TTK1000 – TDOA RTLS SOFTWARE

Product Overview

Decawave's TTK1000 simplifies and accelerates product development by providing a software source code suite for a TDoA RTLS based on Decawave's DW1000 IEEE 802.15.4 UWB wireless transceiver IC. This software allows developers to easily acquire the base RTLS functionality, kick-starting their RTLS product development.

Key Features

- Software source code for a TDoA real-time location system (RTLS)
- Time difference of arrival (TDoA) location
- Wireless time synchronisation
- Based on Decawave's DW1000 ultra-wideband (UWB) RF transceiver IC
- IEEE802.15.4-2011 UWB compliant RF and ISO 24730-62 compliant tag blink messages
- Provides source code for tags (mobile nodes), anchors (fixed infra-structure), central location engine (CLE) position solver and example upper layer client applications
- The CLE API delivers tag ID and tag location to client applications. This allows connection of customers upper layer client to deliver the RTLS for the target application

Key Benefits

- Tag and anchor source code readily portable to customers own DW1000 based designs to suit their target application needs.
- Functional location engine incorporating wireless anchor clock synchronisation and tag multilateration and optional result filtering.
- Kick-starts the development of your RTLS system

Applications

- Precision real time location in a variety of markets:
  - Healthcare
  - Consumer
  - Industrial
  - Other
- Real-time location of valuable assets or people

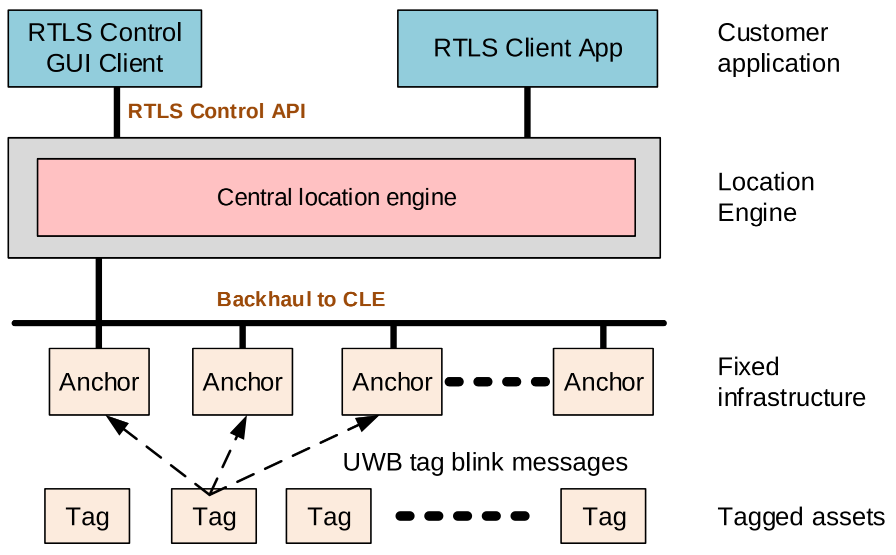

Figure 1: High Level System Architecture Diagram

Table of Contents

DOCUMENT INFORMATION

Disclaimer

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.

REGULATORY APPROVALS

The DW1000, as supplied from Decawave, has not been certified for use in any particular geographic region by the appropriate regulatory body governing radio emissions in that region although it is capable of such certification depending on the region and the manner in which it is used.

All products developed by the user incorporating the DW1000 must be approved by the relevant authority governing radio emissions in any given jurisdiction prior to the marketing or sale of such products in that jurisdiction and user bears all responsibility for obtaining such approval as needed from the appropriate authorities.

SOFTWARE LICENSES

The TTK1000 software is available only under license from Decawave. If you have not signed such a license, you may not use the TTK1000 software. Please consult your Decawave representative.

The TTK1000 software as delivered includes or refers to source code provided by third parties (see section 3.6 in this data sheet which lists the third party software used or referred to in the TTK1000 software). Before using this third party software, you should familiarize yourself with and agree to the terms of the respective licenses governing the use and distribution of such software.

1 SYSTEM OVERVIEW

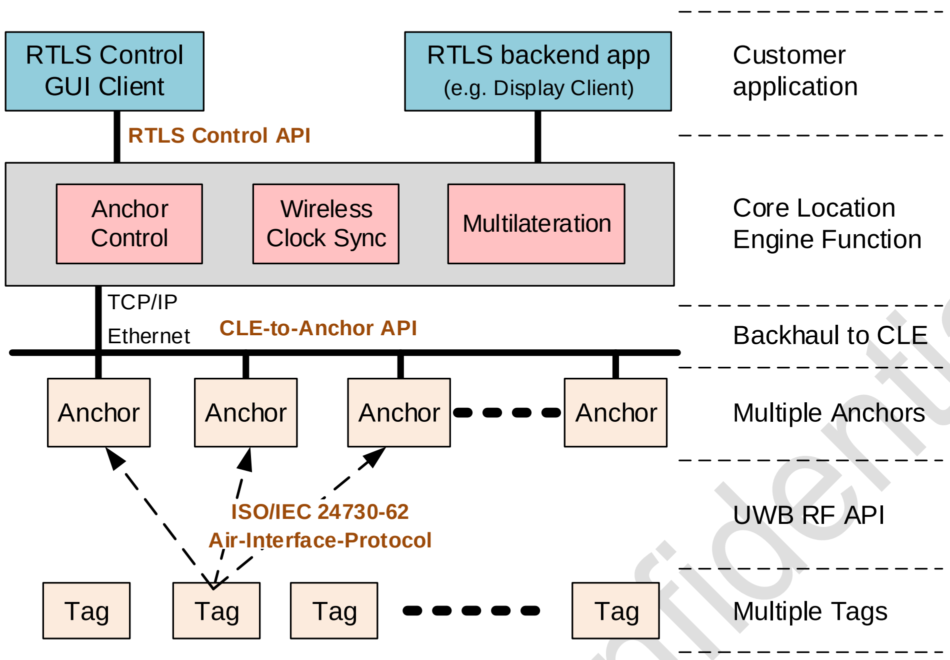

Figure 2: System Architecture Overview

In TDOA RTLS, tags send periodic *blink* messages that are received at the fixed infrastructure (anchor) nodes. The anchors report the blink reception time-of-arrival (TOA) timestamps to the central location engine (CLE), and the CLE *multilaterates* to estimate the tag location.

In the TTK1000 the CLE reports the tag ID and its estimated location through a defined API to the upper layer backend application.

Customers of the TTK1000 will in general add their own application specific functionality on top of this API. As an example, a display client application is included in the TTK1000, with its source code. This may be replaced or used as the starting point for the customer's application development.

An RTLS control client application is also included in the TTK1000 software suite. This provides the mechanisms to configure the RTLS system, e.g. to set the physical coordinates of the fixed anchor nodes, relative to which the tags are located.

The RTLS control client application may be used as supplied, but the source code is provided enabling customers to add their own functionality to customise it to their own target application.

This TTK1000 software suite provides the base functionality of a TDOA RTLS system. This facilitates a speedy time-to-market of customers' DW1000 based RTLS products.

The main components of the TTK1000 are:

- Source code for the tag. The tag software controls the DW1000 transceiver to transmit a blink message periodically.
- Source code for the anchor. Anchors receive the tag blink messages and report the tag ID and blink message RX timestamps to the CLE. For inter-anchor time synchronisation anchors also send and receive clock calibration packets (CCP) and report CCP TX and RX timestamps to the CLE.
- Source code for the central location engine (CLE) incorporating both TDOA multilateration and Decawave's wireless synchronisation scheme for the anchors' time-base and some basic data filtering algorithms.
- Source code for the "Control client" GUI used to perform system configuration and report and log diagnostics.
- Source code for the "Display client" GUI application. This is an example an RTLS backend application.
- Documentation package including: Software architecture, guides to each component, and API documents.

2 SYSTEM ARCHITECTURE

2.1 *Architectural overview*

The main work of the RTLS is solving the blink message TDOA data to get an estimate of the location of the tag that sent the blink message. Figure 3 below gives a more detailed view of the system architecture concentrating on the central location engine (CLE) that delivers the core RTLS functionality.

