Università degli Studi di Camerino

Scuola di Scienze e Tecnologie

Corso di Laurea in Informatica (Classe L-31)

Ultra wideband based Real Time Localization System infrastructure using Decawave's DWM1000 modules

Laureandi
Michele Biondi
Matricola 096138

Andrea Salvatori
Matricola 095867

Relatore
Prof. Fausto Marcantoni

A.A. 2017/2018

Abstract

Uno degli argomenti più discussi negli ultimi anni è la ricerca di sistemi di localizzazione al chiuso. Siccome la tecnologia GPS è molto diffusa e usata giornalmente sappiamo già quante possibilità un sistema del genere potrebbe avere, ma tuttora ancora non si è riusciti a creare una soluzione economica e facile da installare poiché non è possibile utilizzare l'infrastruttura GPS già esistente per via della dispersione del segnale all'interno di edifici.
Tuttavia basandosi sulle esperienze degli ultimi anni la tecnologia UWB può essere un candidato ideale per questo ruolo. L'utilizzo di impulsi della durata di picosecondi anche in ambienti pieni di riflessioni, permettono la separazione di segnali che arrivano da percorsi diversi per via della propagazione multipath.
Perciò il tempo di arrivo di un impulso ad un ricevitore può essere definito con una precisione utile all'utilizzo di algoritmi ben noti per la localizzazione.
In questa tesi andremo ad analizzare specificamente questa tecnologia, facendo prima però una panoramica su tutte le tecnologie disponibili al momento.
Per verificarne l'effettiva validità e facilità di implementazione è stato sviluppato un progetto basato su dei moduli commerciali UWB.
Il lavoro ha portato alla creazione di due librerie open source utilizzabili per la creazione di Real Time Location Systems.
Le maggiori difficoltà riscontrate durante il periodo di studio e sperimentazione sono dovute alla mancanza di hardware con un semplice interfacciamento per l'utilizzo dei migliori algoritmi utilizzabili.

Indice

RTLS - Real-time Location System

La dicitura RTLS (Real time location system) descrive una classe di sistemi che producono informazioni in tempo reale sulla posizione oggetti, animali, persone etc..
Per locazione non si intende soltanto un punto preciso su una mappa ma anche ad esempio la prossimità ad un determinato oggetto o luogo, in generale che diano informazioni utili all'utente del sistema.
L'applicazione RTLS più famosa e utilizzata è il GPS, che però non è utilizzabile per quanto riguarda la localizzazione all'interno di edifici.

Casi d'uso RTLS

I casi d'uso[1] più comuni sono:

- Prossimità:
    - La distanza tra le chiavi di un'auto e l'auto stessa.
    - La distanza tra una persona e il proprio laptop.
- Posizionamento assoluto rispetto ad elementi fissi (localizzazione)
- Distanza relativa tra un gruppo di nodi

Possibili tecnologie per sistemi di localizzazione indoor

Di seguito andremo ad analizzare le varie tecnologie utilizzabili per sistemi RTLS indoor:

- WiFi: Economico perchè ormai onnipresente, tuttavia ha dimostrato una precisione molto bassa (errore 1-8 m).
- BLE (Bluetooth Low-Energy): Consiste nell'utilizzo di vari beacon BLE per localizzare dispositivi BLE enabled (come gli smartphone). Migliore precisione (errore fino a 1m) rispetto al WiFi e autonomia dei beacon che può raggiungere anni. Gli smartphone sono però lenti a rispondere ai vari 'beacon' impiegando dai 3 ai 6 secondi all'incirca, ottenendo così un sampling basso di localizzazione.
- RFID: Il migliore per quanto riguarda la precisione (errori fino a 5cm). I problemi per creare soluzioni basati su questa tecnologia sono più che altro dovuti agli elevati costi per ottenere distanze utilizzabili in un sistema RTLS (e comunque non paragonabili a quelle ottenibili da altre soluzioni)
- UWB (Ultra-wideband): tecnologia a radiofrequenza a largo spettro di frequenza (3-7 Ghz). Fa utilizzo di dispositivi chiamati *ancore* che rappresentano le posizioni fisse e dei *tag* che corrispondono ai dispositivi

[1] "APS003 DW1000 RTLS Introduction - Decawave." https://www.decawave.com/wp-content/uploads/2018/10/APS003_DW1000-RTLS-Introduction_v1.1.pdf. Accessed 15 Jan. 2019.

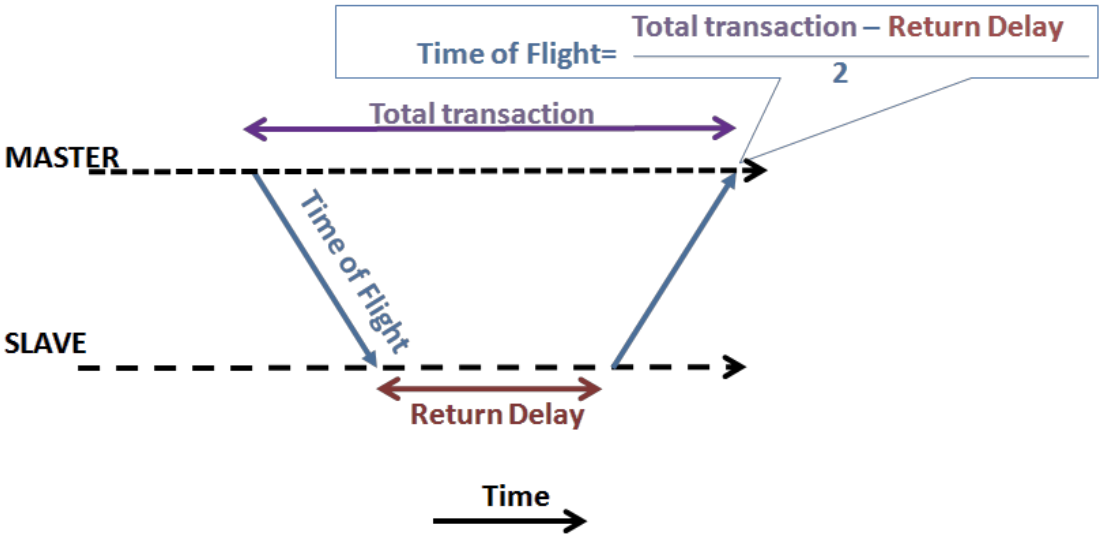
mobili da localizzare. Le ancore vengono solitamente posizionate a una distanza massima di 25 metri a cui i tag inviano impulsi radio per avviare il processo di localizzazione, con una precisione con un errore massimo di 30 cm con aggiornamenti fino ad ogni 50 ms. La precisione è dovuta all'utilizzo di timestamp che hanno come unità qualche picosecondo.

Per le sperimentazioni è stata scelta la tecnologia UWB poichè ci è sembrata la più promettente in questo settore.

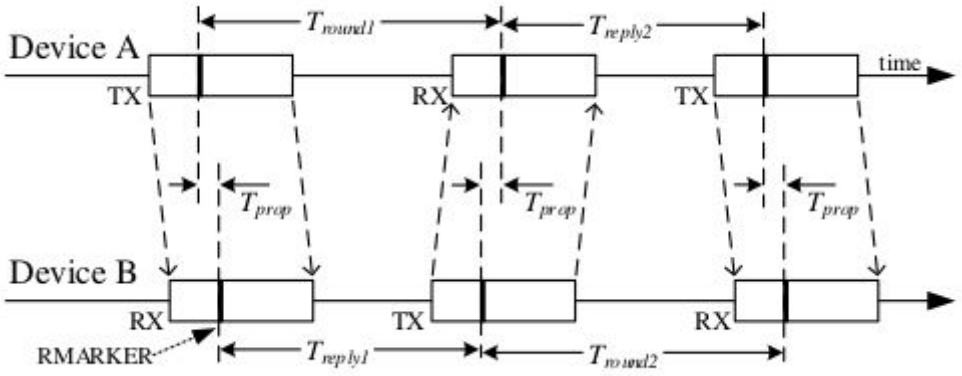
Metodi di implementazione

Sono disponibili vari metodi per l'implementazione di un sistema che utilizza UWB per la localizzazione, ognuno con vari pregi e difetti. Li andremo ad analizzare brevemente qui di seguito:

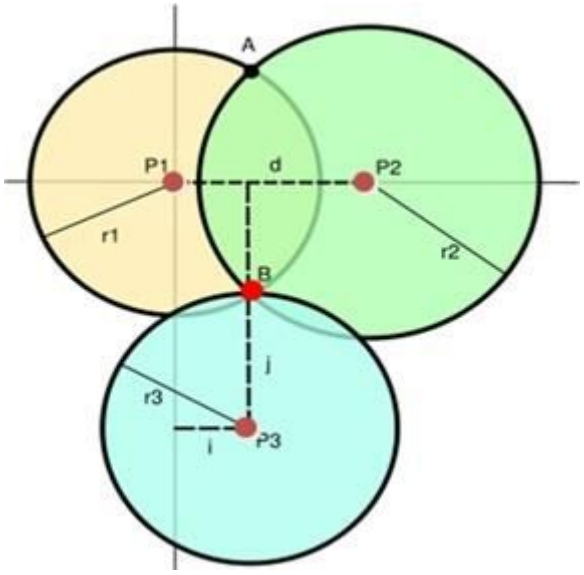
- METODI BASATI SULLA POTENZA: Questi metodi utilizzano la potenza del segnale in arrivo su un ricevitore. Conoscendo la potenza di trasmissione del segnale, le caratteristiche di propagazione di un particolare segnale in aria e l'ambiente di utilizzo si può calcolare dove la trasmissione è stata generata. Seppur adatti ad alcune circostanze, questi metodi hanno una bassa precisione e per aumentarla hanno bisogno di altre tecnologie ausiliarie (come ad esempio gli ultrasuoni). Per questo motivo non sono stati presi in considerazione.
- METODI BASATI SUL TEMPO:
    - TOF (Time of Flight): I sistemi basati su questo metodo si basano sul calcolo del tempo impiegato da un segnale radio per propagarsi da un trasmettitore ad un ricevitore. Conosciuto precisamente questo tempo sarà possibile determinare la distanza utilizzando la velocità della propagazione del segnale radio.



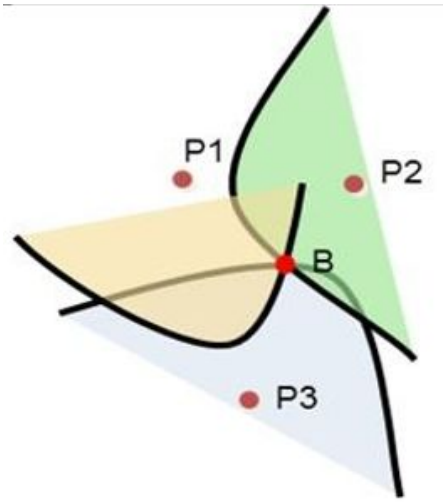
Questa trasmissione risente, tuttavia, del cosiddetto effetto *clock drift*, cioè una effettiva differenza nel calcolo del tempo dei due dispositivi (si pensi ad una sveglia che rimane indietro rispetto ad un orologio). Per ovviare a questo problema e ridurre l'errore si utilizza una tecnica chiamata Double-Sided Two-Way Ranging, dove vengono utilizzati 3 messaggi tra i 2 dispositivi, andando così a compensare questo effetto.



