SOURCE CODE GUIDE

DECARANGERTLS PC SOURCE CODE

Understanding and using the DecaRangeRTLS PC source code

Version 1.00

This document is subject to change without notice.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

DOCUMENT INFORMATION

Disclaimer

DecaWave reserves the right to change product specifications without notice. As far as possible changes to functionality and specifications will be issued in product specific errata sheets or in new versions of this document. Customers are advised to check the DecaWave website for the most recent updates on this product

LIFE SUPPORT POLICY

DecaWave products are not authorized for use in safety-critical applications (such as life support) where a failure of the DecaWave product would reasonably be expected to cause severe personal injury or death. DecaWave customers using or selling DecaWave products in such a manner do so entirely at their own risk and agree to fully indemnify DecaWave and its representatives against any damages arising out of the use of DecaWave products in such safety-critical applications.

Caution! ESD sensitive device.

Precaution should be used when handling the device in order to prevent permanent damage

DISCLAIMER

This Disclaimer applies to the DecaRanging RTLS-ARM source code and the DecaRanging RTLS-PC source code (collectively “Decawave Software”) provided by Decawave Ltd. (“Decawave”).

Downloading, accepting delivery of or using the Decawave Software indicates your agreement to the terms of this Disclaimer. If you do not agree with the terms of this Disclaimer do not download, accept delivery of or use the Decawave Software.

Decawave Software incorporates STSW-STM32046 (STM32F105/7, STM32F2 and STM32F4 USB on-the-go Host and Device library (UM1021)) software (“STM Software”) provided to Decawave by ST Microelectronics (“STM”) under STM’s Liberty V2 software license agreement dated November 16th 2011 available here (“STM Software License Agreement”). Downloading, accepting delivery of or using STM Software as incorporated in Decawave Software indicates your agreement to the terms of the STM Software License Agreement and in particular the requirement that the STM Software be used only with STM microcontrollers and not with microcontrollers from any other manufacturer. If you do not wish to accept the terms of the STM Software License Agreement then you may still use the Decawave Software on the condition that you do not use the STM Software incorporated therein.

Decawave Software is solely intended to assist you in developing systems that incorporate Decawave semiconductor products. You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your systems and products. THE DECISION TO USE DECAWAVE SOFTWARE IN WHOLE OR IN PART IN YOUR SYSTEMS AND PRODUCTS RESTS ENTIRELY WITH YOU.

DECAWAVE SOFTWARE IS PROVIDED "AS IS". DECAWAVE MAKES NO WARRANTIES OR REPRESENTATIONS WITH REGARD TO THE DECAWAVE SOFTWARE OR USE OF THE DECAWAVE SOFTWARE, EXPRESS, IMPLIED OR STATUTORY, INCLUDING ACCURACY OR COMPLETENESS. DECAWAVE DISCLAIMS ANY WARRANTY OF TITLE AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS WITH REGARD TO DECAWAVE SOFTWARE OR THE USE THEREOF.

DECAWAVE SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY THIRD PARTY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON THE DECAWAVE SOFTWARE OR THE USE OF THE DECAWAVE SOFTWARE WITH DECAWAVE SEMICONDUCTOR TECHNOLOGY. IN NO EVENT SHALL DECAWAVE BE LIABLE FOR ANY ACTUAL, SPECIAL, INCIDENTAL, CONSEQUENTIAL OR INDIRECT DAMAGES, HOWEVER CAUSED, INCLUDING WITHOUT LIMITATION TO THE GENERALITY OF THE FOREGOING, LOSS OF ANTICIPATED PROFITS, GOODWILL, REPUTATION, BUSINESS RECEIPTS OR CONTRACTS, COSTS OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION), LOSSES OR EXPENSES RESULTING FROM THIRD PARTY CLAIMS. THESE LIMITATIONS WILL APPLY REGARDLESS OF THE FORM OF ACTION, WHETHER UNDER STATUTE, IN CONTRACT OR TORT INCLUDING NEGLIGENCE OR ANY OTHER FORM OF ACTION AND WHETHER OR NOT DECAWAVE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, ARISING IN ANY WAY OUT OF DECAWAVE SOFTWARE OR THE USE OF DECAWAVE SOFTWARE.

You are authorized to use Decawave Software in your end products and to modify the Decawave Software in the development of your end products. HOWEVER, NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER DECAWAVE INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY THIRD PARTY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT, IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which Decawave semiconductor products or Decawave Software are used.

You acknowledge and agree that you are solely responsible for compliance with all legal, regulatory and safety-related requirements concerning your products, and any use of Decawave Software in your applications, notwithstanding any applications-related information or support that may be provided by Decawave.

Decawave reserves the right to make corrections, enhancements, improvements and other changes to its software at any time.

Mailing address: -

Decawave Ltd.,
Adelaide Chambers,
Peter Street,
Dublin 8

1 INTRODUCTION

This document describes the DecaRangeRTLS PC application and its source code. The application connects to TREK1000 anchors or tags (running DecaRangeRTLS ARM application), and consumes the Time of Flight (TOF) reports coming from the anchors/tags and produces tag's location estimate based on the anchors' location.

The reader is encouraged to read the accompanying TREK documents, in particular the DecaRangeRTLS ARM Source Guide, and TREK1000 User Manual.