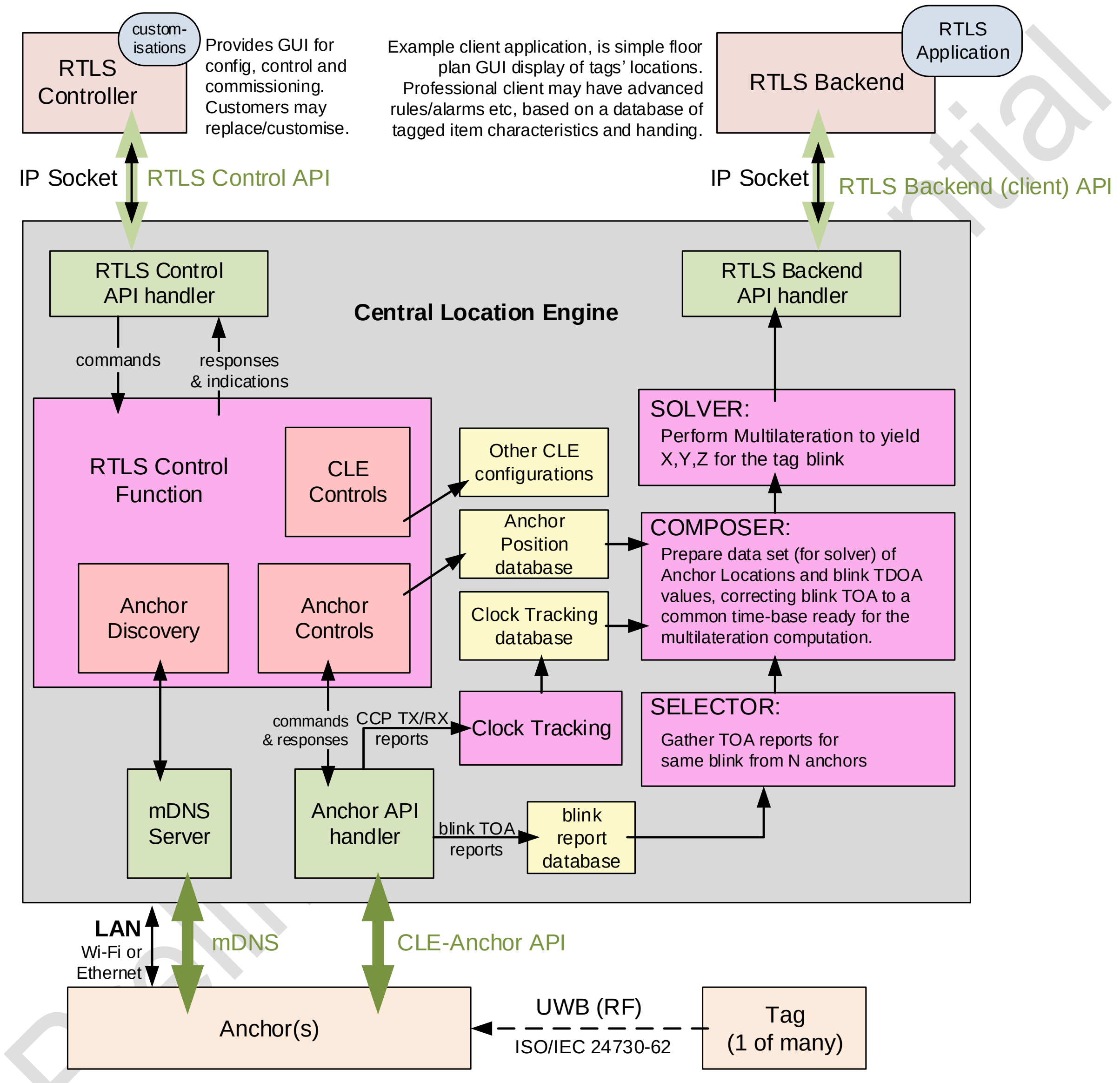
RTLS Controller
custom-isations
Provides GUI for config, control and commissioning. Customers may replace/customise.
Example client application, is simple floor plan GUI display of tags' locations. Professional client may have advanced rules/alarms etc, based on a database of tagged item characteristics and handing.
RTLS Backend
RTLS Application
IP Socket
RTLS Control API
IP Socket
RTLS Backend (client) API
RTLS Control API handler
Central Location Engine
RTLS Backend API handler
commands
responses & indications
RTLS Control Function
CLE Controls
Anchor Discovery
Anchor Controls
Other CLE configurations
Anchor Position database
Clock Tracking database
SOLVER:
Perform Multilateration to yield X,Y,Z for the tag blink
COMPOSER:
Prepare data set (for solver) of Anchor Locations and blink TDOA values, correcting blink TOA to a common time-base ready for the multilateration computation.
SELECTOR:
Gather TOA reports for same blink from N anchors
commands & responses
CCP TX/RX reports
Clock Tracking
mDNS Server
Anchor API handler
blink TOA reports
blink report database
LAN
Wi-Fi or Ethernet
mDNS
CLE-Anchor API
Anchor(s)
UWB (RF)
ISO/IEC 24730-62
Tag (1 of many)

Figure 3: Detailed system architecture

For successful TDOA multilateration all tag blink RX timestamps need to have a common concept of time. The TTK1000 includes Decawave's wireless synchronisation scheme, which employs UWB messages transmitted between anchors to track the anchors' relative clock drift and offset and convert the tag blink timestamp values into a common time-base for TDOA multilateration. For time synchronisation, anchors send (and receive) periodic CCP messages and report their TX and RX timestamps to the CLE. The CLE incorporates a clock tracking algorithm that uses these CCP timestamps to track the relative clock differences between the anchors. This clock tracking information is used to correct the tag blink timestamps from multiple anchors to a single common time base for TDOA multilateration.

Clock synchronisation may also be achieved via wired clock distribution, with appropriate hardware design. The TTK1000 has rudimentary support for this, incorporating a tag training mode where the fixed clock offset between wired anchors can be determined by averaging the TDOA of blinks from a fixed reference tag of known location.

2.2 *Typical system deployment*

Figure 4 shows a minimal system deployment. Four anchors connected via Ethernet LAN to the PC running the CLE. One or more tags blink periodically. In the figure tag T2 is shown sending a blink message that is received by the four anchors, labelled A1, A2, A3 and A4.

To configure the RTLS the control client application is run. The RTLS control client connects to the CLE via a TCP/IP socket interface. The RTLS control client application can run on the same PC as the CLE or on another PC. The control client is used to configure the anchor locations, select which anchors (e.g. anchor A2 in the figure), act as *master* anchors for clock synchronisation by sending the periodic CCP frames (used to track the anchors' relative clock drift), and start the RTLS running.

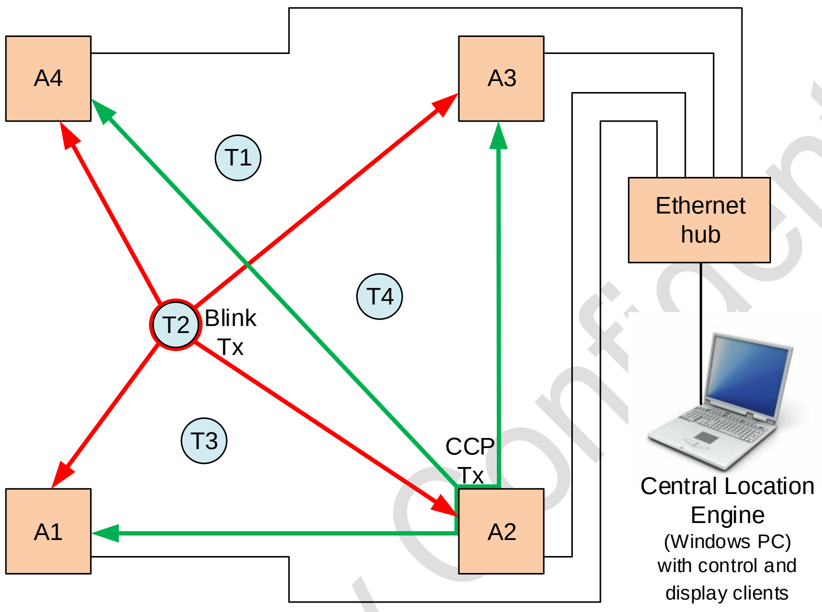

Figure 4: Minimum system deployment

Where it is required to deploy the RTLS over a larger area beyond the range of the minimum system, additional anchors can be added as necessary. The TTK1000 documentation package includes the *TTK1000 RTLS System Installation Guide* document which describes system setup with multiple anchors.

To display the resultant tag location(s) the display client application is run. This also connects to the CLE via an IP socket interface, and can be on the same PC as the CLE, or on another machine on the same network. The CLE supports multiple connections of "display" client applications. The RTLS results will be sent to all such connected applications.

As mentioned previously it is expected that customers may replace the "display" client application with their own client RTLS application targeted to their own specific application use case. For example, a typical RTLS application will associate the tag ID (i.e. the tagged item) with specific entities in a "managed items" database, and may log the item's position in real-time for records, and/or use zone information or other rules associated with the item to manage it and/or raise alarms according to the needs of the application.

3 SYSTEM COMPONENTS SOURCE CODE OVERVIEW

3.1 *Tag C source code*

The tag software transmits periodic blink messages and puts the DW1000 and on-board microcontroller into low-power sleep mode between blinks. The tag software is a bare metal implementation without any OS. The sleep period between blinks is randomised (see section 4.1) to avoid continuous collisions (e.g. so no two tags stay in lock-step transmitting at the same time). The tag code is delivered to run on an ARM Cortex M3 based tag hardware (HW) design. The tag source code is readily portable to other processors.

3.2 *Anchor C source code*

The anchor software runs under the FreeRTOS real time operating system, on an ARM Cortex M4 based anchor HW design. The open source LWIP TCP/IP stack is employed to link each anchor to the CLE via an Ethernet LAN connection. Anchor configuration, control, and timestamp reporting is achieved via this TCP/IP backhaul connection to the CLE. The DHCP client protocol is employed for assigning the anchor's IP address. The mDNS protocol is employed for discovery of anchors by the CLE.

For wireless clock sync, designated *Master* anchors transmit Clock Calibration Packets (CCP) and report CCP TX timestamps to the CLE. All anchors receive CCP and tag blink frames and report the CCP RX timestamps and the blink RX timestamps to the CLE.

To aid installation anchor-to-anchor two-way ranging is supported.
A target specific Ethernet TCP/IP boot loader is included for easy anchor firmware upgrade during development.