Il posizionamento si ottiene intersecando 3 circonferenze che hanno come raggio la distanza tra il tag e le 3 ancore, il punto di intersezione è la posizione effettiva del tag.

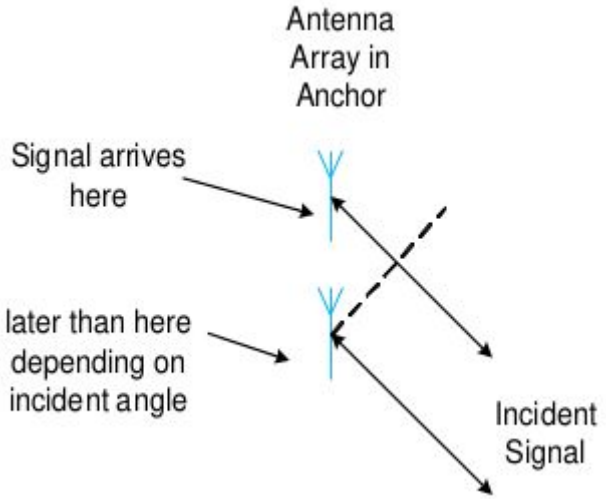


- TDoA(Time difference of arrival): Con questo metodo le ancore sono posizionate in punti fissi conosciuti e devono categoricamente avere i clock sincronizzati. Il tag trasmette un solo messaggio, che viene ricevuto da tutte le ancore a distanza di ricezione. Siccome le onde radio viaggiano ad una velocità costante, il messaggio arriverà in tempi diversi alle ancore in base alla distanza dal tag. Essendo i clock sincronizzati, la differenza di tempo può essere utilizzata per calcolare la



posizione del tag utilizzando una tecnica nota come multi-lateration.
La difficoltà di implementazione di questo sistema è dovuta alla corretta sincronizzazione del clock interno dei dispositivi, che rende difficile e costosa la progettazione hardware e l'installazione del sistema.

- PDoA (Phase Difference of Arrival): utilizza un array di antenne in un ancora. Il time of arrival della prima antenna viene confrontato con gli altri. Se il tag è perpendicolare, allora il segnale arriva allo stesso momento a tutte le antenne, altrimenti se il tag è angolato e i timestamp misurati in modo preciso, si può dedurre attraverso la misura della differenza di fase di arrivo la posizione del tag.
  Questo metodo soffre della propagazione a più vie del segnale e quindi è utilizzabile maggiormente in applicazioni *LoS* (Line of sight, a vista).



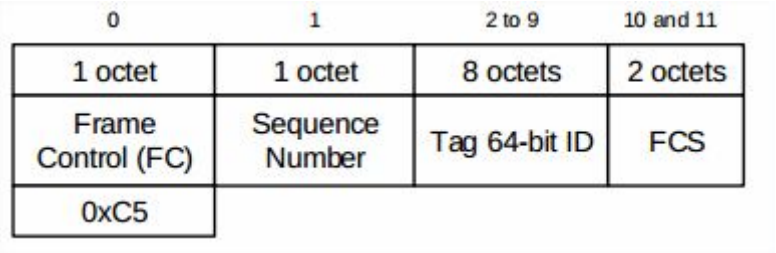
Per gli esempi di questo progetto è stato utilizzato l'algoritmo *Time of Flight* (in particolare Double-sided Two-way ranging) poiché è sicuramente il più veloce da implementare e richiede un hardware più semplice. d'altro canto ci siamo però trovati a confrontarci con le problematiche relative alla gestione di diversi scambi di messaggi tra ancora e tag durante un flow di ranging e la sincronizzazione dei 3 flow che il tag deve effettuare con le 3 ancore per ottenere una localizzazione.

Standard UWB RTLS

L'insieme di standard che riguarda i sistemi RTLS utilizzando l' UWB è l' ISO/IEC 24730.
Per l'impostazione dei dispositivi e la formattazione dei messaggi a livello MAC abbiamo utilizzato lo standard ISO/IEC 24730-62:2013[2] .
I messaggi di questo standard hanno un encoding allineato a quello dello standard tecnico IEEE 802.15.4 .

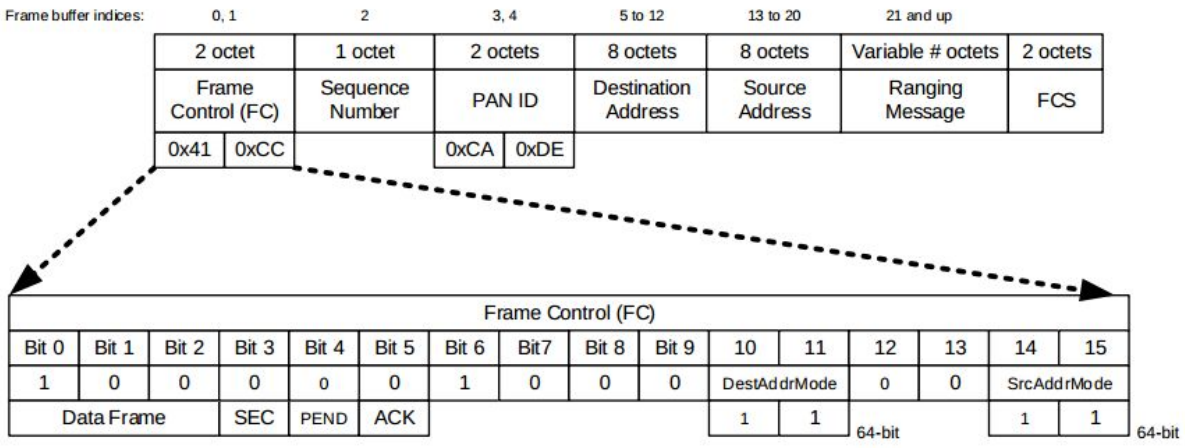
[2] "ISO/IEC 24730-62:2013 - Information technology -- Real time locating ...." https://www.iso.org/standard/60379.html. Accessed 15 Jan. 2019.

Questo standard fa utilizzo di un messaggio denominato *blink* per notificare l'infrastruttura (di cui fanno parte le ancore) della presenza di un tag, l'infrastruttura poi deciderà come gestire la localizzazione (Ad esempio quale algoritmo scegliere).
Un blink minimale ha la seguente forma:



0	1	2 to 9	10 and 11
1 octet	1 octet	8 octets	2 octets
Frame Control (FC)	Sequence Number	Tag 64-bit ID	FCS
0xC5			

Per i messaggi relativi alla trasmissione di dati il formato è simile al seguente:

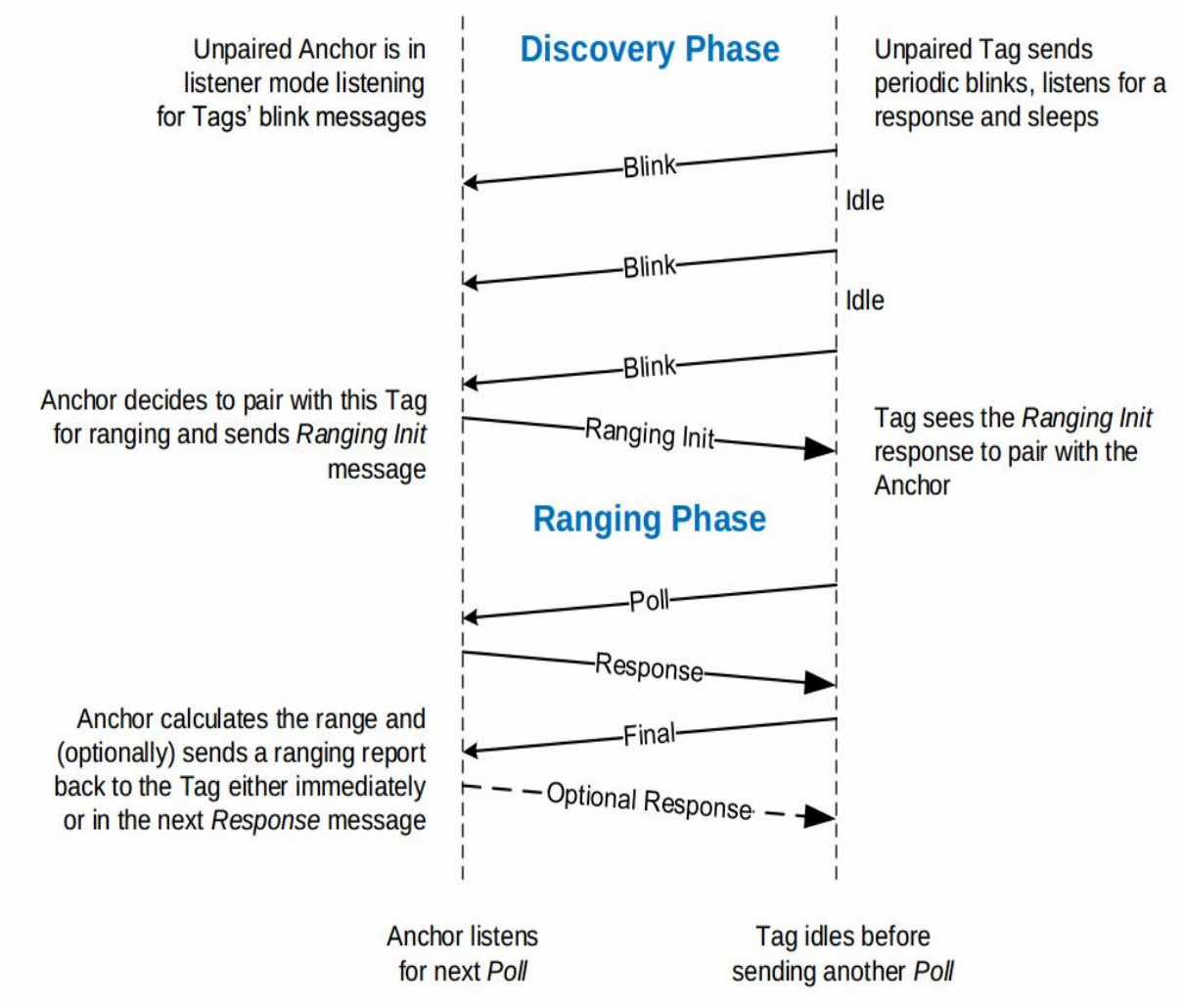


Frame buffer indices:	0, 1	2	3, 4	5 to 12	13 to 20	21 and up	
	2 octet	1 octet	2 octets	8 octets	8 octets	Variable # octets	2 octets
	Frame Control (FC)	Sequence Number	PAN ID	Destination Address	Source Address	Ranging Message	FCS
	0x41 / 0xCC		0xCA / 0xDE				

Frame Control (FC)															
Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit7	Bit 8	Bit 9	10	11	12	13	14	15
1	0	0	0	0	0	1	0	0	0	DestAddrMode		0	0	SrcAddrMode	
Data Frame			SEC	PEND	ACK					1	1	64-bit		1	1

64-bit

Nel nostro caso, utilizzando il two way ranging si avrà una serie di messaggi dati simile al seguente:



DW1000 Integrated Circuit

Il DW1000[3] è un chip integrato CMOS RF transceiver UWB (Ultra Wideband) a basso consumo conforme allo standard IEEE802.15.4-2011 .