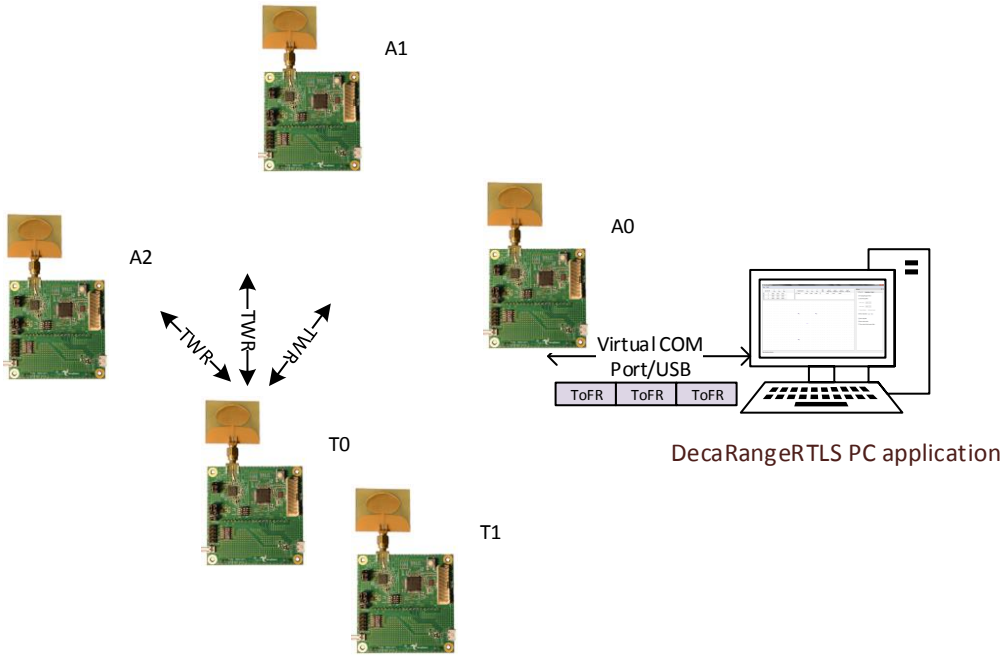


Figure 1: TREK1000 system components (with 2 tags)

The DecaRangeRTLS PC application is built with the Qt framework. Qt SDK Installation Guide describes how to set up and build the project. The main parts of the DecaRangeRTLS PC application are:

- Main window – this is the main user interface, it contains:
  - Drop down menus: View and Help
  - Table of known tags, anchors and the main display area, which shows position of tags and anchors on a floor plan.
- Serial connection – this is used to establish and monitor Virtual COM port connection with a TREK1000 Tag or Anchor over USB.
- RTLS client/Display client – this processes the TOF report messages from Virtual COM port connection, and sends location estimate to the main display.
- Anchor list – this is a list of known tags and it stores their properties (e.g. x, y, z coordinates etc.)
- Tag list – this is a list of known tags and it stores their properties (e.g. x, y, z coordinates etc.)
- Configuration – used for the use case settings and floorplan configurations
- Loading and Saving of anchor, tag parameters and view configuration

These are described in more detail in the following chapters.