3.3 *Central Location Engine (CLE) C/C++ source code*

The CLE is an MS Visual Studio C++ Windows based application. While MS Windows is the delivery platform, the CLE is readily portable to compatible operating systems (e.g. Linux).

Windows C Runtime Library and Windows Sockets API implement commonly used POSIX API functions for file, time, environment, and socket access, although the support remains largely incomplete and not fully interoperable with POSIX-compliant implementations.

The main function of the CLE is to perform the TDOA multilateration using the tag blink TOA reports from the anchors to estimate each tag's location. The CLE also incorporates the wireless clock synchronisation function which tracks the clock drift between anchors (using reported CCP TX and RX timestamps), and, then converts Blink RX timestamps to a common time base to yield TDOA values for multilateration.

The CLE has a TCP/IP socket based API to its upper layer client applications. The CLE has a number of filtering functions that can be enabled from the control client.

3.4 *Control Client C++ source code*

The control client software is based on the Qt Framework for Windows PC, enabling easy porting to other platforms supported by Qt.

The control client connects to the CLE via a TCP/IP based socket interface and employs a Decawave defined Control Client API to drive the RTLS. The control client is used to configure the system, including setting the UWB mode, specifying the anchors' positions, selecting which anchors are the wireless clock sync master anchors (i.e. sending CCP). In systems with more than one master anchor, one master is designated as the main master, while additional masters are set as secondary masters, configured to transmit their CCP synchronised (with a specified lag time) to the receipt of a CCP from a master they are configured to follow.

3.5 *Display Client C++ source code*

The display client software is based on the Qt Framework for Windows, enabling easy porting to other platforms supported by Qt. The display client connects to the CLE via a TCP/IP based socket interface and consumes the location reports output by the CLE RTLS, each consisting of a tag ID and its X, Y, Z location estimation. As this represents the application layer, customers will typically replace the display client and use this interface API to provide their target application specific functionality. The delivered display client operates to show the tag location results on a floor-plan graphic.

3.6 3rd Party IP requirements

The main mathematical operations of the CLE employ the Armadillo C++ linear algebra library, distributed under the terms of the Mozilla Public License 2.0 (MPL) [6].

The tinyXML (libxml2) library is used to parse the xml based CLE configuration file, and the command and data interfaces to the Control and Display client applications. This library is distributed under the MIT license [7].

The anchor employs the lwIP TCP/IP stack and FreeRTOS.

4 TTK1000 COMPONENT SPECIFICATIONS

4.1 *Tag source code*

The tag is the transmit-only application and supports the UWB physical layer modes configurations of the DW1000, as described in 5.3.

The delivery platform hardware in conjunction with target specific tag software allows for the tag's UWB operating mode and blink rate to be configurable via its UART interface. In a practical system deployment the tag probably will not have a UART interface for configuration, and would be hard coded by the system designer to operate in one fixed mode.

The UWB mode configuration of the tag should be set depending on the desired use case (e.g. high density of tags or long range), can be reconfigured via USB-COM port using Controller application. With respect to recommended air utilization rate of 15%, the recommended number of tags, blinking at 1 Hz rate is in the table below.

Table 1: Recommended number of tags, blinking once per second

UWB configuration operating mode	Duration of 12-octet tag blink frame	Recommended number of tags in the area with 1 Hz blink rate at recommended 15% air utilization
6.81 Mbps, PSR 64, PRF 16 MHz	111.54 µs	1344
6.81 Mbps, PSR 128, PRF 16 / 64 MHz *	175.13 µs	856
850 kbps, PSR 512, PRF 64 MHz	706.54 µs	212
110 kbps, PSR 1024, PRF 64 MHz	2.46 ms	61

*- default UWB configuration is 6.81 Mbps, PSR 128, PRF 64 MHz

The default blink rate of the tag is 10 blinks per second. This is used when the tag is moving and defined by a pause delay of 100 ms between blinks. A slower blink rate such as 1 blink per second (with the pause of 1,000 ms) or once every 10 seconds (with the pause of 10,000 ms) may be applicable in many applications. When the tag is stationary, the default pause between blinks is 5 sec and is not configurable.

To prevent any two tags overlapping transmissions for extended periods of time, a randomization of the pause delay between blinks is applied, which is also configurable via the Terminal application. The randomization level can be varied between 1% and 50% of the blink pause duration in steps of 1%. The default blinking randomization is 30%, which means that the value of default blink delay will be between 70 and 130ms. The distribution of the standard pseudo-randomization function in tag is almost uniform.

The TTK1000 tag source code as supplied is compiled using ***gnu-arm-none-eabi-gcc*** **compiler v.7.2.1 with memory optimization option enabled.** The TTK1000 tag HW delivery platform has an nRF52832 MCU, which has 512 KB of flash memory and 64 KB of RAM. The binary image for tag occupies 25 KB of flash memory and 4.8 KB of RAM. The memory usage requirements depend on the platform, functionality, compiler and the optimization used.

Table 2 shows the memory usage of different sub components of the tag application. The core functionality uses less than 16 KB of flash memory and 2 KB of RAM. The size can be optimized further if needed.

Table 2: Tag application memory usage

Tag component	Program code	Constant Data	Static SRAM	Notes
DW1000 low-level driver	5.1 KB	-	0.1 KB	Core functionality
Main code tag blink	1.5 KB	-	0.2 KB	Core functionality, not including stack/heap
Target specific SPI	0.9 KB	-	0	Core functionality

Tag component	Program code	Constant Data	Static SRAM	Notes
CMD UART Interface	2.3 KB	1.5KB	0.3 KB	Note 1
Libraries/other	11.9 KB	1.2KB	2.6 KB	Note 1
Stack	-	-	2 KB	

Note 1: UART functionality is used for tag configuration.

4.2 *Anchor source code*

The anchor is the main component of infrastructure for the RTLS system. Anchors are the fixed reference points receiving tag blinks and reporting to the Central Location Engine (CLE). A minimum of four anchors is needed to perform location estimation. The anchors can be configured to any DW1000 UWB modes of operation as described in [3], **Table 1** lists commonly used configuration modes.

In the TTK1000 RTLS system, the anchor communicates with the Central Location Engine (CLE) via Ethernet network. On a power up, the anchors listen for network commands from the CLE. On reception of the "START RTLS" command, the anchors in the RTLS begin operation as per their configuration in either "master" or "slave" roles and start sending UWB frame reports to the CLE (CCP TX, CCP RX and Blink RX reports). The master anchors send periodic CCP frames for wireless clock synchronization algorithm. The transmit period of master's clock calibration packets is 150 ms.

The anchor reports the tag blink RX timestamp and tag ID (blink message source address) to the CLE via the TCP/IP Ethernet *backhaul* link. A diagnostics capability is a part of the anchor firmware, the diagnostic modes are described in 5.2.3.3. Enabling the diagnostic reporting reduces the system capacity, see the section 4.2.1.

The delivery platform for the anchor software is an STM32F429ZE/I hardware platform. This is an ARM CORTEX M4F MCU. The "E" version of MCU has 512KB and "I" version has 2MB of flash memory and 256 KB of RAM working on 144 MHz clock speed and has an Ethernet MAC peripheral block.

The TTK1000 anchor source code is compiled using ***gnu-arm-none-eabi-gcc*** **compiler v.5.4.1** with speed optimization option enabled to –O2.
The final anchor's target image consists of three independently built components, they are described in the **Table 3**.

Table 3: The anchor MCU memory split for target image components

System component	Flash	Const. Data	Static SRAM	Section start	Available memory	Description
Bootloader	2.6 KB	16 KB	1.6 KB	0x08000000	32 KB Flash / 192+64 KB RAM. 16 KB of constant data reserved in this section for sharing between all components	A small bare-metal application. The main purpose of Bootloader is to start the execution of either the Network bootloader application or the Anchor application
Network bootloader	92 KB	0.6 KB	175 KB	0x08008000	224 KB Flash / 192+64 KB RAM +16 KB of shared constant data from Bootloader section	A utility application that allows for reprogramming of the main anchor application firmware via the Ethernet interface
Anchor application	152 KB	1 KB	184 KB	0x08040000	256 KB Flash / 192+64 KB RAM +16 KB of shared constant data from Bootloader section	The main TTK1000 RTLS infrastructure component delivering the anchor functionality.

The Network Bootloader and the Anchor application run using FreeRTOS. The FreeRTOS operating system BSP provides the HAL (hardware abstraction layer), the lwIP (lightweight TCP/IP stack) and the RTOS tasks management functionalities. The Anchor application memory usage is in **Table 4** below.

Table 4: Anchor application memory usage

Anchor application functional component	Program code	Constant Data	Static SRAM	Notes
DW1000 low-level driver	7.5 KB		0.1 KB	Started from section 2 - 0x08008000
Main code	50 KB		68 KB	
Target specific SPI	1.2 KB		0.1 KB	
FreeRTOS BSP (RTOS, LWIP, USB)	70 KB		107 KB	
Other/ ext. sensors support	-	-	-	
TOTAL	152 KB	16 KB	184 KB	From 256 KB FLASH / 192+64 KB RAM

4.2.1 Anchor Ethernet backhaul throughput

The TTK1000 anchor and the CLE use the TCP/IP socket interface for their backhaul communications. The anchor employs the lwIP TCP/IP stack and FreeRTOS. To support the TCP/IP stack the TTK1000 anchor software reserves approximately 42 KB of RAM for network buffer operations (30 KB in the main application and 12 KB in the lwIP stack).