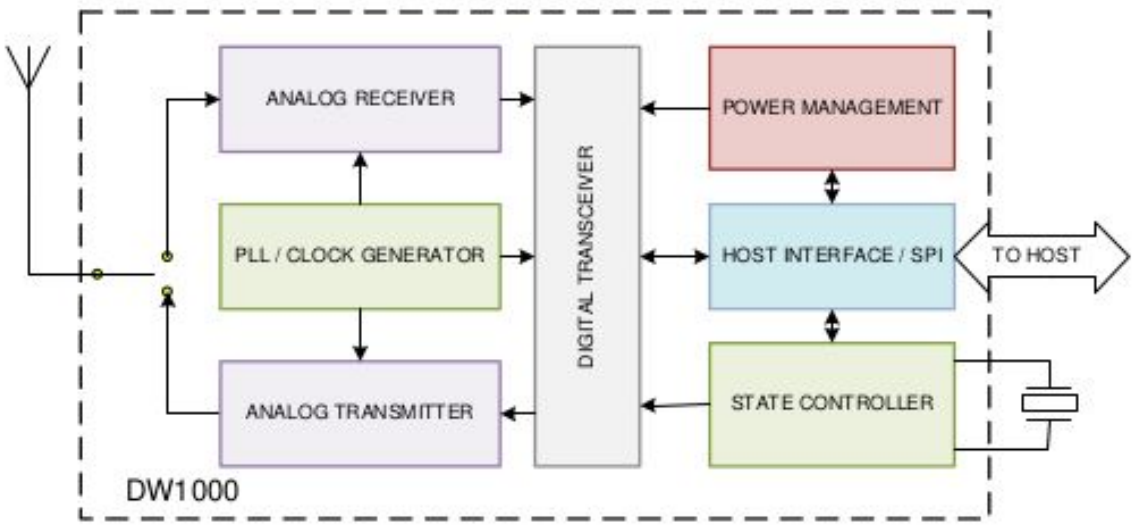
Features

Le principali features[4] di questo integrato sono:

- Supporta 6 canali RF da 3.5GHz a 6.5GHz con potenza regolabile
- L'input voltage è compreso tra 2.8V e 3.6V
- Modalità sleep con consumo di 1μA/h e deep sleep 50nA/h
- Data rates di 110Kbps, 850Kbps e 6.8Mbps
- Lunghezza massima dei pacchetti 1023 bytes
- Varie feature di supporto a livello MAC come CRC e frame filtering
- Supporta Two Way Ranging e TDOA
- Interfaccia SPI per comunicazione con l'host (slave only), con clock massimo 20MHz
- Dimensioni 6mm x 6mm 48-pin, formato QFN
- Precisione fino a 10 cm
- Immunità al multipath fading
- Sensori interni per il voltaggio in input e la temperatura
- Memoria One-Time Programmable (OTP) per salvare valori di calibrazione
- Memoria Always-On, usata per mantenere la configurazione del dispositivo durante gli stati di basso consumo

[3] "dw1000 user manual - Decawave." https://www.decawave.com/wp-content/uploads/2018/09/dw100020user20manual_0.pdf. Accessed 15 Jan. 2019.
[4] "DW1000 Datasheet - Decawave." https://www.decawave.com/sites/default/files/resources/dw1000-datasheet-v2.09.pdf. Accessed 15 Jan. 2019.

Diagramma a blocchi di alto livello che descrive l'integrato:

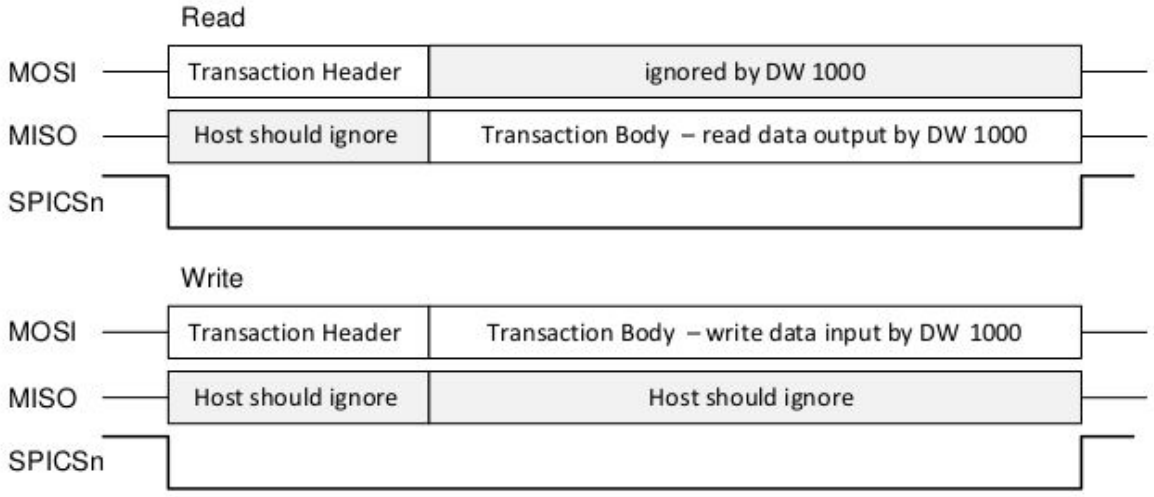


High Level Block Diagram

Overview

Interfacciamento al dispositivo

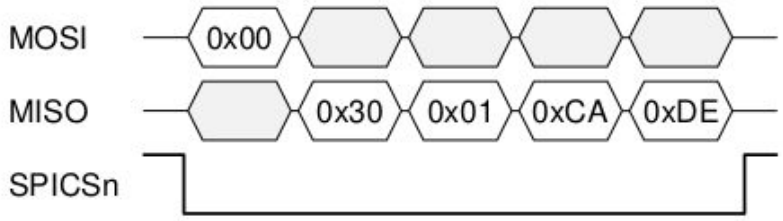
La comunicazione ad un dispositivo host da parte del DW1000 è di tipo slave-only SPI. Il sistema host legge e scrive i registri dell'IC tramite SPI.
Ogni transazione inizia con un header da 1 a 3 byte, seguito da un numero variabile di byte che formano i dati della transazione. Per le transazioni di lettura tutti gli ottetti oltre quelli dell'header sono ignorati dal DW1000, allo stesso modo tutti gli ottetti in output dal DW1000 sono ignorati dall'host durante la scrittura.



In base alla lunghezza dell'header si hanno tipi di transazioni diverse.

Transazione con header a 1 ottetto:

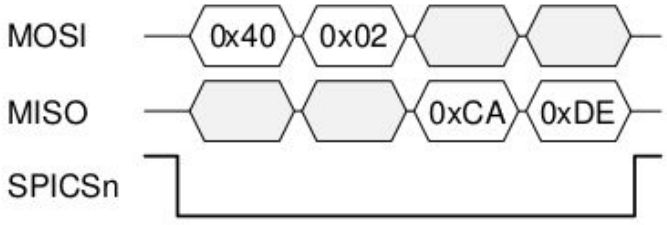
Bit number:	7	6	5	4	3	2	1	0	
Meaning:	Operation: 0 = Read 1 = Write	Bit = 0, says sub-index is not present	Register file ID – Range 0x00 to 0x3F (64 locations)						Transaction Header Octet



Non sub indexed read from register file ID 0x00, this 32-bit register contains the Device ID, the 4-octet result is 0xDECA0130

Transazione con header a 2 ottetti:

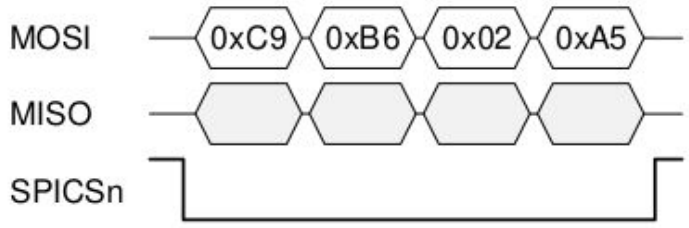
Bit number:	7	6	5	4	3	2	1	0	
Meaning:	Operation: 0 = Read 1 = Write	Bit = 1, says sub-index is present	Register file ID – Range 0x00 to 0x3F (64 locations)						Transaction Header Octet 1
	Extended Address: 0 = no	7-bit Register File sub-address, range 0x00 to 0x7F (128 byte locations)							Octet 2



Short sub-indexed read 2-octets beginning at index 2 in register file ID 0x00. This reads the high two octets of this 32-bit register, the result is 0xDECA.

Transazione con header a 3 ottetti:

Bit number:	7	6	5	4	3	2	1	0	
Meaning:	Operation: 0 = Read 1 = Write	Bit = 1, says sub-index is present	Register file ID – Range 0x00 to 0x3F (64 locations)						Transaction Header Octet 1
	Extended Address: 1 = yes	Low order 7 bits of 15-bit Register file sub-address range 0x0000 to 0x7FFF (32768 byte locations)							Octet 2
	High order 8 bits of 15-bit Register file sub-address range 0x0000 to 0x7FFF (32768 byte locations)								Octet 3



Long sub indexed write of octet value 0xA5 to the transmit buffer (Reg ID 0x09) at index 310 (0x136 in hex)

Il DW1000 può essere configurato in modo che al verificarsi di eventi di stato interni, esso utilizzi il pin IRQ per creare degli interrupt.

Sono presenti anche 8 pin GPIO (General Purpose Input Output), configurabili anch'essi tramite SPI.

Stati operazionali

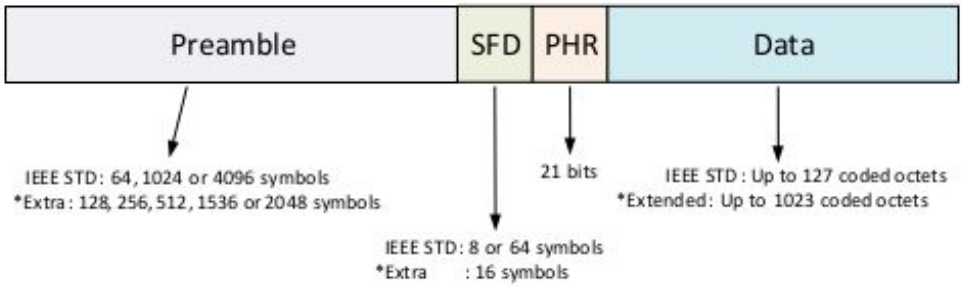
Il DW1000 ha un diverso numero di stati operazionali:

- OFF - Il dispositivo è completamente spento.
- WAKEUP - Periodo di avviamento del dispositivo, dopo 4 ms si passa allo stato INIT del dispositivo.
- INIT - Periodo di stabilizzazione del clock interno per poi passare allo stato IDLE del dispositivo.
- IDLE - In questo stato l'SPI può funzionare fino a 20 MHz, in questo stato si è pronti a ricevere comandi per poi entrare in ricezione o trasmissione.
- TX - Stato durante il quale il dispositivo si trova in trasmissione.
- RX - Stato durante il quale il dispositivo si trova in ricezione.
- SLEEP - In questo stato il circuito alimenta un clock interno per il conteggio del tempo di wake-up (che deve essere programmato).
- DEEPSLEEP - Lo stato con i consumi più bassi (escluso OFF), in questo stato solo l'AON (Always On Memory) è alimentata, rimane in questo stato fino ad un evento di wakeup (linea SPICSn a LOW oppure la linea WAKEUP ad HIGH) dopo il quale si verifica una transizione verso lo stato WAKEUP.