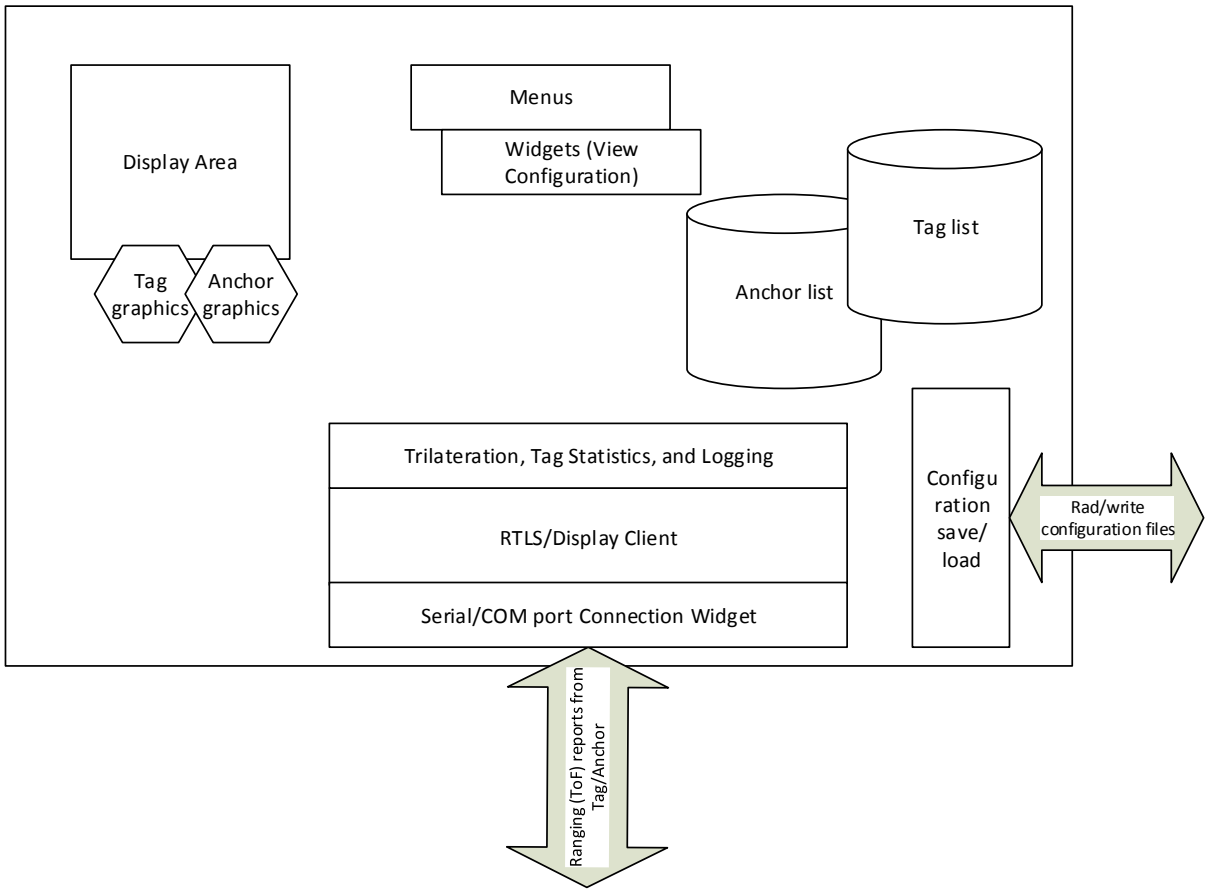


Figure 2: DecaRangeRTLS PC/RTLSDisplay application - main functional blocks

Table 1: List of source files in the DecaRangeRTLS ARM application

Filename	Brief description
main.cpp	This is the application main entry point
RTLSDisplayApplication.cpp	The DecaRangeRTLS application contructor – source file
RTLSDisplayApplication.cpp	The DecaRangeRTLS application contructor – header file
ViewSettings.cpp	View configuration class – source code
ViewSettings.h	View configuration class – header
RTLSClient.cpp	RTLSClient (consumes the TOF reports) class – source code
RTLSClient.h	RTLSClient (consumes the TOF reports) class – header
SerialConnection.cpp	Serial (COM port) connection management class – source code
SerialConnection.h	Serial (COM port) connection management class – header
AbstractTool.h	Abstract tool – header
OriginTool.cpp	Origin manipulation tool – source file
OriginTool.h	Origin manipulation tool – header file
RubberBandTool.cpp	Rubberband tool – source file
RubberBandTool.h	Rubberband tool – header file
ScaleTool.cpp	Scale manipulation tool – source file
ScaleTool.h	Scale manipulation tool – header file
trilateration.cpp	Trilateration functions – source code
trilateration.h	Trilateration functions – header file
QPropertyModel.cpp	QProperty Model – source code
QPropertyModel.h	QProperty Model – header file

Filename	Brief description
connectionwidget.cpp	Connection widget - source
connectionwidget.h	Connection widget - header
connectionwidget.ui	Connection widget - UI
GraphicsView.cpp	GraphicsView widget - source
GraphicsView.h	GraphicsView widget - header
GraphicsWidget.cpp	GraphicsWidget widget - source
GraphicsWidget.h	GraphicsWidget widget - header
GraphicsWidget.ui	GraphicsWidget widget - UI
mainwindow.cpp	mainwindow widget - source
mainwindow.h	mainwindow widget - header
mainwindow.ui	mainwindow widget - UI
MinimapView.cpp	MinimapView widget - source
MinimapView.h	MinimapView widget - header
ViewSettingsWidget.cpp	ViewSettingsWidget widget - source
ViewSettingsWidget.h	ViewSettingsWidget widget - header
ViewSettingsWidget.ui	ViewSettingsWidget widget - UI

2 DECARANGERTLS PC OVERVIEW

This chapter gives an overview about each of the DecaRangeRTLS PC application functional blocks and UI components as shown in Figure 2 and Figure 3:

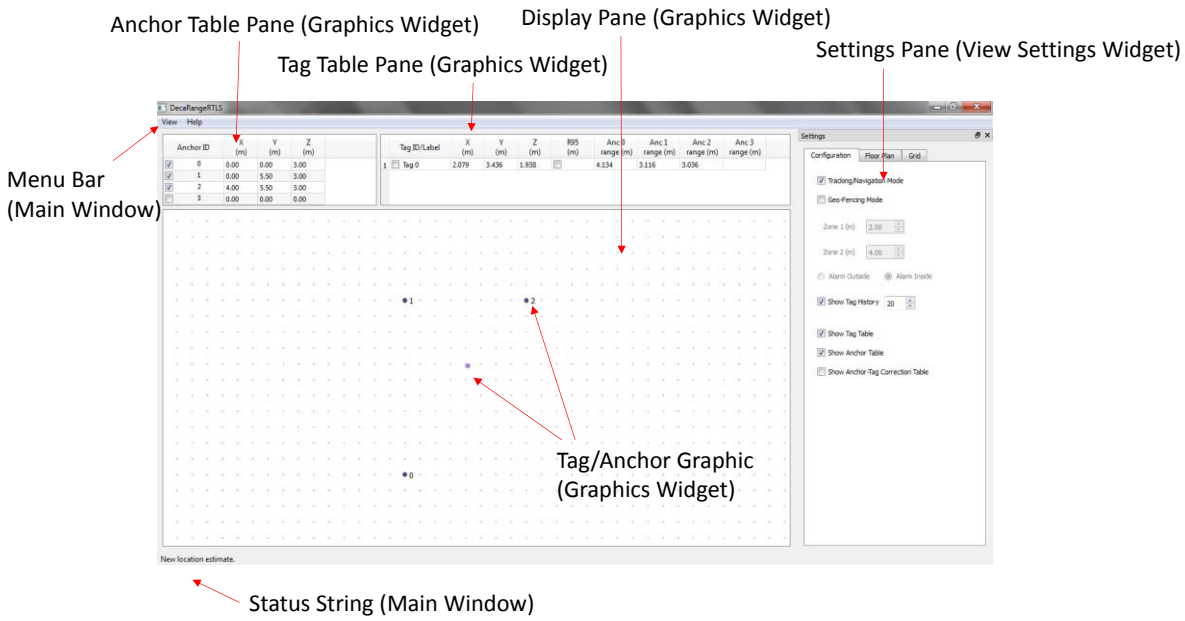


Figure 3: DecaRangeRTLS application – UI components

2.1 RTLSDisplay Application

The *RTLSDisplay Application* class initialises the application and its parts: the _serialConnection is used for managing the COM port connection, the _client consumes the data received over the COM port connection and sends the processed data to the graphical display (graphicsWidget) part of the _mainWindow which holds the various GUI parts (connection widget (_cWidget), statusBar, viewSettings_w (configuration) and dockable widgets: viewSettings_dw and minimap_dw) and menu items (viewMenu, helpMenu), the _viewSettings is used to hold many properties about the view, such as the grid and viewplan settings for configuration of the graphical display.