The TTK1000 anchor implementation is able to transport up to one megabit per second of data to the CLE. This data transfer throughput limitation is lwIP stack and network dependent, for more information refer to [5]. With a minimum blink report message of 18 bytes, this translates to a throughput of approximately 6000 blink reports per second without diagnostic information or about 1300 blinks per second with UWB diagnostics included in the blink reports, please see **Table 5** below.

Table 5: Backhaul capacity for different operation modes

TTK1000 operation mode	Report length	Backhaul report capacity in reports per second
No diagnostics	18	6,000
With diagnostics	84	1,300

Note: The diagnostic reporting mode of operation is intended for investigation of any issues and general debug during system development and trialling. It is not recommended for general operation or deployment because it reduces system capacity and generates a huge amount of data.

4.2.2 The anchor's UWB handling capacity

The advantage of TDOA is its ability to locate a large number of tags, which is due to the fact that a single message from a tag is sufficient for its location estimation. The main system capacity limitation is the physical 100% air-time utilisation, as shown in **Table 6**.

The maximum anchor's over the air blink handling capacity, in the TTK1000, relates to the interrupt processing latency, SPI clock rate, MCU speed and the time it takes to prepare the DW1000 to receive the next frame. A more detailed description can be found in [5].

In the default mode (6.81 Mbps, PSR 128), the over-the-air handling capacity is 3388 blinks per second which is well in excess of both the recommended air utilization of 856 blinks per second and the backhaul throughput capacity of 6000 blink reports per second.

Table 6: Blink handling capacity of the anchor, blinks per second

Operating mode	Duration of 12-octet tag blink frame	Number of blinks at recommended 15% air utilization	Anchor blink handling capacity
6.81 Mbps, PSR 64, PRF 16 MHz	111.54 µs	1344	4318
6.81 Mbps, PSR 128, PRF 16 MHz	175.13 µs	856	3388
850 kbps, PSR 512, PRF 64 MHz	706.54 µs	212	1296
110 kbps, PSR 1024, PRF 64 MHz	2.46 ms	61	467

4.3 *Central location engine (CLE)*

The central location engine (CLE) is the main computational component in the TTK1000 RTLS software. The CLE uses anchor CCP TX and RX timestamp reports to track the relative drift of the anchors' clocks and correct the anchors' tag blink RX timestamp reports to a common time-base. The CLE then performs the TDOA multilateration to give an estimate of each tag's location each time it blinks (assuming that the blink is received and reported by 4 or more anchors). The CLE is also responsible for the general management of the anchor infrastructure. The CLE provides connections to an external RTLS Control client for the control and configuration of the RTLS and to an RTLS Display client for delivery of the RTLS results.

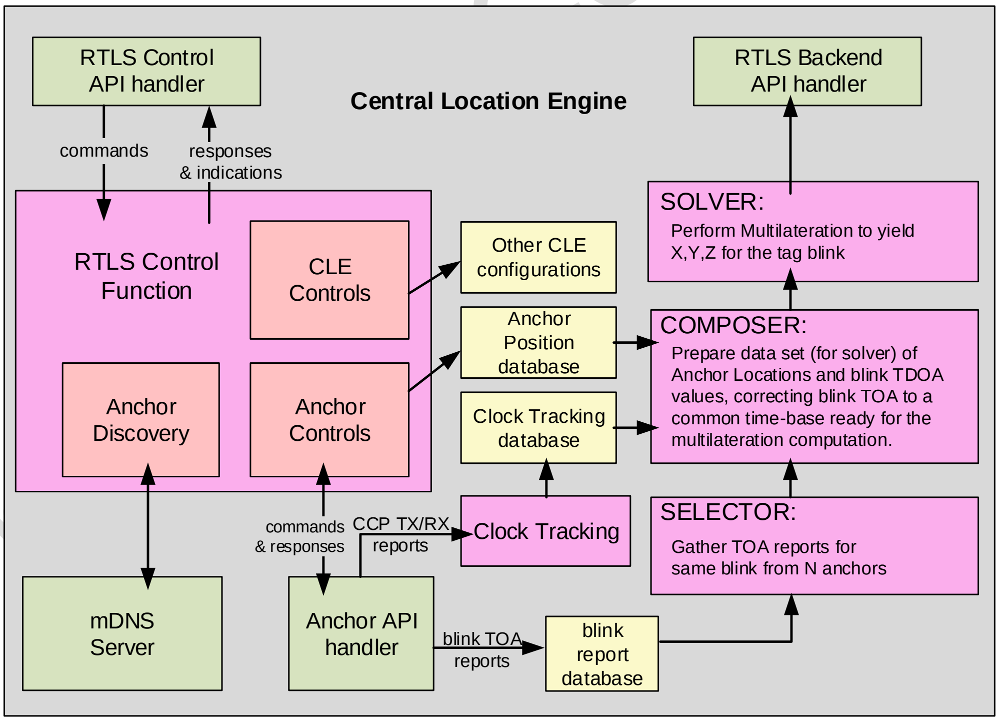

Figure 5: The CLE's main blocks

4.3.1 CLE configuration

When the CLE is first launched its reads the configuration from configuration file (dw_cle.xml) that accompanies the CLE executable. The configuration parameters found in the file are described in **Table 7.**

Table 7: The CLE start-time configuration parameters

Parameter	Default value	Description of the configuration parameter
controlPort	3334	The socket's port number to listen to the incoming Control client connection and commands
dataPort	3335	The socket's port number to listen to the incoming Data client connection. Later the CLE will report any tag location estimates on this port.
logLevel	4	The default logging level configuration. This can be changed during the run-time. For description of the diagnostics logging levels, see 5.2.3.3
wirelessSync	1	This enables the use of the wireless clock synchronization algorithm. If the parameter is 0, then wired clock algorithm will be used instead, see 4.3.2.2
solveTimeWait	50	The expiration time in milliseconds, between reception of the first report of a blink from any anchor and performing multilateration, see 4.3.2.4
solver3D	0	The spherical intersection configuration option specifies whether to use a 3D or 2D solver, see 5.2.3.1
ZERO	1	Enabling of a Zero'th anchor algorithm selection, described in UM.
FPIRF	1	Enabling of a First Path Index Rejection Filter, described in UM.

To complete the configuration and initialization of the RTLS, the Controller application has to be run to upload a run-time configuration and issue the START RTLS command. The run-time configuration supplies the list of anchors that should be included in the RTLS network, the physical geometry of the system, the anchor configuration including UWB operating mode and master/slave configurations as described in the Control application configuration.

4.3.2 CLE components and resource usage

4.3.2.1 CLE anchor connection handler and memory usage

Since we use a TCP/IP sockets for anchor connections and the maximum number of opened sockets in the Windows OS is limited to 65535, this is the ultimate limit of the maximum number of anchor connections supported by the CLE. In addition, the CLE software currently has a limitation of the number of served anchors (anchors that are part of RTLS constellation) to 255 where a single byte is used to identify the anchor number. Please refer to the description given in [5] on how to remove this limitation.

The anchor memory structure allocated for each anchor is 10 KB (9928 bytes). The socket server to the anchor consumes an additional 13 KB (12812 bytes), which means each connected anchor requires 23 KB of operational system memory in the CLE. This scales linearly with the number of anchors in the system.

4.3.2.2 CLE clock synchronization of anchors

The CLE wireless clock tracking can handle up to 16 masters tracked for each slave anchor. In a normal installation the number of tracked masters will typically be in the range of 1 to 4. The memory needed for this is included in the anchor memory structure described above.

4.3.2.3 CLE resource usage for each tag

The permanent size of the data structure needed for each unique tag is 8.5 KB (8336 bytes).

4.3.2.4 CLE handling of tag blinks

The temporary size of memory needed to handle a tag blink, (including up to 16 separate anchors' RX timestamp reports), is 2328 bytes of RAM. This temporary memory persists for a period, which represents the time from its allocation (when the first anchor's RX timestamp report arrives) to its deallocation after the multilateration solver is called. This period is set by the *solveTimeWait* configuration (see **Table 7**) which is 50 ms by default. It needs to be long enough in order to receive all blink reports from anchors before multilateration is attempted. It may need to be increased if poor network latency causes some of the blink reports to continually arrive later than this.

To assess the memory load of this we could for example, imagine there are 5000 separate tag blinks arriving at the CLE per second (this is more than the recommended air-utilisation, but could perhaps represent a population of tags spread over a wider campus area). Assuming the 5000 blinks are uniformly spread in time, then an average of 250 blinks would need to be processed in any 50 ms period, so the memory requirements would be 250 x 2328 bytes, or approximately 570 KB RAM.

4.3.2.5 CLE handling of control client

Only one control client application is allowed to connect to the CLE at any one time. The control client application TCP/IP socket connection requires 12.5 KB of RAM.

4.3.2.6 CLE handling of display clients

The CLE allows connection of up to 50 display clients. Each display client application TCP/IP socket connection requires 12.5 KB of RAM.

Table 8: CLE memory usage in kilobytes

CLE component	Program code
TOTAL CLE application size	464 KB

4.3.2.7 CLE performance

The CLE has been optimised for multicore CPU usage. The CLE has been tested on Windows PCs with Intel Core i5 (8 cores desktop and 4 cores laptop) and Intel Core Duo (2 cores old desktop) processors. We have not experienced any difficulties relating to processing and memory load on those platforms when handling up to 1000 blinks per second in our test environment with either 4 or 8 anchors. Customers wanting to port the CLE to other environments may need to look more closely at the CLE functionality to optimise it. This assumes the GUI display client is running on different PC.