- SNOOZE - Simile all'INIT con la differenza ad eccetto che lo stato successivo è l'RX.

Trasmissione

Il dispositivo parte in stato IDLE aspettando istruzioni dall'host.
Per trasmettere l'host deve scrivere i dati nel buffer di trasmissione del dispositivo e le necessarie configurazioni di trasmissione nel registro di controllo della trasmissione dei frame.
L'host inizia la trasmissione scrivendo il bit TXSTRT nel registro di controllo del sistema. Dopo la richiesta di trasmissione il DW1000 invierà il frame completo, il CRC per il controllo dell'integrità viene aggiunto automaticamente alla fine del messaggio.
Il device prima di trasmettere i dati, a livello fisico trasmette un preambolo, un delimitatore di inizio del data frame e un header fisico, come rappresentato dalla seguente figura:



La trasmissione può essere immediata o differita (inserendo un tempo UWB) e deve essere selezionata prima di ogni singola trasmissione tramite appositi registri di sistema.
La trasmissione differita è soprattutto utilizzata nel Double-sided Two-Way Ranging.
La lunghezza massima dei dati trasmessi secondo lo standard IEEE 802.15.4 può essere di 127 bytes, il dispositivo però permette anche di estendere la lunghezza dei frame trasmessi fino ad un massimo di 1023 bytes, in questo caso il PHY header viene esteso di 3 bit.

Ricezione

La ricezione di un frame è abilitata da una richiesta dell'host o dalla riattivazione automatica del ricevitore (Quest'ultima tramite apposita preconfigurazione). Il

ricevitore cercherà di continuo per un preambolo per poi iniziare la demodulazione.
Le fasi della ricezione sono:

- Individuazione del preambolo
- Accumulazione del preambolo
- Individuazione del delimitatore di inizio frame (SFD)
- Demodulazione dell'header fisico (PHR)
- Demodulazione dei dati

Anche la ricezione può essere svolta in differita configurando gli appositi registri.

Timestamp

Durante la trasmissione e la ricezione vengono salvati all'interno del dispositivo, in particolari registri, dei timestamp.
Nel caso della trasmissione questo avviene nell'istante in cui viene trasmesso il primo simbolo del PHR, questo è definito come *RMARKER*.
Per quanto riguarda la ricezione invece lo RMARKER avviene alla fine della individuazione del delimitatore di inizio frame.
I timestamp sono molto importanti per quanto riguarda la localizzazione poiché sono proprio essi che permettono un'accurata misurazione dei range tra i dispositivi, per quanto riguardo il Two Way Ranging, o utilizzati direttamente per il calcolo della localizzazione quando si utilizza l'algoritmo TDoA (poichè si basa appunto sulla differenza dei tempi di ricezione di vari dispositivi).
L'unità dei timestamp, all'interno del DW1000, corrisponde all'incirca a 15.65 ps.

Funzionalità aggiuntive a livello MAC

Il DW1000 oltre ad implementare il livello fisico per la modulazione e trasmissione dei dati è fornito di un livello media access control che usa per effettuare controlli e ricezione o trasmissioni automatiche.

Cyclic Redundancy Check (CRC)

L'IC include una funzione di generazione CRC in grado di calcolare automaticamente e accodare 16 bits di CRC frame check sequence (FCS) alla fine di ogni frame.
E' anche inclusa la funzione di check automatico dei 16 bits CRC durante la ricezione del frame. La verifica del CRC avviene mettendo a confronto i 2 byte ricevuti con quelli calcolati in ricezione dalla funzione. La non uguaglianza tra il CRC ricevuto e quello calcolato indica che il frame di ricezione contiene errori.

Frame Filtering

E’ una funzionalità che permette di analizzare i dati ricevuti e di controllare che siano coerenti con lo standard 802.15.4, identificando il tipo di frame e il destinatario.
Le regole del frame filtering sono le seguenti:

- Regole configurabili:
    - Accettazione dei frame Beacon
    - Accettazione dei data frame
    - Accettazione degli acknowledgment frame
    - Accettazione dei MAC command frame
    - Accettazione dei tipi riservati 802.15.4 numeri da 4 a 7
    - Accettazione del tipo riservato 802.15.4 numero 4
    - Accettazione del tipo riservato 802.15.4 numero 5
- La versione del frame deve essere 0x00 o 0x01
- L’indirizzo PAN ID, se presente, deve essere:
    - 0xFFFF
    - Oppure il PAN ID programmato all’interno dell’apposito registro
- L’indirizzo di destinazione, se presente, deve essere:
    - OxFFFF
    - Oppure l’indirizzo programmato all’interno dell’apposito registro
    - Oppure l’indirizzo con formato lungo Extended Unique Identifier (EUI) programmato all’interno dell’apposito registro
- Se il frame è di tipo Beacon il PAN ID sorgente deve corrispondere a quello programmato all’interno del device
- Il CRC deve essere valido

Automatic Acknowledgement

Questa funzionalità permette di inviare automaticamente un acknowledgment frame dopo che un frame con richiesta di acknowledgement è stato ricevuto.
Per funzionare deve essere attivo il frame filtering.

Set di registri

Il DW1000 è controllato dal microcontrollore host tramite interfaccia SPI.
I registri (64 in totale) inclusi nel dispositivo sono di configurazione, di stato, controllo , buffer dati e diagnostica.
Alcuni di questi registri sono di tipo Read/Write, altri soltanto di tipo Read con alcune eccezioni di registri riservati per utilizzo interno.

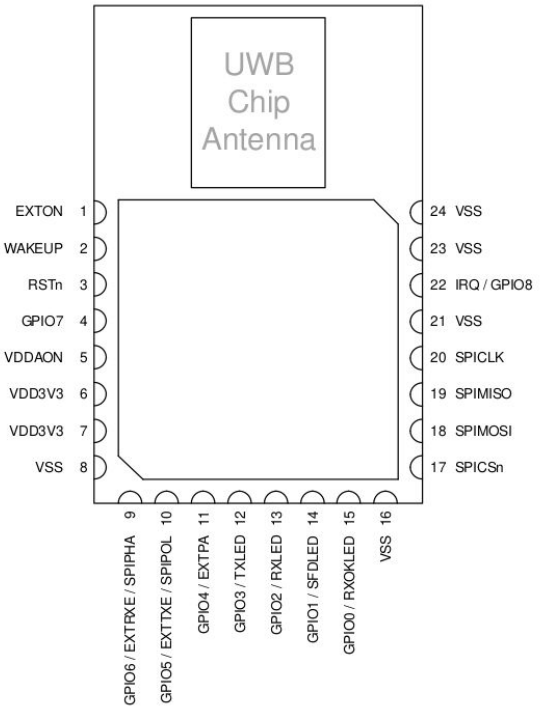
Prototipo

DWM1000

Il DWM1000[5] è un modulo integrato che fornisce un circuito esterno al DW1000 già pronto, questo circuito include:

- Un'antenna integrata e relativo circuito a radiofrequenza
- Un clock on-board fornito da un cristallo a 38.4 MHz, già calibrato in produzione per ottenere un errore di frequenza approssimativamente di 2 ppm
- Circuito di alimentazione

Il DWM1000 è distribuito con un formato package più semplice per la costruzione di prototipi. Questo ci ha permesso di velocizzare il processo di prototipazione. E' stata, però, esclusa la possibilità di inserire un clock esterno per la sincronizzazione necessaria per il TDoA (non abbiamo problemi in questo caso dato l'utilizzo del 2-way ranging).



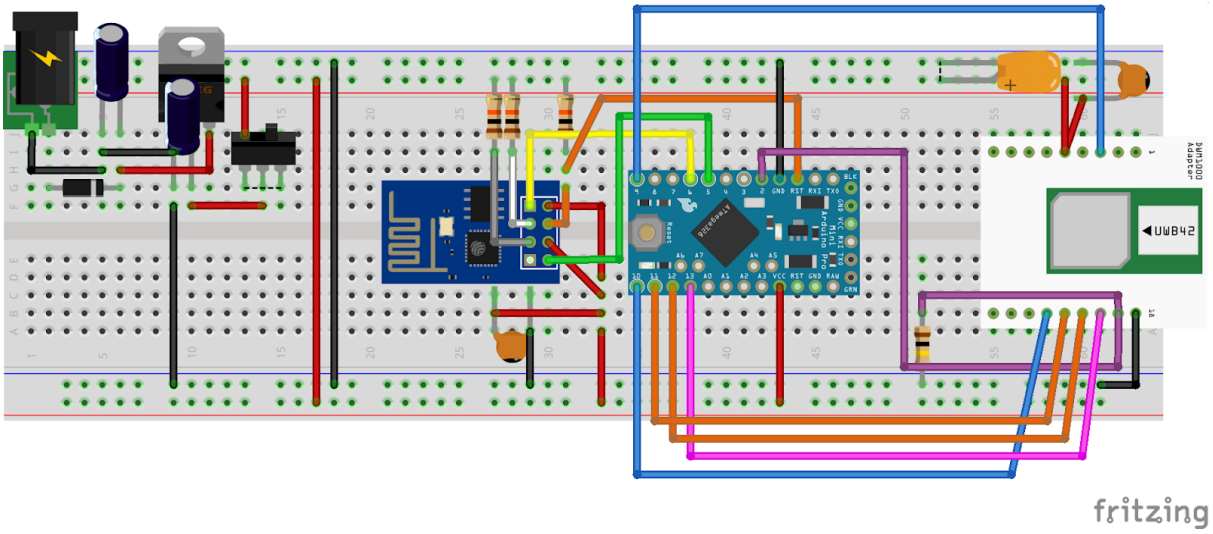
Il DWM1000 è stato saldato su un adattatore pcb per ricavare dei collegamenti facili da utilizzare con breadboard o basetta millefori.

[5] "DWM1000 Datasheet - Decawave." https://www.decawave.com/sites/default/files/dwm1000-datasheet.pdf. Accessed 15 Jan. 2019.