The application will automatically try to connect to a Tag/Anchor. If the COM port connection fails the user will be prompted to close the application or try again. If the COM port connection is successful, the application will then open the *Main display window* and finish configuration/initialisation of all of its parts. As part of initialisation the configuration files, if present, will also be loaded.

2.2 Serial/COM Port Connection

The *Serial Connection* class is used to establish and monitor the COM port connection to the Tag/Anchor. When the application is started it will scan all of the COM ports available in the system (PC) and check which ones have "STMicroelectronics Virtual COM Port" description (findSerialDevices()). It will then try and connect to the 1st COM port in the list which matches the "STMicroelectronics Virtual COM Port" description (openConnection()).

For the connection to be successful the TREK1000 Tag/Anchor needs to be already plugged into the PC. (The PC has to have "STMicroelectronics Virtual COM Port" driver installed, so that there is COM port associated with the connected Tag/Anchor.) After opening of the COM port connection the application will send "deca$" string to the Tag/Anchor to find out what is the Tag/Anchor SW version and configuration. If the returned

string matches "nVer. v.st TREK" then the application will assume it has connected successfully.

The *Connection state* can be: Disconnected, Connecting, Connected or Connection Failed.

2.3 RTLS client

If the COM port connection as part of initialisation of *Serial Connection* class was successful the COM port data handler will be assigned to the *RTLS client.*
The *RTLS client* will open a log file (./Logs/yyyyMMdd_hhmmssRTLS_log.txt) and initialise its data structures. Then it will handle any the incoming TOF report messages from the Tag/Anchor via the Serial/COM port connection. The reports are firstly grouped into sets of 4 (max 4 TOF reports – one form each anchor (A0, A1, A2 and A3) for each range sequence (RSN)). Once all the range reports with RSN are received, the trilateration function will be called to try and calculate location of the Tag. If there is a solution, Tag's location estimate will be sent to the the graphical display (graphicsWidget) to update Tag's position information on the application display window.

There is no option to close or open new log files. If new log file is required the application will need to be restarted.

2.3.1 Trilateration Function

The trilateration function (GetLocation()) which is used to estimate Tag's location is based on spherical intersection. It can be called with 3 or 4 range measurements. The user should review the function in trilateration.cpp to familiarise themselves with its operation.

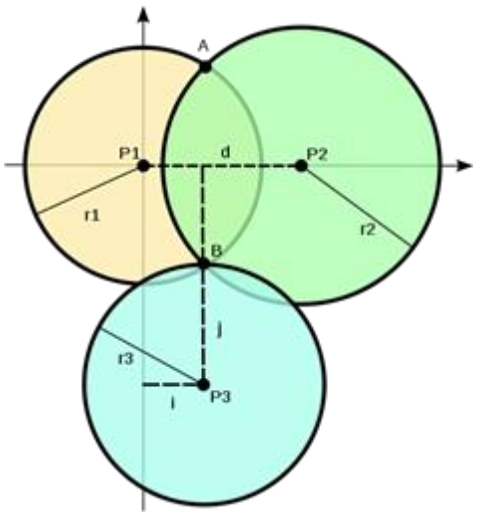


Figure 4: Trilateration function - intersection of 3 spheres

There should be only one solution ideally when three spheres are intersecting. From the figure above, assuming that all of spheres are on the same plane, point B is the only solution which has exactly r1 distance to P1, r2 distance to P2, and r3 distance to P3. However because of range measurement error the intersection B will be given by 2 points equidistant from the plane of the 3 anchors (Ps). The algorithm used in DecaRangeRTLS application will automatically pick the solution which is below the plane. However if there are 4 TOF reports, i.e. if there are 4 anchors present in the system, the solution that is picked is the one closer to anchor 4 (anchor with ID 3).

2.3.2 Range Report Format

The format of ranging report message as sent over the USB port is:
"ma00 t00 range rawrange rangenum rangeseq rangetime txad rxad zZ"

Each ranging report message starts with “m”.

a00	this is the Anchor ID, 0x00 is used for Gateway, 0x01 is A1, 0x02 is A2 and 0x03 is A3.
t00	this is the Tag ID (range is 0x00 to 0x00)
range	this is range in mm, it is bias corrected 32-bit hex number of the calculated range between A and T as reported in the 1st two fields.
rawrange	this is a raw range in mm, 32-bit hex number of the calculated range between A and T as reported in the 1st two fields.
rangenum	this is a 16-bit hex number of the number of ranges since the unit has been powered on
rangeseq	this is the range sequence number (modulo 256)
rangetime	this is the time of receiving a range report at Tag or calculation the ToF at Anchor, units are ms (local Anchor/Tag MCU counter), (32 bit hex number)
txad	the TX antenna delay
rxad	the RX antenna delay
zZ	this is the ID of the Tag or Anchor or Listener the PC terminal is connected to (z = “t” or “a or “l”) (Z = 0, 1, 2, 3)

2.4 Main display window

The *Main display window* is a class that contains the drop down menus, main display area, tag information table and other information and control widgets as described in the sections below:

2.4.1 Graphics

The graphical display (graphicsWidget) part of the _mainWindow which holds the various GUI parts namely these are Tag and Anchor table pane, the display pane. The display pane consists of *Graphics View* which has visible rectangle to keep track of the visible rectangle 2.5.1 and tools 2.6 which operate on it and the *Grid Layout*.

