre Frequency
LCD	Liquid Crystal Display
Linux 16.04	Open source operating system with long term support of version 16.04
OTP	On Time Programmable
PC	Personal Computer
PI-3	Raspberry's single board computer, version 3
PRF	Pulse Repetition Frequency
Range	measured distance by UWB between two UWB modules
Reference-device	a tag with calibrated antenna-configuration
RF	Radio Frequency
ROS	Robot Operating System
ST	ST Microelectronics
Tag	UWB device which is mobile within the delimited space by anchors
TOF	Time Of Flight
TREK1000	Decawave IEEE802.15.4-2011 compliant evaluation kit

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

TWR	Two Way Ranging
UWB	IEEE802.15.4 standard
UWBox	Hardware compilation of an UWB module with (re)programmability possibilities and computer for maintenance and updates (e.g. calibration values)

10. Appendix 1 – Example of the calibration values update process

10.1. Example configuration file

```
UWBox:
  192.168.10.200:
    biasCorrectionEnable: 1

    # Actual antenna configuration for this UWBox with the data listed below
    ActualConfig: AntennaConfig_3

    # Theoretical values, as per firmware default
    # shown only as reference
    AntennaConfig_0:
      DelayValues:
        CH1:
          PRF16: 16483
          PRF64: 16489
        CH2:
          PRF16: 16469
          PRF64: 16471
        CH3:
          PRF16: 16459
          PRF64: 16462
        CH4:
          PRF16: 16418
          PRF64: 16422
        CH5:
          PRF16: 16455
          PRF64: 16461
        CH7:
          PRF16: 16408
          PRF64: 16411

    # Your antenna configuration - values for calibrated channels only
    AntennaConfig_3:
      DelayValues:
        CH1:
          PRF16: 16781
          PRF64: 16791
        CH2:
```

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

```
            PRF16: 16777
            PRF64: 16780
          CH3:
            PRF16: 16775
            PRF64: 16780

  192.168.10.201: # Another UWBox

    biasCorrectionEnable: 1

    # Actual antenna configuration for this UWBox with the data listed below
    ActualConfig: AntennaConfig_3

    # Camco antenna via 1m cable - values for calibrated channels only
    AntennaConfig_3:
      DelayValues:
        CH2:
          PRF16: 16777
          PRF64: 16780


TAG:

  AGV_T0:  # REFERENCE tag

    biasCorrectionEnable: 1

    # Actual antenna configuration for this UWBox with the data listed below
    ActualConfig: AntennaConfig_3

    # Camco antenna via 1m cable - values for calibrated channels only
    AntennaConfig_3:
      DelayValues:
        CH2:
          PRF16: 16757
          PRF64: 16763

  AGV_T4:

    biasCorrectionEnable: 1

    # Actual antenna configuration for this UWBox with the data listed below
    ActualConfig: AntennaConfig_3

    # Camco antenna via 1m cable - values for calibrated channels only
    AntennaConfig_3:
      DelayValues:
        CH2:
          PRF16: 16767
```

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

PRF64: 16775

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

10.2. Example output of script [S3]

```
Current Antenna Delay Table:

   Antenna Type :  0

   Delay      PRF16  |  PRF64
   -------------------+--------
   CH1 - Tx   16483  |  16489
      Rx   16483  |  16489
   CH2 - Tx   16469  |  16471
      Rx   16469  |  16471
   CH3 - Tx   16459  |  16462
      Rx   16459  |  16462
   CH4 - Tx   16418  |  16422
      Rx   16418  |  16422
   CH5 - Tx   16455  |  16461
      Rx   16455  |  16461
   CH7 - Tx   16408  |  16411
      Rx   16408  |  16411

   Bias correction enabled :  1

Configuration data:

   UWBox              : 192.168.10.200
   Requested configuration  :  AntennaConfig_3
   Found configuration      :  AntennaConfig_3
   Delay -  CH2  PRF16 / 64 :  16777 / 16780
   Delay -  CH1  PRF16 / 64 :  16781 / 16791
   Delay -  CH3  PRF16 / 64 :  16775 / 16780
   Bias correction enabled  :  1
   Antenna configuration #  :  3

New Antenna Delay Table:

   Antenna Type :  3

   Delay      PRF16  |  PRF64
   -------------------+--------
   CH1 - Tx   16781  |  16791
      Rx   16781  |  16791
   CH2 - Tx   16777  |  16780
      Rx   16777  |  16780
   CH3 - Tx   16775  |  16780
      Rx   16775  |  16780
   CH4 - Tx   16418  |  16422
      Rx   16418  |  16422
   CH5 - Tx   16455  |  16461
```

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

Rx 16455 | 16461
CH7 - Tx 16408 | 16411
Rx 16408 | 16411

Bias correction enabled : 1

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.