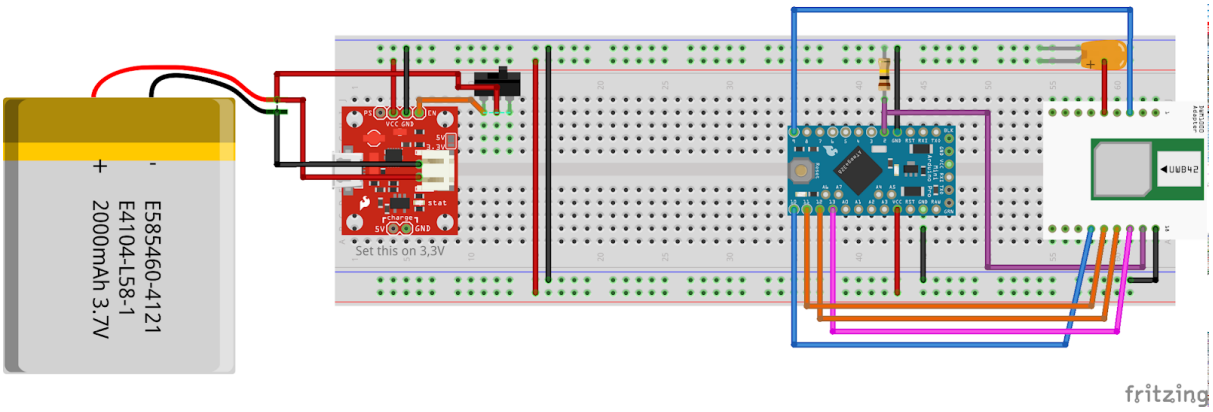
Arduino

Una tipologia di prototipo è stato realizzato utilizzando come microcontrollore host un Arduino Pro Mini a 3.3v con oscillatore a 8MHz.
Per quanto riguarda il DW1000, abbiamo dunque utilizzato il modulo integrato fornito dalla stessa casa madre DWM1000.
All'inizio i prototipi sono stati realizzati con breadboard, successivamente sono stati saldati su scheda millefori per una maneggevolezza maggiore.
La scheda Arduino Pro Mini non fornisce uscite di corrente in grado di alimentare un modulo DWM1000, per questo al prototipo è stata aggiunta circuiteria di alimentazione dedicata in grado di soddisfare entrambi.

Schema circuitale delle ancore:



Circuito del tag:



ESP8266

Il prototipo basato su Arduino essendo poco performante, rispetto alle MCU consigliate dalla Decawave, diminuisce la precisione e la stabilità dei range.
La documentazione ufficiale del modulo consiglia di usare una MCU più veloce possibile ed una frequenza SPI massima (che si attesta a 20 Mhz) al fine di aumentare le prestazioni e ridurre il consumo energetico.
Si è dunque deciso, successivamente ad un periodo di test con Arduino, di utilizzare moduli ESP8266. Questa scelta è stata dettata dalla velocità di sviluppo (essendo compatibile con l'ecosistema Arduino). In questo modo abbiamo ottenuto prestazioni uguali a quelle offerte da altre MCU professionali (quindi di molto superiori all'Arduino Pro Mini) , con l'aggiunta del grande vantaggio di integrare una connettività WiFi.
Nello specifico abbiamo utilizzato il modulo NodeMCU 1.0 . Il modulo è stato collegato al DWM1000 nel seguente modo:

DWM1000		ESP8266
RST	→	GPIO 5
IRQ	→	GPIO 4
CLK	→	D5
MISO	→	D6
MOSI	→	D7
CS	→	D8
3.3	→	3V3
GND	→	GND

Un ulteriore vantaggio è dato dal fatto di non avere bisogno di circuiti supplementari di alimentazione, in quanto è già dotato di un regolatore di tensione con uscita 3,3 Volt con sufficiente amperaggio.

Considerazioni sui prototipi

I prototipi in questione sono stati creati in modo minimale per creare un Proof Of Concept. In una situazione di produzione potrebbero essere aggiunti dispositivi di ausilio come un altimetro ed un accelerometro. Rispettivamente per rimuovere l'errore dovuto alla differenza di altezza tra ancore e tag e per venire a conoscenza di quando il tag rimane fermo così da ridurre ancora di più gli elevati consumi dovuti alla trasmissione e ricezione utilizzati dal two way ranging, interrompendoli. Questo può aumentare l'autonomia da qualche decina di ora a qualche giorno/settimana.
I consumi dei prototipi si aggirano mediamente a 210mA/h per le ancore e 80mA/h per i tag in pieno funzionamento durante la localizzazione.

Driver

La nostra intenzione iniziale era di sviluppare fin da subito un sistema di localizzazione e durante la fase di ricerca abbiamo trovato il seguente repository[6] con una libreria arduino per utilizzare i moduli basati su DW1000 :
https://github.com/thotro/arduino-dw1000

Testando poi il codice attraverso i suoi esempi abbiamo riscontrato un'imprecisione nel calcolo delle distanze e quindi siamo passati ad analizzare il codice.
Durante l'analisi abbiamo notato che molte feature discusse nel manuale utente del modulo non erano presenti o se presenti erano incomplete e/o affette da bug.
Molte di queste funzionalità non funzionanti o mancanti erano necessarie per la realizzazione del progetto.
Oltretutto il codice non era di facile comprensione a causa di un'evidente assenza di progettazione di un'architettura. A complicare la lettura e la comprensione partecipavano la natura di basso livello e soprattutto alcune funzioni che presentavano dei side effects. Inoltre l'astrazione utile ad effettuare un processo di ranging era stata implementata attraverso un loop e quindi gestibile soltanto negli eventi che la libreria provvedeva.
Abbiamo deciso quindi di riscrivere in parte tutte le sezioni che potevano creare problemi alla logica dei futuri programmi basati sul driver e di implementare le funzioni mancanti e correggere quelle con bugs, questo lavoro ha portato alla riscrittura quasi totale del driver di base.
Dunque è stato creato un fork[7] poiché alcuni cambiamenti, soprattutto per astrazioni di più alto livello non coincidevano con la filosofia del repository di origine. Abbiamo però contribuito con varie pull request a risolvere alcuni problemi di esso.
Abbiamo deciso di usare un approccio di più basso livello ma che al contempo aiutasse nello sviluppo e permettesse di avere un maggiore controllo durante lo scambio dei messaggi del Two Way Ranging.
Altri maggiori cambiamenti riguardano appunto le API del driver, rimaste simili al repository originale, ma le sue funzioni sono state rese molto più indipendenti tra di loro poiché all'inizio era sottointeso un ordine nelle loro chiamate.
Per di più il repository di origine non è completamente abbandonato ma comunque risente di uno sviluppo molto lento poiché l'autore principale ha smesso di svilupparlo da più di un anno e il repo viene soltanto mantenuto.

Andremo di seguito ad analizzare i principali cambiamenti apportati.

---

[6] "GitHub - thotro/arduino-dw1000: A library that offers functionality to ...." https://github.com/thotro/arduino-dw1000. Accessed 15 Jan. 2019.
[7] "GitHub - F-Army/arduino-dw1000-ng: Arduino driver and library to use ...." https://github.com/F-Army/arduino-dw1000-ng. Accessed 15 Jan. 2019.

Migliorie a livello di codice

Uno dei primi dei primi step è stato un refactor del codice di base del driver.
Il codice presentava delle cattive pratiche nella sezione principale del driver, il tutto era implementato come una classe con tutti i suoi elementi dichiarati static e poi inizializzata all'interno dei file di codice stessi.
Il motivo di questa scelta era facilmente deducibile, l'autore originale voleva creare una sintassi molto simile ad altre librerie di Arduino come ad esempio quella per l'SPI. Questo di per sé non sarebbe un problema ma molte delle funzioni che sarebbero dovute essere usate all'interno dell'implementazione dell'interfaccia erano state dichiarate pubbliche, come anche alcune variabili di appoggio interne. Per gli utilizzatori e per gli sviluppatori, questo è causa di confusione e di errori di programmazione dovute alla mancata differenziazione tra l'astrazione dell'API pubblica e la sua effettiva implementazione. Essendo Arduino basato sul linguaggio C++ abbiamo pensato non di utilizzare una classe, ma un namespace per le funzioni riguardanti il driver. L'unica differenza nell'utilizzo è quella che riguarda come richiamare le funzioni, ad esempio per entrare in ricezione la chiamata da effettuare era DW1000.startReceive(), con l'utilizzo dei namespace (supponendo di averne dichiarato uno col nome DW1000) diventa DW1000::startReceive().
L'utilizzo del namespace ci ha permesso anche di utilizzare un altro tipo di namespace nei file di implementazione.Il namespace in questione è denominato 'anonimo' e infatti viene dichiarato in C++ senza attribuirgli nessun nome e ciò permette di raggruppare tutto ciò che effettivamente è da considerare privato al suo interno.
Questo ha portato a un file header più snello e con soltanto la dichiarazione dell'API pubblica. Avendo le funzioni e le variabili (usate per lo storage di valori e configurazioni) all'interno del namespace anonimo, evitiamo che esse vengano modificate da qualche chiamata erronea inclusa in file sorgenti non appartenenti al diretto controllo low level del device, infatti ora il programma non compilerà più se vengono effettuati riferimenti ad essi se non all'interno dello stesso file.
Molte funzioni sono state anche scomposte in funzioni private più piccole per una maggiore comprensibilità e riutilizzabilità, ad esempio una funzione utilizzata nell' ottimizzazione di alcuni parametri di funzionamento interni dopo aver caricato la configurazione del dispositivo (che precedentemente era lunga più di 300 righe di codice) ora è stata scomposta in molte sottofunzioni relative ai singoli parametri su cui effettuare l'ottimizzazione, questo ha permesso anche di utilizzarle in altre funzioni non connesse direttamente a questa funzionalità, rendendole funzioni indipendenti dal resto del codice, creando così una maggiore modularità e riutilizzabilità dello stesso.

In generale abbiamo preferito utilizzare idiomi del linguaggio C++ a sostituzione delle loro controparti legacy inserite per retrocompatibilità con C. Una di queste è la direttiva #define per l'utilizzo di costanti, che in C++ è soppiantato da const e constexpr, poiché permettono di definire il tipo della costante ed evitare spiacevoli conseguenze a livello semantico.
Un altro idioma utilizzato del linguaggio C++ riguarda il cosiddetto enum class, un'evoluzione del classico enum del linguaggio C, ma staticamente tipizzato.
Questo porta a un codice leggermente più verboso, ma type safe, andando così ad aumentare la facilità di modifica del codice.
Le funzioni dell'API all'interno del file header sono state commentate per facilitare chiunque volesse sviluppare o utilizzare il driver.
Infine alcune funzioni che non erano direttamente collegate al driver, sono state spostate in altri file, come ad esempio le funzioni ausiliarie per la manipolazione di byte.
Il nuovo codice, che è stato esteso nelle funzionalità, risulta comunque più snello della vecchia versione, in alcuni casi fino ad un 20% in meno per quanto riguarda la memoria flash e dinamica occupata.