As part of initialisation the graphical display will configure tag settings based on the setting in TREKtag_config.xml configuration file.

2.4.1.1 Tag and Anchor Tables

Tag and Anchor tables are Qt Table widgets, the Tag table contains Tag’s ID, position information and range measurements, as shown in Figure 5.

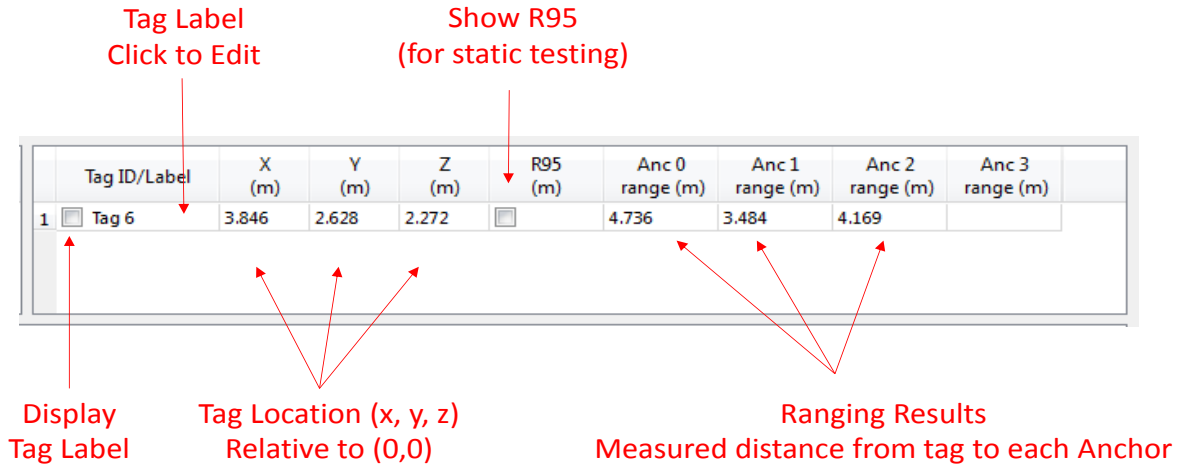
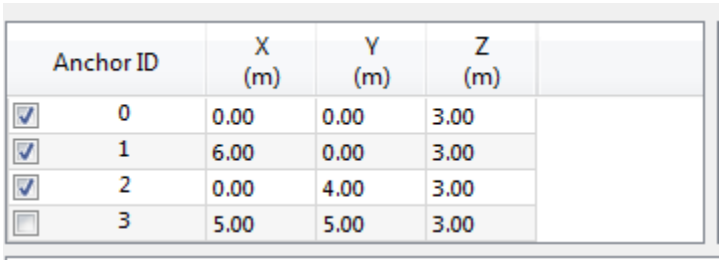


Figure 5: Tag table

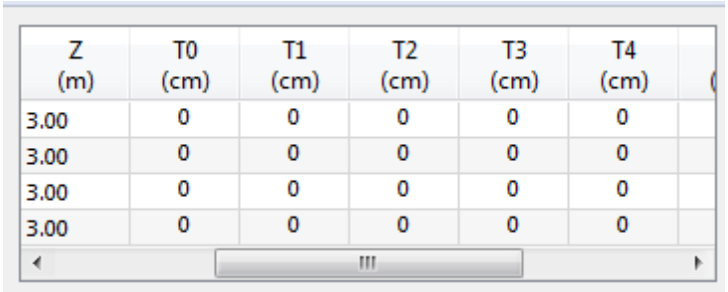
The Anchor table contains Anchor's ID, position information and optional Tag correction table as shown in Figure 6 and Figure 7.



Anchor ID	X (m)	Y (m)	Z (m)
0	0.00	0.00	3.00
1	6.00	0.00	3.00
2	0.00	4.00	3.00
3	5.00	5.00	3.00

Figure 6: Anchor table

The Tag correction table is used to apply correction (in cm) to Tag-Anchor range measurement. These correction values can be used when TREK1000 SW is used on non-TREK calibrated EVB1000s (e.g. on EVB1000 from and EVK1000).



Z (m)	T0 (cm)	T1 (cm)	T2 (cm)	T3 (cm)	T4 (cm)
3.00	0	0	0	0	0
3.00	0	0	0	0	0
3.00	0	0	0	0	0
3.00	0	0	0	0	0

Figure 7: Anchor's tag correction table

2.4.1.2 Display pane

The display pane has a _scene (QGraphicsScene) element, it provides a surface for managing a large number of 2D graphical items. It is the _scene part of *Graphics View* 2.5.1. When the graphical display widget receives data from the *RTLS client* it will:

- Add/update tag's new visual position
- It will remove (gray out) tag's old positions
- Update labels
- Update R95 circle if shown

The graphical display will also change the view from Tracking/Navigational use case to Geo-Fencing use case depending on user TREK use case configuration.

2.4.2 Connection Widget

The *Connection widget* contains the *Serial Connection* (COM port) configuration options. This widget is hidden from the user as the RTLS application connects automatically.

2.4.3 Status Bar

The *Status bar* is used to display status messages/strings:

- "DecaRangeRTLS Anchor/Tag ID Mode" – on COM port open it will display which Tag/Anchor the PC application is connected to.
- "Connected to Anchor/Tag/Listener ID" – on reception of TOF it will display which Tag/Anchor the PC application is connected to.
- "No location solution" – if the trilateration function cannot calculate Tag's position from 5 or more consecutive TOF report sets.

- "Open error" – if the COM port did not open successfully.

2.4.4 View Settings

The *View settings* widget contains 3 Tabs: configuration, floorplan and grid.