4.4 *Control and Display client applications*

The Control and Display client applications are created using Qt cross-platform application development framework. Qt supported platforms include Linux, OS X, Windows, VxWorks, QNX, Android, iOS, and others. The Display and Control applications compiled using the shared Qt library model. The memory usage can be found in **Table 9.**

The Display client and display part of control application were tested with a set of 500 tags* blinking twice a second each. They can both successfully handle 100% of the incoming tag location estimation data (1000 locations per second).

Table 9: Control and Display applications memory usage in kilobytes

CLE component	Program code
The Control application size	980 KB
The Display application size	670 KB
The shared QT libraries size	~45 MB

* One tag blinking sequentially with a changing of a tag-ID every 1 ms with a loop period of half a second simulates blinks from 500 time-slotted tags.

5 TTK1000 DETAILED SPECIFICATION

In this section, the operation states, the physical channel model and the frame formats are described.

5.1 *Operational states of the TTK1000*

The TTK1000 system has a number of basic operating states as described in **Table 10**. The operational state of the system is defined with respect to the mode of operation of each sub component: anchor, tag, CLE, Control and Display client applications.

Table 10: Operating States of the system

State	Description
IDLE	The system is waiting for a command from a user
RTLS	The main state of operation, every component in the system is executing its RTLS role
TWR	The state when anchors are performing two way ranging
COMMUNICATION Test	The state used to measure the quality of communication between two anchors
POWER Test	The state used to measure the transmit power of an anchor

5.1.1 IDLE state

This is the state when every component of the system is awaiting to serve the user request, see **Table 11**.

Table 11: Operation of the individual system components in the IDLE state

System component	Operation description
Controller	Is connected to the CLE, awaiting the commands from the user
CLE	Is connected to anchors, awaiting any commands from controller
Displayer	Is connected to the CLE, awaiting of any RTLS data to display from the CLE
Anchor	Is connected to the CLE, awaiting commands from the network
Tag	Not applicable, awaiting to be switched on

5.1.2 RTLS state

This is the main state of operation, where every component in the system is executing its RTLS role, see **Table 12.** Here we are assuming that the user has configured the system (anchor coordinates, role, etc.) and has started RTLS operation.

Table 12: Operation of the individual system components in the RTLS state

System component	Operation description
Controller	Is connected to the CLE, the “Configure” and “START RTLS” commands have already been sent to CLE to put other sub components into RTLS state. Waiting for new commands from the user or status reports from the CLE
CLE	Is connected to the anchors, the “Configure” and “START RTLS” commands have been sent to anchors. Processing incoming data from Anchors, solving RTLS tasks and reporting to Display clients the location estimation results. Also waiting for any commands from the Controller

System component	Operation description
Displayer	Is connected to the CLE, displaying tags locations estimations from the CLE
Anchor	Is connected to the CLE, “Configure” and “START RTLS” commands have been received from the CLE. Receiving UWB Blinks from tags and CCPs from any master anchors and reporting the timestamps to the CLE. The master anchors also periodically transmit CCP
Tag	Has been switched on and is blinking periodically

5.1.3 TWR state

This is the special state of operation, in which the selected anchors perform the two-way ranging between them, see **Table 13**.

Table 13: Operation of the individual system components in the TWR state

System component	Operation description
Controller	Is connected to the CLE, the “TWR” command (with selected anchors) has been sent, awaiting the ranging results from the CLE and any further commands from the user
CLE	Is connected to the anchors. The “TWR” command has been sent to the anchors. Awaiting the result of ranging from the anchors. Any range results are sent to the Controller
Displayer	Not involved in TWR operation, can be connected to the CLE
Anchor	Is connected to the CLE, “TWR” commands received from the CLE. Configured as wither the Initiator or the Responder and performing the TWR and then reporting the result to the CLE
Tag	Should be switched off to minimise interference

5.1.4 COMMUNICATION test state

The communication state is designed to help in installation in a complex environment by measuring the “packet” loss and determining which of the anchors to configure as the Primary and Secondary masters, **Table 14**.

Table 14: Operation of the individual system components in the COMMUNICATION test state

System component	Operation description
Controller	Is connected to the CLE, the “COMM” command has been sent, awaiting the communication test results from CLE and any further commands from the user
CLE	Is connected to the anchors. The “COMM” command has been sent to the anchors. Awaiting the communication test to complete and receive a response from the anchors. On the reception of test completion response, the CLE asks anchors for the communication results and reports them to the Controller
Displayer	Not involved in Communication test operation, can be connected to the CLE
Anchor	Is connected to the CLE, “COMM” command received from the CLE. If configured as the *Transmitter* it will transmit packets, all other anchors are *Receivers* and are in reception mode listening counting any received packets. *Transmitter* reports to the CLE that it has finished and the CLE will then request all of the *Receivers* for a report of number of successful receptions
Tag	Should be switched off to minimise interference

5.1.5 POWER test state

The state to measure the TX power of the anchor for test/production purposes, **Table 15.**

Table 15: Operation of the individual system components in the POWER test state

System component	Operation description
Controller	Is connected to the CLE, the “POWER” command has been sent, awaiting any further commands from the user
CLE	Is connected to anchor. The “POWER” command has been sent to the anchor. Awaiting any further commands from the Controller
Displayer	Not involved in Power test operation, can be connected to the CLE
Anchor	Is connected to the CLE, “POWER” commands received from the CLE. The anchor will transmit long frames in continuous frame mode to enable measure the TX power on external test equipment
Tag	Should be switched off to minimise interference

5.2 *Summary of component APIs*

This briefly lists the common APIs of the individual system sub components.

5.2.1 Tag

The tag has only one mode of operation, i.e. it transmits blink messages at the configured UWB operating mode and sleeps for the defined sleep period. The UWB frame is the only API, see 5.4.1.

5.2.2 Anchor

In RTLS state slave anchors are in permanent receive at the configured UWB mode of operation, reporting received CCP and blink RX times. Master anchors are mostly in permanent receive like the slaves, except once per CCP transmission period they briefly go into transmission mode to send their CCP frame.

Table 16 below is a summary of the API between anchor and the CLE.

Table 16: Anchor/CLE API

API	Direction	Description
Blink RX report	To CLE	Anchor report RX of blink message, with tag ID and RX timestamp
CCP RX report	To CLE	Anchor report RX of CCP message, with master ID and RX timestamp
CCP TX report	To CLE	Master anchor report of CCP transmission with TX timestamp
Configure	To anchor	The CLE configuration of anchor operation modes
Do COM Test	To anchor	Request to perform a communications test
COM test report	To CLE	Report of the result of the communications test
Do TWR	To anchor	Request to perform a two-way ranging measurement
TWR report	To CLE	Report of the result of two-way ranging measurement
Start RTLS	To anchor	Instruct anchors to begin normal RTLS operation reporting CCP and blink RX, and masters periodically sending CCP

5.2.3 CLE configuration options and APIs

5.2.3.1 Multilateration 2D / 3D

The positions of the mobile tag nodes are estimated by the CLE relative to the origin of the fixed anchors (infrastructure). The estimated location of the tag is reported as X, Y co-ordinates when using the 2D solver, or X, Y, Z coordinates when using the 3D solver. The estimated location is also included a list of anchors, which were used in current multilateration.

When using the 3D solver, the anchor nodes should be positioned at different heights, for example with four anchors in a rectangle one diagonal should be low and the other diagonal high. For larger areas we recommend a layout with alternating higher and lower anchors. The larger the Z separation that can be achieved between highest and lowest anchors the better the location estimation result of the 3D solver.

In practice, a good Z separation is hard to achieve and where XY location is sufficient, the 2D location solver will give better results. For 2D solver the anchors should be positioned at the same height to give the best accuracy.

In TTK1000 RTLS system, 2D solver is configured by default, see **Table 7.**

5.2.3.2 Clock sync: wired, wireless

Wireless clock synchronization is used in TTK1000 by default as shown in **Table 7.**

Rudimentary support for wired clock synchronization is implemented in the CLE and can be chosen at start-up time, by the *wirelessSync* parameter (see **Table 7**). Wired clock support requires different anchor hardware and an additional wired clock distribution/synchronization box. These are not part of the TTK1000 delivery.

5.2.3.3 System diagnostic modes

The initial diagnostic mode is configured in the CLE configuration file Table 7. During run time, the Control application can start and stop logging in the CLE with the "*log start*" and "*log stop*" *command* message, see 5.2.4. The diagnostics logging level is indicated in the log start message. Higher logging levels aggregate the lower levels. The diagnostic/logging level modes supported by the TTK1000 are listed below.

Table 17: Diagnostic modes

Logging level	Description
0	All logging is switched off
1	Errors and warnings in the CLE
2	As level 1 and adds RX/TX time stamps
3	As level 2 and adds Blinks, Clock Calibration Packets from tracked masters only, DW1000 RX frame diagnostics, Kalman synchronization output, multilateration input and output
4	As level 3 and adds Clock Calibration Packets from all masters

5.2.3.4 Static IP configuration option of CLE

By default the CLE discovers the Anchors in the network using mDNS discovery protocol. However in large installations we recommend to use a "Static" configuration where all the anchor IP addresses are known and fixed in the DHCP server configuration file. This improves the speed of anchor discovery and increases the robustness of the whole RTLS system. The use of static IP's is enabled automatically when the CLE detects the existence of and reads from a configuration file called "cle_dns.cfg", which contains a list of anchors and corresponded IP addresses.