Funzionalità migliorate e corrette

Come precedentemente detto, abbiamo provveduto alla miglioria e correzione di alcune funzioni dell'API del codice originale.
La modifica più sostanziale è stata eseguita su alcune funzioni riguardanti la configurazione del dispositivo, il frame filtering e la configurazione degli interrupt in base agli eventi. In tutti e 3 i casi, le singole funzioni riguardanti la configurazione di questi campi sono state raggruppate in una singola funzione che ha come input una struttura che va a definire, per l'appunto, la configurazione. Questo rende il codice più facilmente utilizzabile poiché la funzionalità è raggruppata in una singola funzione dell' API e non più sparpagliata nel codice. Nel caso della configurazione del dispositivo questo ci ha permesso anche di raggruppare tutti i parametri che hanno bisogno di un processo definito di *tuning*, e cioè che vanno a modificare indirettamente altri parametri interni del dispositivo, che se non eseguito può comportare malfunzionamenti. Precedentemente questa logica era suddivisa in varie funzioni che dovevano essere chiamate prima di un'altra funzione nominata 'commitConfiguration()' dove veniva per l'appunto effettuato questo tuning e, naturalmente, poteva portare a situazioni di mal configurazione poiché questa era una logica nascosta.
Per quanto riguarda il frame filtering abbiamo anche aggiunto due parametri mancanti nella possibile configurazione, precisamente quelli riguardanti i frame di tipo riservato 4 e 5 (discussi nella precedente sezione del frame filtering).
Le altre funzionalità migliorate sono queste:

- *Parte del codice relativa all'inizializzazione del dispositivo*: alcune righe riguardanti il salvataggio di alcune configurazioni di startup del dispositivo, sono state inserite con una lettura diretta da esso, così da renderle esenti da errori e compatibili con future revisioni dell'integrato. Alcune funzioni riguardanti il caricamento del microcode e le letture dei valori di riferimento di voltaggio e temperatura per le successive rilevazioni sono state spostate di ordine poiché devono essere eseguite necessariamente utilizzando una particolare variazione di frequenza del clock interna (XTI). Infine è stata aggiunta una porzione di codice relativa alla pulizia della memoria AON, necessaria per la funzionalità di deep sleep.
- *Selezione SPI*: Aggiunte delle direttive per compatibilità con i dispositivi ESP8266 e ESP32.
- *Funzione di gestione degli eventi di stato:* sono stati corretti i vari registri che vengono resettati in base agli eventi, questo ha portato alla risoluzione di alcuni bug relativi al ritorno in ricezione.
- *Funzione deep sleep*: La funzione di deep sleep nel codice originale non salvava correttamente tutti i parametri nell' AON e non ripristinava il delay di trasmissione dell'antenna.
- *Funzione del reset software:* Ora viene attivato il clock XTI prima di effettuare la scrittura dei vari registri riguardo il power management ed è stato aggiunto il clear ad alcuni registri dell'AON per via di alcune problematiche hardware note che potrebbero verificarsi allo startup del dispositivo.
- *Funzione relativa alla forzatura dello spegnimento della ricetrasmittente:* inserito un nome più appropriato alla funzione, da forceIdle() a forceTRXoff()
- *Rimossa una variabile che tiene traccia dello stato interno del dispositivo:* non veniva usata se non per una funzione che è stata rimossa poichè piena di side effects, discussa nel prossimo punto.
- *Rimossa la funzione permanentReceive:* questa funzione era stata inserita dall'autore originale ma era puramente software e non rispecchiava nessun funzionamento hardware, per di più andava in conflitto con varie funzionalità non ancora implementate che sarebbero state difficili da usare assieme.
- *Aggiunte modalità di ricezione e trasmissione in differita:* questa funzionalità era presente ma veniva effettuata indirettamente alla specifica del delay stesso, questo significa che la funzione di delay doveva essere ripetuta ogni qualvolta si volesse usare la modalità differita, ora invece basta specificare il tempo una volta sola e poi richiamare le funzioni di trasmissione o ricezione con l'argomento TransmitMode o ReceiveMode impostato a DELAYED.
- *Aggiunti controlli alla corretta validità della configurazione del dispositivo*
- *Eliminato l'oggetto DW1000Time:* questo oggetto seppur utile per la manipolazione dei tempi del dispositivo veniva usato effettivamente solo per convertire i tempi in un valore con unità di misura interna del DW1000. Questa funzionalità è rimasta, ma è stata convertita semplicemente in una funzione,

questo ha comportato una diminuzione sostanziale dello spazio utilizzato dal driver.
- *Creata funzione riguardante la correzione del bias del range:* questa funzione veniva erroneamente eseguita ad ogni passaggio del two way ranging all'interno degli esempi, ora invece è una funzione richiamabile sul range stesso appena eseguito.

Funzionalità aggiunte

Sono anche state aggiunte nuove funzionalità molto utili per la creazione di software performante e a basso consumo energetico:

- *Aggiunti getters per quanto riguarda alcuni parametri che prima erano pubblici*
- *Aggiunte varie funzioni relative alla regolazione della potenza di trasmissione*: Ora è possibile variare la potenza di trasmissione del dispositivo, questo permette di regolare la trasmissione nei limiti di legge di utilizzo in base all'antenna utilizzata.
- *Aggiunte funzioni per la calibrazione dell'antenna:* queste funzioni vanno a cambiare i timestamp di trasmissione e/o ricezione del dispositivo per compensare i delay hardware.
- *Aggiunta opzione di compilazione con ottimizzazione della potenza per il modulo DWM1000:* Attivando il flag di compilazione DWM1000_OPTIMIZED la potenza automatica viene impostata al valore più alto consentito dalla legge in base alla configurazione dei vari parametri quando si utilizza il modulo DWM1000. Attenzione attivarlo con un' altro modulo o circuito non compromette la compilazione, ma può portare a valori non conformi alle norme relativi alla potenza di trasmissione.
- *Aggiunta modalità di test per misurazione dello spettro della potenza:* attivando questa modalità il dispositivo trasmette di continuo per un intervallo precisato, così da permettere una facile misurazione dello spettro del segnale trasmesso e dunque calibrarlo in base ai limiti di legge.
- *Aggiunte varie funzioni per impostare vari timeout in modalità ricezione*
- *Aggiunte funzione singole per ottenere i valori di temperatura e voltaggio dell'alimentazione*
- *Aggiunta funzione per disabilitare il frame filtering*
- *Aggiunta funzione per abilitare il wait for response:* Il dispositivo si rimette in ricezione dopo un delay specificato successivamente a una trasmissione.
- *Aggiunto un parametro di configurazione per migliorare la ricezione in condizioni NLOS:* questo parametro modifica internamente la sensibilità di ricezione per ottimizzarla quando la trasmissione non avviene in linea d'aria.

- *Creato un header file con costanti riguardanti lo standard RTLS ISO/IEC 24730:62-2013*

Astrazioni per il two way ranging

Successivamente è stato svolto un lavoro di semplificazione per l'utilizzo dell'algoritmo di two way ranging in favore di una maggiore semplicità di utilizzo da parte degli utenti meno esperti e di quelli che non hanno a disposizione una copia dello standard riguardante la comunicazione.
I file aggiunti per l'astrazione sono appunto basati sugli standard RTLS e si trovano nei file denominati DW1000NgRTLS.hpp e DW1000NgRTLS.cpp.
Questi file aggiungono comode macro che astraggono i valori di basso livello usati nella formattazione dei frame inviati dai dispositivi e delle funzioni per astrarre la logica di two way ranging.
E' possibile adesso eseguire un flow completo di two way ranging utilizzando poche funzioni a seconda dello step in cui ci si trova durante la localizzazione o di quanta astrazione si ha bisogno.
Il modo più immediato per ottenere il range è eseguire l'invocazione da parte del tag della funzione DW1000NgRTLS::tagTwrLocalize. Il dispositivo che ha il compito di ascoltare i blink del tag utilizza invece DW1000NgRTLS::transmitRangeInitiation seguito da DW1000NgRTLS::AnchorRangeAccept, quest'ultimo utilizzato poi anche dalle ancore successive utilizzate per la ricerca dei range utili al fine di localizzare il tag.
Queste funzioni restituiscono delle strutture che permettono di analizzare il successo o il fallimento delle varie richieste.
Altre funzioni create sono le seguenti (alcune utilizzate internamente dalle precedenti):

- increaseSequenceNumber: incrementa il numero di sequenza interno, utile se lo sviluppatore invia dei messaggi senza usare le astrazioni
- transmitTwrShortBlink: invia un blink minimale per l'utilizzo nel Two Way Ranging, utilizzato nei tag
- transmitPoll: trasmette il messaggio della sequenza Two Way Ranging, utilizzato nei tag
- transmitResponseToPoll: trasmette la risposta al poll inviato con la funzione precedente, utilizzato dalle ancore
- transmitFinalMessage: trasmette il messaggio finale contenente i tempi utilizzabili per il calcolo del range, inviato dal tag
- transmitRangingConfirm: trasmesso dall'ancora come messaggio finale del flow Two Way Ranging ed indica al tag una nuova ancora per eseguire un altro range partendo direttamente dal poll

- transmitActivityFinished: trasmesso dall'ancora in alternativa al Ranging Confirm nel caso si voglia comunicare al tag di ritornare ad eseguire dei blink potendo specificare il nuovo blink rate.
- receiveFrame: semplice astrazione della messa in ricezione del dispositivo senza la necessità di controllare i registri interni per l'avvenuta ricezione
- waitForTransmission: utilizzato per interrompere il flusso del microcontrollore in attesa dell'invio di un messaggio da parte del DW1000
- tagRangeRequest: utilizzato dal tag per notificare una struttura della possibilità di eseguire il ranging.
- tagRangeInfrastructure: utilizzato dal tag per iniziare il flow di ranging dopo aver ricevuto l'approvazione attraverso una Range Initiation.

Nei casi in cui si utilizzino dei cicli while, ad esempio nella funzione waitForTransmission, sono state inserite delle righe di codice con clausola di compilazione per i dispositivi ESP8266, poichè essi hanno bisogno di eseguire una chiamata al dispositivo watchdog interno per l'esecuzione di routine per il corretto funzionamento della componente wireless, tutto ciò può essere eseguito con una semplice chiamata alla funzione yield() della libreria Arduino specifica per l'ESP8266.