2.4.4.1 Configuration tab

The Configuration tag contains the general RTLS application settings for use case modes and showing and hiding of Tag and Anchor tables and Tag history function. Here we can switch between 'Tracking/Navigation' mode and 'Geo-Fencing' mode using the checkboxes.
In Geo-Fencing mode:

- Set 2 zone perimeters on the Display Pane – enter the desired perimeters in metres
- Select 'Alarm Outside' or 'Alarm Inside' depending on whether the no-go area is far from or near to the Anchor

4 User Checkboxes:

- Show / Hide the Tag history
- Show / Hide the Anchor Table
- Show / Hide the Tag Table
- Show / Hide the Anchor-Tag Correction Table

2.4.5 Minimap View

The *Minimap view* widget allows the user to by using the mouse, select different regions of the floorplan to be displayed in the main scene. It is only operational if a floorplan has been loaded.

2.4.6 Drop Down Menus

2.4.6.1 VIEW

View menu contains the following items: -

- ***View Settings*** – selecting this will open a view configuration widget. This allows the user to upload a floor plan and specify the grid, X and Y axis scale and origin positions: -

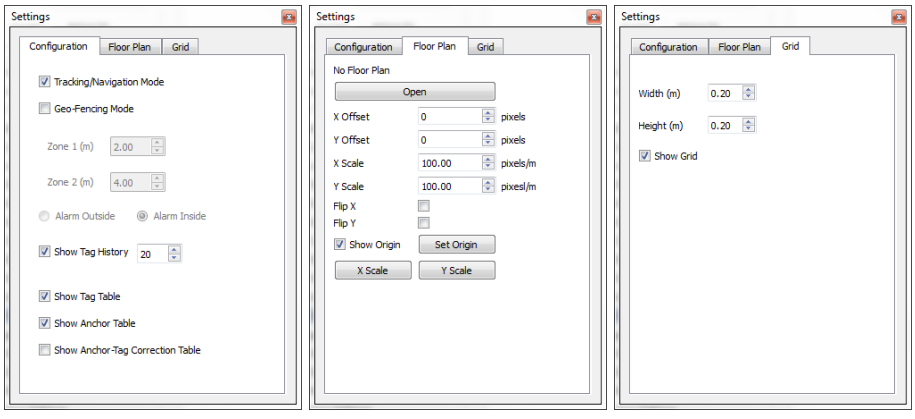


Figure 8: View Settings widget

- *Configuration* tab:

Field Name	Description
Tracking/Navigation Mode	Selects Tracking/Navigation as the RTLS application use case

Field Name	Description
Geo-Fencing Mode	Selects Geo-Fencing as the RTLS application use case
Zone1	Configures Zone 1 radius (only in Geo-Fencing use case)
Zone2	Configures Zone 2 radius (only in Geo-Fencing use case)
Alarm Outside/Inside	Configures if the Alarm is triggered when Tag enters into the minimum zone or leaves the maximum zone. (only in Geo-Fencing use case)
Show Tag History (N)	User can configure if to show the last N Tag positions
Show Tag Table	Shows/hides Tag table
Show Anchor Table	Shows/hides Anchor table
Show Anchor-Tag Correction Table	Shows/hides Anchor-Tag correction table

- *Grid* tab:

Field Name	Description
Width	Sets the horizontal grid spacing, unit is meters
Height	Sets the vertical grid spacing, unit is meters
show	Shows or hides the grid

- *Floor Plan* tab:

Field Name	Description
Open	This lets the user upload an image of the floor plan of the area where the anchors are installed
X offset	This is the origin offset in the X direction from the 0, 0 point on the floorplan (in pixels).
Y offset	This is the origin offset in the Y direction from the 0, 0 point on the floorplan (in pixels)
X scale	This is the scale (pixels per meter) of the x axis (in pixels per meter)
Y scale	This is the scale (pixels per meter) of the y axis (in pixels per meter)
Flip X	This flips the image in the x-axis
Flip Y	This flips the image in the y-axis
show	This shows or hides the origin in the map
Set Origin	This button lets the user click and set the origin coordinates
X Scale button	Pressing this button produces a measuring tool with which the user can firstly select a distance on the map and then enter the actual distance (in meters) that range corresponds to, this sets the *X scale* value
Y Scale button	Pressing this button produces a measuring tool with which the user can firstly select a distance on the map and then enter the actual distance (in meters) that range corresponds to, this sets the *Y scale* value

- ***Minimap*** – selecting this opens a Minimap widget, which shows the loaded image and the zoomed in area (which is shown in the Main Display Area window).

2.4.6.2 HELP

- ***About*** – this opens the "About" window which provides information on the revision of the client.

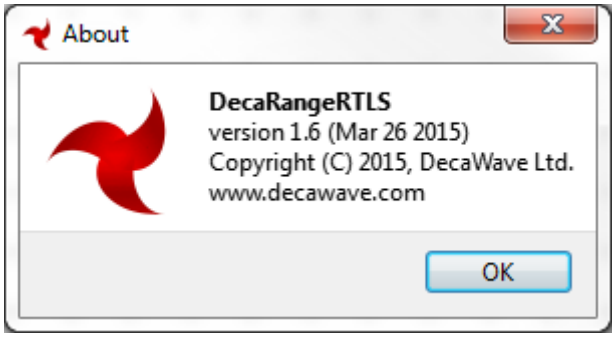


Figure 9: About window

2.5 Widgets and Views

The DecaRangeRTLS application has a number of Widgets and View classes they are used to display the data, and allow user to change/configure the various parameters.