5.2.4 CLE APIs to upper layer applications

Table 18 and **Table 19** below summarise the API between the CLE and the upper layer client applications.

Table 18: CLE/Control Client API

API	Direction	Description
Anch list request	To CLE	Controller requests a list of discovered anchors from the CLE
Anch list response	From CLE	The CLE responds with the list of discovered anchors
UWB configuration request	To CLE	Controller configures the CLE UWB operating mode: Channel, PRF, preamble code, PSR(PLEN), PAC size and SFD type to be used in the system
UWB configuration response	From CLE	The CLE confirms that it has received UWB configuration command
Anch configuration request	To CLE	Controller requests the CLE to connect and provides the anchors' configuration: - installation positions X,Y,Z coordinates, - role in the RTLS network either master or slave, - indicating which masters CCP should be tracked, - for "secondary" masters in multi master RTLS network - which "primary" master should it follow
Anch configuration response	From CLE	CLE responds indicating whether the requested anchors have been successfully configured and/or are successfully been connected to
START/STOP RTLS command	To CLE	Controller requests the CLE to START/STOP RTLS operation.
START/STOP logging command	To CLE	Controller requests the CLE to START/STOP the logging operation. Upon the start specific diagnostic log level can be provided to change the CLE startup logging level configuration
Communication test request	To CLE	Controller requests the CLE to perform a communication test with - the number of packets to transmit from the *Transmitter* - the list of *Receiving* anchors
Communication test response	From CLE	The CLE responds to the controller with the list of reports from the communication test between anchors
Range measurement request	To CLE	Controller requests to perform a Two Way Ranging test with - the number of ranging attempts - the list of anchors involved in TWR
Range measurement response	From CLE	CLE responds to the controller with range measurements between anchors

On connection of each of Data clients, the CLE provides the system configuration information and then continuously reports tags estimated locations as per **Table 19**.

Table 19: CLE/Display Client API

API	Direction	Description
Anchor data report	From CLE	For each anchor the CLE gives a position - anchor ID (64-bit unique address) - X,Y,Z coordinates
Tag position report	From CLE	The CLE gives a location estimate for a tag - tag ID (64-bit unique address) - X,Y,Z coordinates - Number of anchors in multilateration List - List of anchors' IDx, used in multilateration
RTLS data statistic report	From CLE	The CLE gives statistics for a tag - tag ID (64-bit unique address) - this is the % of received blinks - this is the % of successful multilaterations

5.3 *UWB physical layer modes and APIs*

There are a large number of UWB operation modes in DW1000. The TTK1000 can configure any. Decawave has performed tests of the TTK1000 performance measurements using channel 2 (centre frequency of 3993.6 MHz and bandwidth of 499.2 MHz) for the configuration modes specified in **Table 20** below.

For the full list of DW1000 supported channels and modes of operation please see the DW1000 datasheet [3]. Please refer to the IEEE802.15.4-2011 [1] standard for details of the UWB PHY modulation.

Table 20: Tested operating modes

PSR / PRF / Data rate	Grid test	TWR test	Capacity test
128 / 64MHz / 6.81 Mbps	√	√	√
128 / 16MHz / 6.81 Mbps	√	√	
256, 512, 1024, 2048 / 64 MHz / 6.81 Mbps		√	
512 / 64 MHz / 850 Kbps		√	√
1024 / 64 MHz / 110 Kbps		√	√

5.4 *TTK1000 over-the-air frame formats*

This describes the over-the-air formats of the UWB frames transmitted by tags and anchors. The general IEEE 802.15.4-2011 UWB frame structure is as shown in Figure 6. Detailed descriptions of the frame format are given in the standard [1]. The frame consists of a synchronisation header (SHR) which includes the preamble symbols and start of frame delimiter (SFD), followed by the PHY header (PHR) and then the PHY data payload.

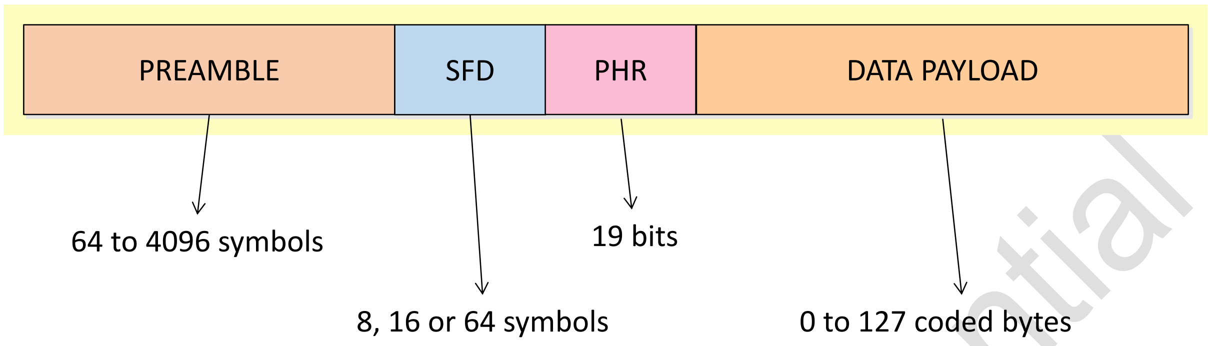

Figure 6: IEEE802.15.4-2011 PPDU Structure

5.4.1 The data blink message format used in TTK1000 RTLS

The tag blink message format is shown on Figure 7. It is 12 bytes long and follows ISO/IEC 24730-62 Air-Interface-Protocol.

Byte	0	1	2	3	4	5	6	7	8	9	10	11
Parameter	FC	Seq#	Tag Address								CRC	

Figure 7: Blink frame byte format

Table 21: Field definitions within the Bink frame

Parameter	Value	Description
FC	0xC5	1 byte frame control as per Figure 7
Seq#	-	This is a modulo 256 frame sequence number
Tag Address	-	8 byte address (tag ID), 64-bit unique address
CRC	-	This is the frame check sequence

5.4.2 The CCP message format used in TTK1000 RTLS

The CCP message format is shown on Figure 8. The structure is defined in **Table 22** below.

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Parameter	FrameCtrl		Seq#	PANID		Dest Addr		Src Addr								FnCd	CRC	

Figure 8: CCP frame byte format

Table 22: Field definitions within the CPP frame

Parameter	Value	Description
FrameCtrl	0x41C8	2 byte frame control
Seq#	-	This is a modulo 256 frame sequence number
PANID	0xDECA	2 byte PAN ID (hardcoded to 0xDECA)
Dest Addr	0xFFFF	2 byte destination address (this is a broadcast frame, 0xFFFF)
Src Addr	-	8 byte address of the master anchor (64-bit unique address)
FnCd	0x2B	1 byte Function Code - identifies this as a CCP
CRC	-	This is the frame check sequence

6 Typical Performance

This chapter outlines the performance of the whole system as well as performance of individual components.

6.1 *CLE wireless clock synchronization performance*

The TTK1000 includes an implementation of Decawave's clock synchronization algorithm to track all anchors' timebase in one clock domain. Below are the results of clock synchronization algorithm performance.

This CCP tracking algorithm performance result comes from a system with one master anchor sending wireless clock calibration packets and three slave anchors receiving them. A tag, sending a blink messages is placed in a fixed central position.

The CLE gets CCP TX timestamps from the master anchor (MA) and CCP RX time-stamps from each of the three slave anchors (SA) in their individual local time-bases. Then the CLE runs separate clock tracking instances, one for each of the three slave anchors. Also the CLE gets a blink RX time-stamps from all four anchors, each in the anchor's own local time-base.

For each slave anchor, the algorithm uses the state of the clock tracking instance to correct the blink RX time-stamps into the time-base of master anchor. Then the corrected time-stamps which are in the master time base are used to calculate the time-difference-of-arrival (TDOA), of each blink message, between the master anchor and each slave anchor. Below is the plot of TDOA (Y) from a blink (X) - the time difference results showing how the wireless synchronisation scheme is working in performing the clock tracking and time-stamp corrections. The plot shows the three traces, which represent individual time difference of arrival of the blinks for an individual slave anchor with respect to the master anchor. This has the following characteristics:

A CCP was transmitted from the master anchor every 150 milliseconds.

A blink was transmitted from the tag on average every 333 milliseconds.

The standard deviations of the TDOA for the three slaves respectively were 0.18 ns, 0.19 ns and 0.14 ns over the period of the test, which was more than 12 hours. The largest of these is 197 picoseconds which represents a standard deviation of less than 6 cm.

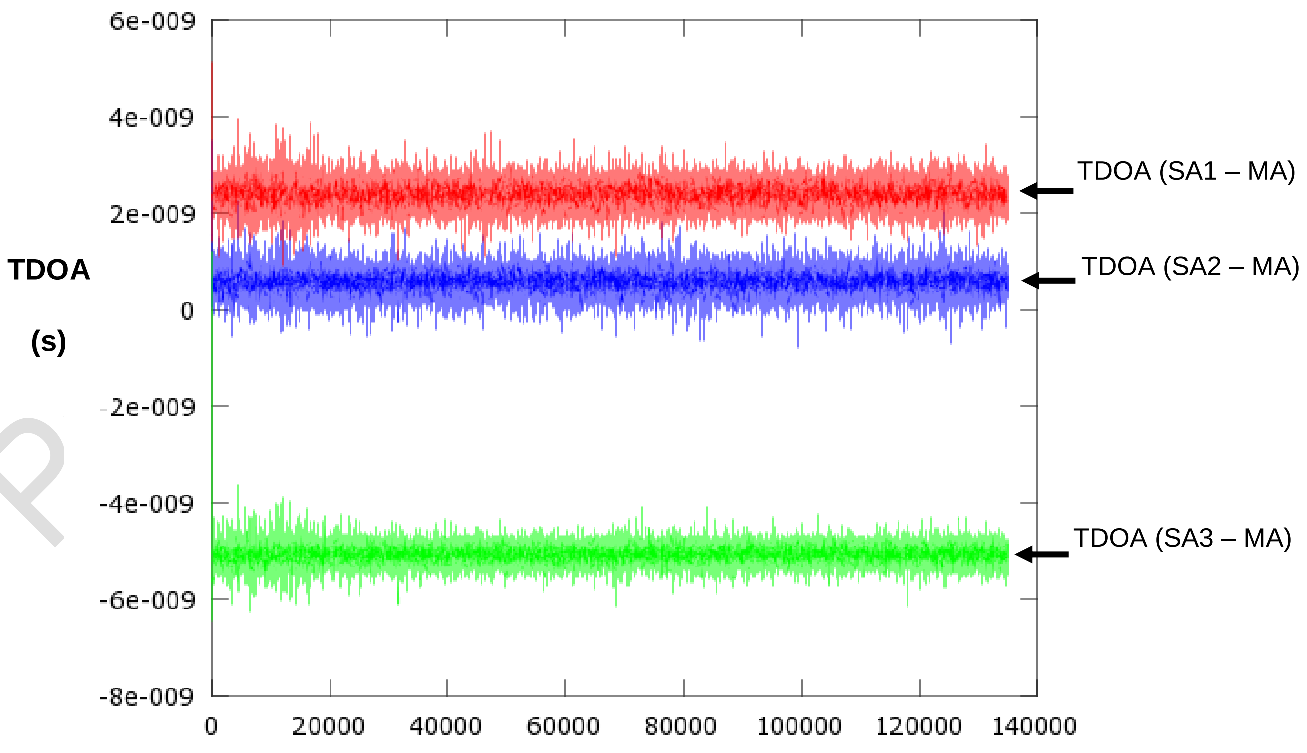

Figure 9: Wireless clock synchronization performance (TDOA vs RX frame number)

6.2 *Overall RTLS SYSTEM performance*

The single master performance test was performed on a grid test at 64 MHz PRF on channel 2 with a data rate of 6.81 Mb/s, using a 2D-solver (default CLE configuration). Individual anchor antenna delays were not determined, instead a global fixed antenna delay figure was used. Decawave would recommend individual determination of anchor antenna delays in anchor HW production/manufacturing to give optimum performance.

The methodology is as follows: first, the isolated network was set up with DHCP server running on it. Then all the anchors were placed on the walls at the same height 2.07 m to emulate a typical small system installation. One master (M3) and three slave anchors were configured (S1, S2 and S4). One tag, blinking at 10 Hz was used at the height of 1.46 m to perform test in each of the grid locations.

Multilateration success rate, R95 and Standard deviation of 2D X and Y coordinates are used to quantify the system performance. The R95 represents the radius wherein 95% of locations are reported. The success rate represents the ability of the CLE solver to multilaterate at the given tag position.

6.2.1 Measured R95 Grid Test

The measured R95 (in cm) for each of the locations tested in the partial grid test is indicated in the **Table 23**. X indicates coordinates that were not tested. Each coordinate test was run for 3 mins (about 2000 blinks).

Table 23: R95 (cm) for the grid accuracy test with 64 MHz PRF.

Y(m) / X(m)	1	2	3	4	5
4	21.8	X	X	X	X
3	X	18.5	24.7	X	18.0
2	X	19.7	26.8	X	X
1	21.1	X	X	X	18.3

The figure below depicts the R95 results from the grid test of the table above.

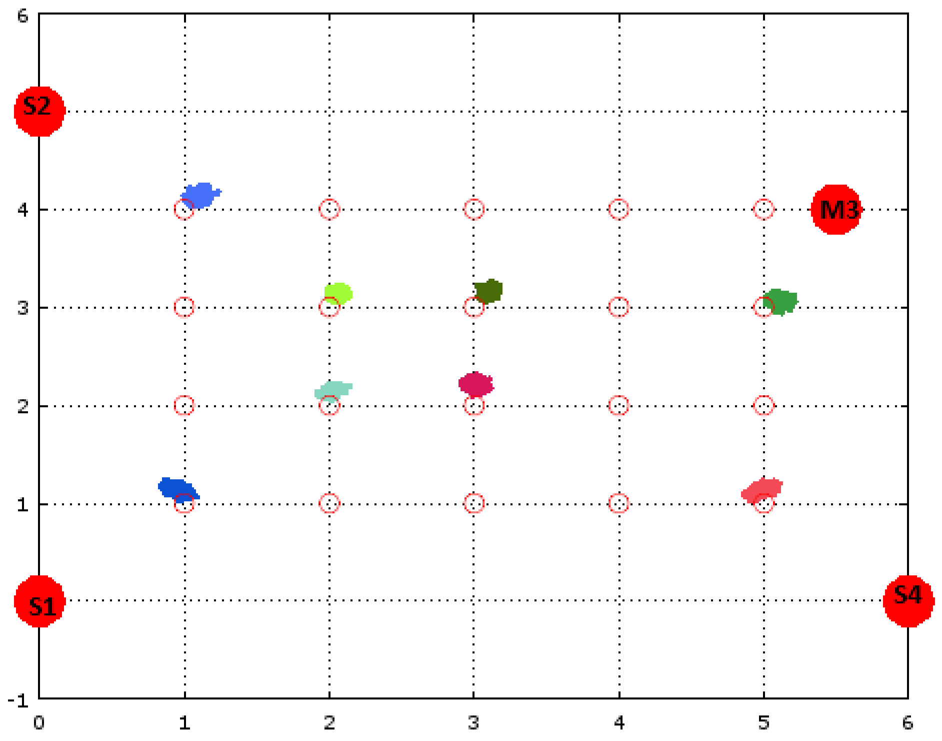

Figure 10: Scatter plot for the grid accuracy test

6.2.2 Multilateration success rate

The measured multilateration success rate for each of the locations tested in the grid test is indicated in the following table. X indicates coordinates that were not tested. Each coordinate test was run for 3 mins (about 2000 blinks).

Table 24: Multilateration rate for the grid accuracy test with 64 MHz PRF

Y(m) / X(m)	1	2	3	4	5
4	100	X	X	X	X
3	X	100	100	X	100
2	X	100	100	X	X
1	100	X	X	X	100

6.2.3 Standard deviation

The measured standard deviation of tag's x coordinate (in cm) for each of the locations depicted in the grid above is indicated in the following table. An X indicates coordinates that were not tested. Each coordinate test was run for 3 mins (about 2000 blinks).

Table 25: Standard deviation (cm) of x for the grid accuracy test with 64 MHz PRF

Y(m) / X(m)	1	2	3	4	5
4	3.1	X	X	X	X
3	X	2.4	2.5	X	3.2
2	X	3.5	2.9	X	X
1	3.6	X	X	X	3.8

The measured standard deviation of tag's y coordinate (in cm) for each of the locations depicted in the grid above is indicated in the following table. An X indicates coordinates that were not tested. Each coordinate test was run for 3 mins (about 2000 blinks).

Table 26: Standard deviation (cm) of y for the grid accuracy test with 64 MHz PRF

Y(m) / X(m)	1	2	3	4	5
4	3.4	X	X	X	X
3	X	2.6	3.3	X	3.2
2	X	2.6	3.2	X	X
1	3.1	X	X	X	3.1

6.3 *Tag capacity test*

This test is used to verify that the CLE performance is within expectations in a high tag density system. The capacity data rate test performed on high data rate 6.8 Mbits/s (PRF = 16 MHz; PSR = 128; PAC = 8) with a 16 tags, blinking independently with an interval of 23 ms each. This give the aggregate blink rate 700 blinks a second. The amount of lost blinks is 8.9% due to blinks collisions.

6.4 *Tag Randomisation*

When a number of TDOA tags are placed in the same area it is important to minimise interference. If the tags are blinking at the same time their transmissions can overlap and hence interfere. This overlapping/collisions need to be minimised, thus the blink period is randomized. A standard C random function is used for this and it was shown that it has a uniform distribution. The default blink period of 100 ms is equally distributed between (90 and 110 ms).

7 TTK1000 DELIVERY

7.1 *Delivery overview*

The TTK1000 is a software delivery of sources, precompiled binaries and documentation for the particular hardware platform based on STM Cortex M4 MCU family. A physical platform, consisting of 10 anchor units and 20 tag units, is provided as the platform to prove the initial system build is operational and operating as intended.