2.5.1 Graphics View

The *Graphics View* class draws the scene and provides user interaction using the mouse. User interaction can be complex as we have to handle many different interaction based on very little mouse events:
- Scene interaction (Selecting anchors, moving them,) - (this is handled by QGraphicsScene and GraphicsDataItem)
- Unselecting all anchors by clicking on the background.
- Zooming using the scroll wheel
- Panning by dragging
- Contextual menu by right-click
- Cancelling the current tool by right-click
- Starting a *Rubber Band Tool* on Ctrl+Drag (2.6.3)
- Tools listening to AbstractTool::clicked() events (2.6.1)
- Tools listening to AbstractTool::mousePressEvent() events (2.6.1)

Most of them are handled by the mousePressEvent()/mouseMoveEvent()/mouseReleaseEvent() cycle. During mousePressEvent(), we find the suited action, and decide of a MouseContext based on that. The MouseContext is kept until mouseReleaseEvent()

The GraphicsView keeps track of the visible rectangle, in scene coordinates. This rectangle will always be visible on the screen. Initially, the visible rectangle is the square from (0, 0) to (1, 1).
The visible rectangle can be transformed using translateView() or scaleView() or changed using setVisibleRect(). Whenever the visible rectangle changes, for any reason, the visibleRectChanged() signal is called.
Tools allow simple interaction inside the scene. A new tool can be set using setTool(). The tool then remains active until it's AbstractTool::done() signal is emitted. When ESC button or right click is pressed, the view attempts to cancel the tool by calling AbstractTool::cancel().

2.5.2 Graphics

The *Graphics* widget is used to display tags and anchors on the main display. It adds tag and anchor items to the scene, and updates the x, y, z co-ordinates as it get new data from the CLE.

The list of tags is also shown, here the user can add tag label, and view multilateration, and blink reception rates as well as turning on/off the R95 indicator.

2.5.3 Minimap

The *Minimap* view shows a zoomed out view of the whole floorplan. It allows the user to quickly select and zoom into a particular place on the floor plan loaded.

2.5.4 View Settings

The *View Settings* widget allows the user to configure the grid size, floorplan settings etc. This is explained in detail in section 2.4.6.1 View Settings.

2.6 Tools and Utilities

2.6.1 Abstract Tool

The Abstract Tool class is a base class any tool implementations should inherit. Tools allow simple, temporary user interaction in the graphics view, based on mouse events. They are used to set the scale (*Scale Tool 2.6.4*) and origin (*Origin Tool 2.6.2*) of the bitmap, and select anchors using a click and drag style rubber band (*Rubber Band Tool 2.6.3*).

Once a tool is enabled, using GraphicsView::setTool(), the cursor changes to the return value of cursor(), the draw() function gets called when drawing the foreground of the scene, and the tool starts receiving mouse events.
The tool can receive the mouse events through two means, pressing and releasing of the left mouse button. The clicked() function gets called when the left button is pressed and released. For more complex interaction, the mousePressEvent()/mouseMoveEvent()/mouseReleaseEvent() functions can be overridden. When a mouse button is pressed, mousePressEvent is called. If it returns true, then tool grabs the mouse interaction, and will receive mouse events until the button is released. Otherwise the scene will be handling mouse events, and mouseMoveEvent()/mouseReleaseEvent() won't be called.

Note that the two mouse event mechanisms are incompatible. If mousePressEvent returns true, clicked() will not be called for this mouse click.

2.6.2 Origin Tool

The *Origin Tool* class is a tool used for setting the floorplan's origin point. It reacts to the clicked() event. When clicked() is called, the origin is calculated based on the click position, and the tool finishes right away.

2.6.3 Rubber Band Tool

The *Rubber Band Tool* class is a tool used to select anchors using a click and drag to draw a selection box. Once the user has started the drag, a rectangular selection box is drawn, and all items within the box get selected. In order to differentiate this tool's click and drag with the one used to move the scene, the tool is started if the Control button is held during the initial mouse press event.
This is handled by the GraphicsView::mousePressEvent(), and is outside the scope of this class.

2.6.4 Scale Tool

The *Scale Tool* class is a tool used to change the floorplan's scale. It allows the user to select two points by waiting for two consecutive clicked() events. Once the user has clicked on two points, the tool shows a popup to enter the distance between the two points. The scale is recalculated and stored.

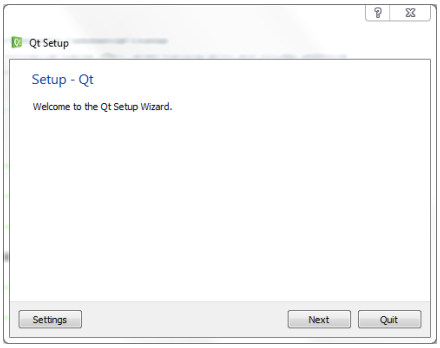
2.6.5 QProperty Model

The *QProperty Model* is a class for turning any QObject-derived subclass with properties into a one-row model. (Copyright 2013 - Harvey Chapman hchapman@3gfp.com Source: https://gist.github.com/sr105/7955969) QPropertyModel creates a single row data model consisting of columns mapping to properties in a QObject. The column list can be retrieved as a QStringList, and a method exists to convert the property names to column numbers.