This TTK1000 delivery contains all of the source code and supporting documentation to accelerate the development of the following products:

1. Tag firmware, targeted for the Nordic nRF52832 ARM on the reference tag (DWM1001 tag) hardware
2. Anchor firmware, targeting the STM ARM on the reference anchor (TDoA anchor) HW
3. Central Location Engine, targeted for a Windows PC.
4. Control Client, targeted for a Windows PC
5. Display Client, targeted for a Windows PC.
6. Update tool, the anchor firmware update utility, targeted for a Windows PC.

7.2 *The list of electronically delivered content of TTK1000*

Documentation

Document	Description
TTK1000_RTLS_System_Architecture.pdf	Overview of the RTLS architecture
TTK1000_Tag_Software_Guide.pdf	Tag firmware guide
TTK1000_Anchor_Software_Guide.pdf	Anchor firmware guide
TTK1000_Controller_Software_Guide.pdf	Control client software guide
TTK1000_RTLS_CLE_Architecture.pdf	Overview of the Central Location Engine (CLE)
TTK1000_RTLS_CLE_Anchor_API_Spec.pdf	CLE Anchor API user guide
TTK1000_RTLS_CLE_Controller_API_Spec.pdf	CLE Control API user guide
TTK1000_Display_Software_Guide.pdf	Display client software guide
TTK1000_RTLS_System_Installer_Guide.pdf	RTLS system installation guide

Source code folders structure

Directory	Sub-Directory/File	Contents
ARM/		
	anchor/	Contains Anchor precompiled binaries
	tag/	Contains Tag precompiled binaries
Logs/		Empty directory for log files
PC/		Contains PC executables for CLE, Controller, Display and Anchor firmware updater
Sources/		
	dw_anchor.zip	Source files for the Anchor firmware. Build using the Makefile targeting any GNU ARM compiler tools chain. We use Code Sorcery from Mentor Graphics and Eclipse IDE
	dw_cle.zip	Source files for the Central Location Engine, build using the project file DW_LE.sln in Microsoft Visual Studio C++ 2015

Directory	Sub-Directory/File	Contents
	dw_controller.zip	Source files for the control application. Use dw_controller.pro project file in QT Creator
	dw_displayer.zip	Source files for the display application. Use dw_display.pro project file in QT Creator
	dw_tag.zip	Source files for the tag firmware. Build using the Makefile targeting any GNU ARM compiler tools chain. We use Code Sorcery from Mentor Graphics and Eclipse IDE. Additional libraries need to be downloaded as described in TTK1000_Tag_Software_Guide.docx
	dw_updater.zip	Source files for the anchor firmware update application. Use dw_updater.pro project file in QT Creator

8 GLOSSARY

Table 27: Glossary of Terms

Abbreviation	Full Title	Explanation
EIRP	Equivalent Isotropically Radiated Power	The amount of power that a theoretical isotropic antenna (which evenly distributes power in all directions) would emit to produce the peak power density observed in the direction of maximum gain of the antenna being used
ETSI	European Telecommunication Standards Institute	Regulatory body in the EU charged with the management of the radio spectrum and the setting of regulations for devices that use it
FCC	Federal Communications Commission	Regulatory body in the USA charged with the management of the radio spectrum and the setting of regulations for devices that use it
GPIO	General Purpose Input / Output	Pin of an IC that can be configured as an input or output under software control and has no specifically identified function
IEEE	Institute of Electrical and Electronic Engineers	Is the world's largest technical professional society. It is designed to serve professionals involved in all aspects of the electrical, electronic and computing fields and related areas of science and technology
LNA	Low Noise Amplifier	Circuit normally found at the front-end of a radio receiver designed to amplify very low level signals while keeping any added noise to as low a level as possible
LOS	Line of Sight	Physical radio channel configuration in which there is a direct line of sight between the transmitter and the receiver
NLOS	Non Line of Sight	Physical radio channel configuration in which there is no direct line of sight between the transmitter and the receiver
PPM	Parts Per Million	Used to quantify very small relative proportions. Just as 1% is one out of a hundred, 1 ppm is one part in a million
RF	Radio Frequency	Generally used to refer to signals in the range of 3 kHz to 300 GHz. In the context of a radio receiver, the term is generally used to refer to circuits in a receiver before down-conversion takes place and in a transmitter after up-conversion takes place
RTLS	Real Time Location System	System intended to provide information on the location of various items in real-time
SPI	Serial Peripheral Interface	An industry standard method for interfacing between IC's using a synchronous serial scheme first introduced by Motorola
TCXO	Temperature Controlled Crystal Oscillator	A crystal oscillator whose output frequency is very accurately maintained at its specified value over its specified temperature range of operation
TWR	Two Way Ranging	Method of measuring the physical distance between two radio units by exchanging messages between the units and noting the times of transmission and reception. Refer to Decawave's website for further information
TDOA	Time Difference of Arrival	Method of deriving information on the location of a transmitter. The time of arrival of a transmission at two physically different locations whose clocks are synchronized is noted and the difference in the arrival times provides information on the location of the transmitter. A number of such TDOA measurements at different locations can be used to uniquely determine the position of the transmitter. Refer to Decawave's website for further information
UWB	Ultra Wideband	A radio scheme employing channel bandwidths of, or in excess of, 500 MHz
WSN	Wireless Sensor Network	A network of wireless nodes intended to enable the monitoring and control of the physical environment

9 REFERENCES

[1] IEEE802.15.4-2011 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/

[2] DW1000 User Manual www.decawave.com

[3] DW1000 Data Sheet www.decawave.com

[4] DWM1001 www.decawave.com

[5] Decawave APS021 Developing an RTLS product

[6] Armadillo library http://arma.sourceforge.net/

[7] Libxml2 library http://xmlsoft.org/

10 DOCUMENT HISTORY

Table 28: Document History

Revision	Date	Description
1.0	07th Nov 2018	TTK1000 Release 1.0

11 KNOWN ISSUES

Table 29: Known Issues

	Issue Description
Anchor Hardware	Version 2.0 and Version 2.1 Anchor hardware has a weakness in the Ethernet circuit design. This could mean that on some boards the Ethernet PHY may get damaged due to voltage transients. Please see updated schematics Version 2.2 with implemented fix in place. Existing hardware can be easily retrofitted with protection diodes to help alleviate the problem.
Ethernet Infrastructure	The anchor has been shown to work correctly with Ethernet switches manufactured by CISCO and HP. We have found that the Ethernet switch “DELL PowerConnect 5448” does not assign a DHCP IP address to the anchor when they are directly connected. However, if the Dell switch and the anchor are connected via a simple Ethernet hub, the switch will assign an IP address correctly, see Figure 11.

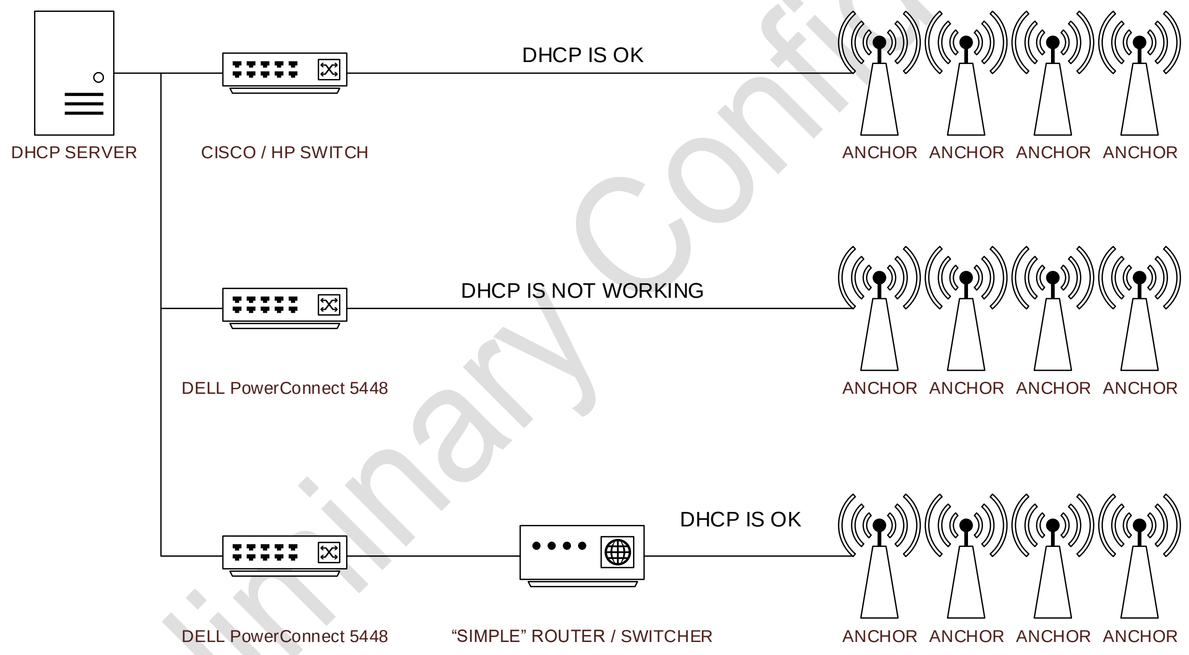

Figure 11 Ethernet infrastructure issue

12 FURTHER INFORMATION

Decawave develops semiconductors solutions, software, modules, reference designs - that enable real-time, ultra-accurate, ultra-reliable local area micro-location services. Decawave's technology enables an entirely new class of easy to implement, highly secure, intelligent location functionality and services for IoT and smart consumer products and applications.

For further information on this or any other Decawave product, please refer to our website www.decawave.com.