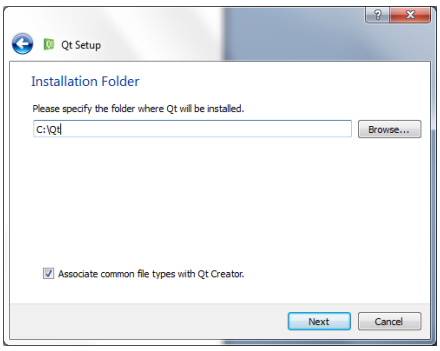
3 QT SDK INSTALLATION GUIDE

The DecaRangeRTLS application is built under the Qt Framework, the released binary is compiled and built with Qt Creator (3.1.2 opensource), based on Qt 5.3.1 (MSVC 2010, 32 bit). The Qt installer for windows can be found online at http://qt-project.org/downloads. Select *Qt Online Installer* for your version of Windows platform:

Once the installer is downloaded, run it and press *Next*:

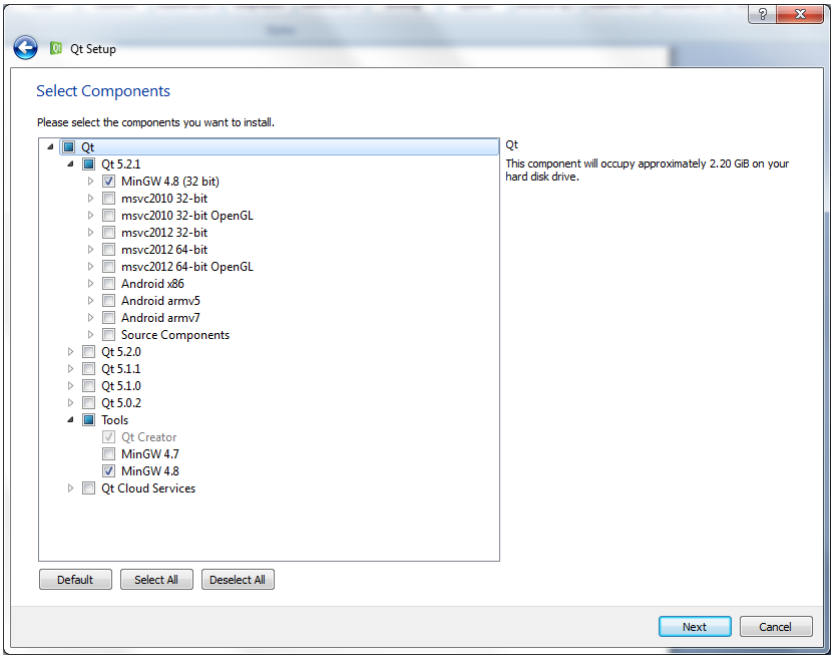


Specify the installation path, and press *Next*:



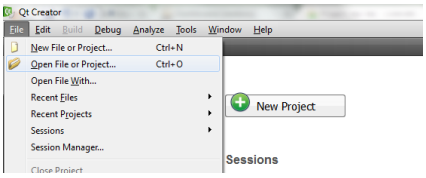
Select the following components, and press *Next* :

- *Qt -> Qt 5.2.1 -> MinGW 4.8*
- *Qt -> Tools -> Qt Creator*
- *Qt -> Tools -> MinGW 4.8*

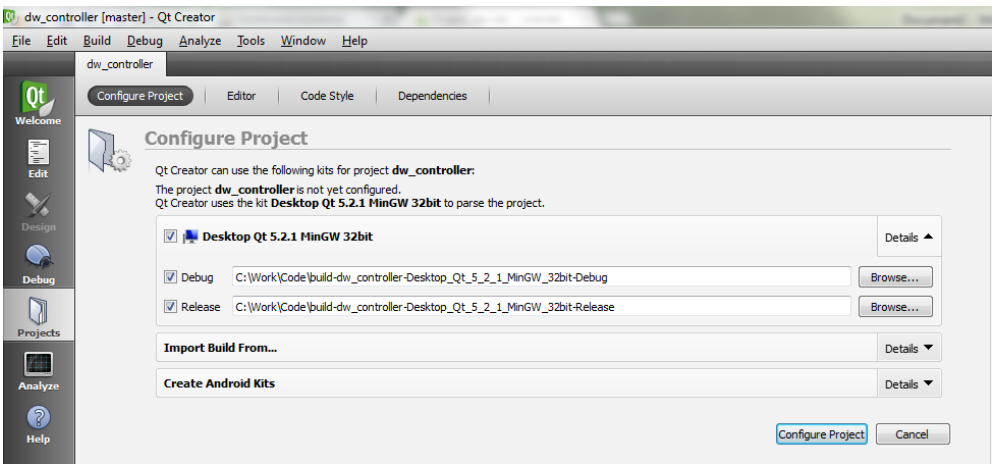


Choose the default options until the end of the wizard.

Run the Qt Creator software. Under the *File* menu, select *Open File or Project*



Select the *.pro* file inside the Project's directory. The first time the project is opened, it must be configured. Leave the default options, and click *Configure Project*



4 APPENDIX A – BIBLIOGRAPHY

Table 2: Table of References

Ref	Title
6	APS003 Real Time Location Systems
7	DecaWave DW1000 Datasheet
8	DecaWave DW1000 User Manual
9	IEEE 802.15.4-2011 or "IEEE Std 802.15.4™-2011" (Revision of IEEE Std 802.15.4-2006). IEEE Standard for Local and metropolitan area networks— Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). IEEE Computer Society Sponsored by the LAN/MAN Standards Committee. Available from http://standards.ieee.org/

5 ABOUT DECAWAVE

DecaWave is a pioneering fabless semiconductor company whose flagship product, the DW1000, is a complete, single chip CMOS Ultra-Wideband IC based on the IEEE 802.15.4 standard UWB PHY. This device is the first in a family of parts.

The resulting silicon has a wide range of standards-based applications for both Real Time Location Systems (RTLS) and Ultra Low Power Wireless Transceivers in areas as diverse as manufacturing, healthcare, lighting, security, transport, and inventory and supply-chain management.

For further information on this or any other DecaWave product contact a sales representative as follows:

DecaWave Ltd,
Adelaide Chambers,
Peter Street,
Dublin 8,
Ireland.

mailto:sales@decawave.com
http://www.decawave.com/