Licenza

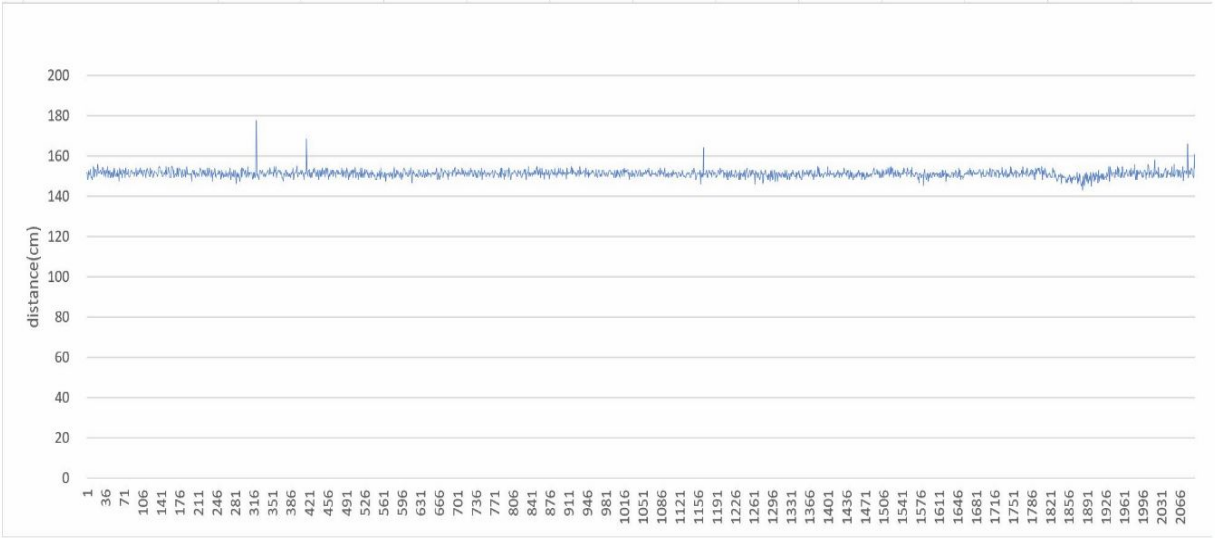
Il nuovo driver è stato pubblicato sotto licenza MIT, alcuni file pertanto sono ora in licenza doppia MIT/Apache 2.0 poiché quest'ultima era la licenza originale del vecchio driver e per poterla cambiare sono necessarie le approvazioni di tutti i contributori.

Sketch e esempi di utilizzo del driver

Arrivati a una versione utilizzabile del nuovo driver, abbiamo creato vari sketches arduino per testare e mostrare l'utilizzo del nuovo codice.

Sketch ranging

Per prima cosa abbiamo riadattato due vecchi esempi di ranging presenti nel repository originale, i quali sono stati usati durante la codifica del driver come test per il corretto funzionamento di base.
Questi esempi però non rispecchiano un utilizzo comune del Two Way Ranging, infatti usavano un protocollo personale del vecchio autore e non rispetta gli standard di localizzazione. Partiti da questi esempi abbiamo prima ricreato il Two Way Ranging con messaggi propri di un sistema di localizzazione standard ISO/IEC 24730-62:2013.
Grazie a queste modifiche abbiamo anche indirettamente verificato il corretto funzionamento della parte di frame filtering che era già stata implementata ed era funzionante ma probabilmente l'autore non era riuscito ad usarla poiché i vecchi messaggi non seguivano il formato 802.15.4 e quindi tutti i frame venivano scartati automaticamente dall'hardware.
Abbiamo notato in generale un miglioramento delle capacità di range e, calibrando il delay dell'antenna per entrambi i dispositivi, la precisione è aumentata ulteriormente. Tutto ciò è stato confermato anche da utilizzatori esterni che ci hanno gentilmente fornito dei feedback.
Di seguito riportiamo dei dati dei test svolti sugli sketch di ranging, svolti in un ambiente privo di disturbi e posizionando i device in polarizzazione verticale(consigliato dagli application notes forniti dalla casa madre) ad una distanza di 150 cm, ottenendo così il miglior risultato possibile:



	A	B	C
37	Range:	151	cm
38	Range:	151	cm
39	Range:	152	cm
40	Range:	153	cm
41	Range:	153	cm
42	Range:	151	cm
43	Range:	152	cm
44	Range:	151	cm
45	Range:	154	cm
46	Range:	154	cm
47	Range:	150	cm
48	Range:	153	cm
49	Range:	152	cm
50	Range:	150	cm
51	Range:	153	cm
52	Range:	151	cm
53	Range:	152	cm
54	Range:	155	cm
55	Range:	149	cm
56	Range:	152	cm
57	Range:	151	cm
58	Range:	153	cm
59	Range:	153	cm
60	Range:	149	cm
61	Range:	152	cm
62	Range:	149	cm
63	Range:	152	cm
64	Range:	149	cm
65	Range:	153	cm
66	Range:	150	cm
67	Range:	154	cm

Come possiamo notare abbiamo ottenuto risultati con una precisione che si attesta intorno ai 5 cm, confermando ciò che dichiara il produttore del dispositivo.
Come si può notare nella prima immagine, si sono verificati sporadici calcoli errati del range probabilmente dovuto a fattori interni dell'hardware o esterni di interferenza elettromagnetica, questo non è un problema poiché si possono adottare delle strategie di filtraggio software in ambienti di produzione.

Sketch RTLS

Partendo dagli sketch del ranging siamo passati poi alla creazione di sketch relativi alla localizzazione di un tag in un sistema composto da 1 tag e 3 ancore.
Per prima cosa è stato attivato il frame filtering per facilitare la comunicazione con più dispositivi ed evitare molte righe di codice riguardanti i controlli sui pacchetti ricevuti.
E' stata in seguito aggiunta una logica di 'next anchor' all'ancora master che permette di restituire dinamicamente l'indirizzo della prossima ancora con cui far effettuare il ranging al tag.
Sono state poi creati gli sketch di altre 2 ancore slave, queste sono le due ancore che effettuano il ranging successivamente a quella master e fanno un report sul valore calcolato ad essa. L'ancora B fornisce l'indirizzo della successiva ancora C, quest'ultima invece restituisce al tag un messaggio per farlo ritornare in modalità di blink specificando un blink rate.
L'ancora master ha al suo interno conoscenza della sua posizione e quella delle ancore slave in un piano cartesiano ed utilizza i range ricevuti per calcolare la posizione del tag.
La formula utilizzata è stata ricavata semplificando un sistema di equazione di tre circonferenze generiche:

$$(x-x_1)^2+(y-y_1)^2=r_1^2$$
$$(x-x_2)^2+(y-y_2)^2=r_2^2$$
$$(x-x_3)^2+(y-y_3)^2=r_3^2$$

Le quali espandendo i quadrati diventano:

$$x^2-2x_1x+x_1^2+y^2-2y_1y+y_1^2=r_1^2$$
$$x^2-2x_2x+x_2^2+y^2-2y_2y+y_2^2=r_2^2$$
$$x^2-2x_3x+x_3^2+y^2-2y_3y+y_3^2=r_3^2$$

Sottraendo la seconda equazione dalla prima otteniamo:

$$(-2x_1+2x_2)x+(-2y_1+2y_2)y=r_1^2-r_2^2-x_1^2+x_2^2-y_1^2+y_2^2.$$

Sottraendo la terza equazione dalla seconda si ottiene:

$$(-2x_2 + 2x_3)x + (-2y_2 + 2y_3)y = r_2^2 - r_3^2 - x_2^2 + x_3^2 - y_2^2 + y_3^2.$$

Questo è un sistema di due equazioni a due incognite:

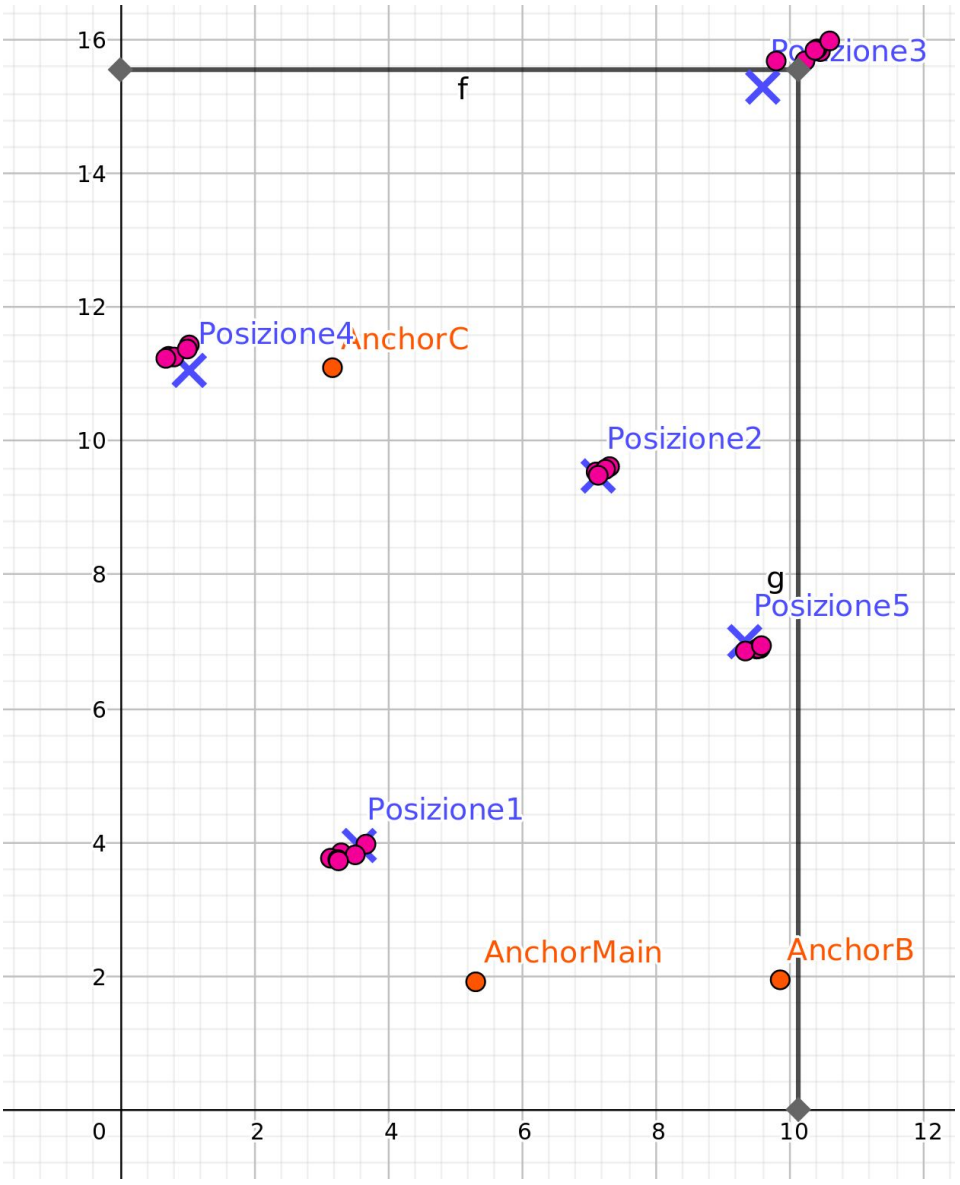
$$Ax + By = C$$
$$Dx + Ey = F$$

Che ha come soluzione:

$$x = \frac{CE - FB}{EA - BD}$$
$$y = \frac{CD - AF}{BD - AE}$$

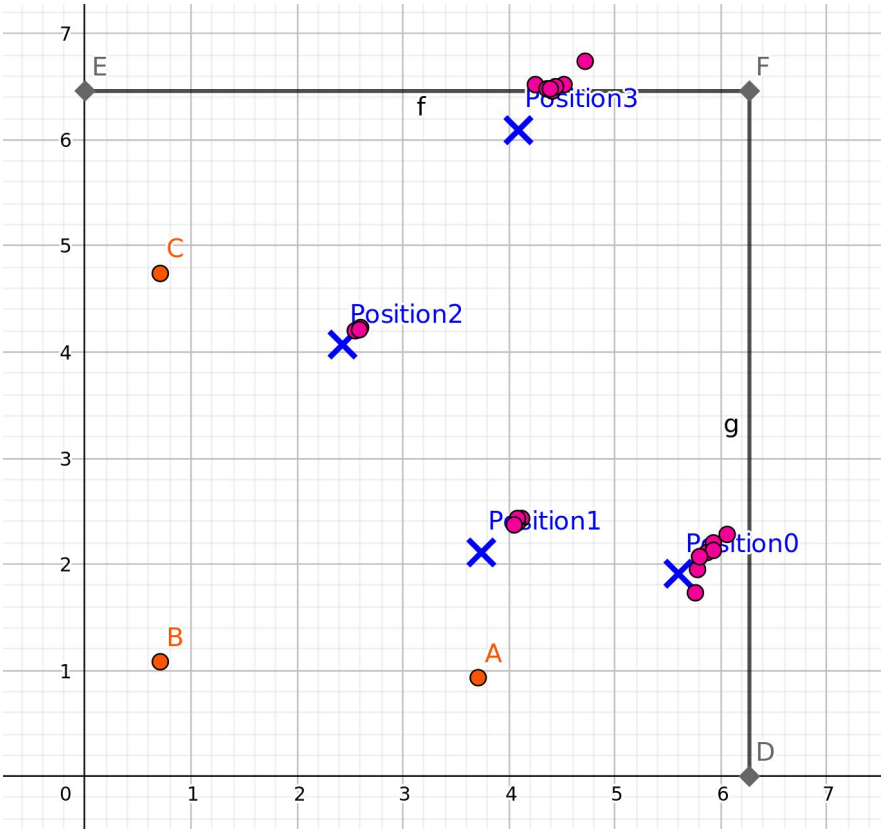
Ecco di seguito i plot dei test effettuati:

Aula A1



Dall'immagine possiamo notare come le posizioni siano abbastanza accurate, e non presentano particolari errori degni di nota se non nella posizione 3, questo è dovuto al fatto che le misurazioni in quella posizione sono state rilevate ad un asse z diverso che ha portato dunque a uno spostamento della posizione. Questi errori sono rimovibili attraverso l'introduzione di una quarta ancora per il calcolo della coordinata z o di un altimetro.

Aula Cisco



In questo caso, a parità di calibrazione, le posizioni sono comunque accettabili, ma presentano tutte uno shifting quasi costante, questo è probabilmente introdotto dalla troppa vicinanza tra le ancore, infatti la stessa casa produttrice dei moduli consiglia di posizionare le ancore a 10 metri di distanza l'una dall'altra.

Sketch Transmit Spectrum Power Calibration Test

Questo sketch è stato realizzato per facilitare i test della potenza di trasmissione, e attraverso un analizzatore di spettro è possibile stabilire i giusti valori per la potenza di trasmissione e la larghezza di banda rimanendo al di sotto dei limiti di legge.

Sketch Antenna Delay Calibration

Questo sketch è una semplice estensione di un'ancora per il Two Way Ranging, che modifica il suo delay dell'antenna fino ad ottenere un range molto vicino a quello programmato (che rappresenta il range reale misurato) e restituendo il giusto valore di delay dell'antenna.
Questo sketch è minimale ed è valido soltanto per il Two Way Ranging tra due dispositivi.

Licenza degli esempi

Gli esempi realizzati ex novo sono di dominio pubblico o Creative Commons 0 a scelta dell'utilizzatore, così da rimuovere ogni clausola di licenza.

Infrastruttura

Panoramica

Una componente fondamentale di un sistema RTLS è avere una solida infrastruttura per la gestione dei dispositivi ed il calcolo della posizione e la facile fruibilità dei dati raccolti per la creazione di vari software di supporto.
Un esempio è la creazione di un'interfaccia per la visualizzazione delle posizioni.
Per la creazione dell'infrastruttura per questo progetto abbiamo fatto riferimento agli standard ISO/IEC 24730-62:2013 e ISO/IEC 24730-1:2014[8] che riguardano rispettivamente la air interface, dove al suo interno è definito il concetto di infrastruttura, e l'interfaccia per la comunicazione con l'esterno dell'infrastruttura.
In sostanza il primo standard riguarda i meccanismi interni e il formato della comunicazione tra i dispositivi, il secondo standard su come interfacciarsi con il mondo esterno.

Air interface

Infrastruttura RTLS definita dagli standard

Lo standard ISO/IEC 24730-62:2013 intende per infrastruttura un'entità astratta rispetto ai semplici device, ed il tag non ne fa parte.
Il tag viene quindi visto anch'esso come un'entità a sé che comunica con questa infrastruttura e riceve delle risposte e si comporta di conseguenza.
Lo standard sopra citato in particolare definisce la comunicazione tra tag e infrastruttura e i possibili messaggi che possono essere scambiati in un sistema RTLS.
Alcuni di questi messaggi sono stati già analizzati precedentemente nel contesto del two way ranging. Andremo ora ad analizzare gli altri.

I messaggi dell'infrastruttura relativi al two way messaging sono identificati attraverso un codice di funzione inserito nel primo byte della componente variabile del messaggio.

[8] "ISO/IEC 24730-1:2014 - Information technology -- Real-time locating ...." https://www.iso.org/standard/59801.html. Accessed 15 Jan. 2019.

I codici di funzione sono i seguenti:

0x10 : Activity control from infrastructure
0x12 : Read tag capabilities – infrastructure command
0x13 : Response to read tag capabilities
0x14 : Read tag configuration – infrastructure command
0x15 : Response to read tag configuration
0x16 : Set tag configuration – infrastructure command
0x17 : Response to set tag configuration
0x19 : Error Response to set tag configuration
0x20 : Ranging initiation – infrastructure command
0x21 : Ranging Tag initial Poll response
0x23 : Ranging Tag final response message with embedded TX time
0x25 : Ranging Tag final response message with no embedded TX time

L'infrastruttura può quindi:

- Inviare attività di controllo
- Leggere le capacità supportate di un dispositivo tag
- Leggere la configurazione di un dispositivo tag
- Impostare la configurazione di un dispositivo tag
- Acconsentire il two way ranging

Attività di controllo

I messaggi di attività di controllo sono usati per comandare un continuo di un'attività durante le varie interazioni attraverso alcuni messaggi descritti nelle clausole successive, e sono anche usati per segnalare la fine di queste attività.

Un messaggio di attività di controllo è formato da 4 byte:

- 1 byte che rappresenta il codice di funzione fisso a 0x10, valore che rappresenta l'activity control.
- 1 byte che rappresenta il codice di attività
- 2 byte relativi ai parametri dell'attività

I codici di attività sono i seguenti:

0x00 : Activity Finished – Il tag deve tornare a eseguire dei blink, nei parametri di attività viene specificato il blink rate.
0x01 : Ranging Confirm – Usato dopo il two way ranging come conferma di un successo dello stesso, il parametro in questo caso rappresenta l'indirizzo del nuovo dispositivo con cui effettuare un'altro two way ranging.

0x02 : Ranging Continue – Usato per confermare la ricezione di un messaggio di poll (che fa parte del flow di two way ranging). Il parametro è ignorato in questo caso.

Lettura delle capacità

Il comando in questo caso non ha bisogno di parametri addizionali ed è formato semplicemente da un singolo byte di valore 0x12.
Il tag risponderà a questa richiesta con un messaggio di 5 byte composto da un byte del valore fisso di 0x13 e 4 byte con una maschera che rappresenta le capacità.

I bit della maschera sono i seguenti:

- 15 bit : Canali supportati
- 1 bit: Support PRF 64
- 3 bit: data rate supportati
- 8 bit: lunghezza di preambolo supportate
- 1 bit: supporto two way ranging
- 4 bit: padding

Lettura della configurazione

Anche in questo caso non si ha bisogno di parametri addizionali e il codice di funzione è 0x14.
Il tag risponderà a questa richiesta con un messaggio di 13 byte composto da un byte del valore fisso di 0x15 e 12 byte che rappresentano la configurazione del dispositivo.

I byte della configurazione sono strutturati in questo modo:

- 8 bit: Configuration ID (PHY) e lunghezza , fisso a 0x03
- 4 bit: Canale
- 3 bit: Lunghezza preambolo
- 1 bit: PRF, 0 equivale a 16 MHz e 1 a 64 MHz
- 5 bit: Codice di preambolo usato in trasmissione durante un two way ranging
- 3 bit: padding
- 5 bit: Codice di preambolo usato in ricezione durante un two way ranging
- 3 bit: padding
- 8 bit: Configuration ID (APP) e lunghezza, fisso a 0x27
- 16 bit: APP blink rate
- 16 bit: RX on time, tempo di attesa ad una risposta dopo un blink
- 16 bit: Response time, tempo che impiega il tag a generare una risposta
- 8 bit: Max poll retries, massimo numero di tentativi successivi alla mancata ricezione di una risposta da parte dell'infrastruttura

Scrittura della configurazione

Il comando di configurazione del tag è composto da un byte fisso del valore di 0x16 e altri 12 byte con la nuova configurazione.
Il tag può rispondere in due modi:

- in caso affermativo con un byte fisso di 0x17 e 12 byte con la nuova configurazione, i byte sono come quelli della lettura.
- in caso negativo con un byte fisso di 0x19 e 2 byte con un codice di errore che è dipendente dalla specifica implementazione.

API RTLS

L'API per i sistemi rtls come definita dallo standard ISO/IEC 24730-1:2014 è un'interfaccia utile alla condivisione con l'esterno dei dati ottenuti dall'infrastruttura. L'API deve poter essere supportata da un device con la minima funzionalità senza la necessità di uno storage permanente o di un database.

Lo standard definisce due modi di comunicare con altri software:

- Text over Socket
- Messaggi XML utilizzando HTTP

La sicurezza non è discussa in questo standard ma è lasciata allo sviluppatore che può utilizzare soluzione ben note dei protocolli specificati (ad esempio HTTPS).

Il protocollo viene chiamato SLMP cioè Simple Location Message Protocol e fa utilizzo di messaggi modellati da un formato denominato SLMF e cioè Simple Location Message Format.

I tipi di dato utilizzati nei messaggi sono:

- Datetime: Data nel formato YYYY-MM-DDThh:mm:ss-hh:mm, T sta ad indicare che seguirà un tempo, e la parte finale -hh:mm indica l'offset dalle coordinate universali
- Double: formato a virgola mobile che può essere visualizzato con o senza esponente e mantissa, il range di valore possibile varia tra 1.7E-308 e 1.7E+308, la lunghezza massima della stringa è di 256 caratteri
- HexBinary: dati che possono essere rappresentati in formato esadecimale dove ogni byte è un ottetto binario. Lunghezza massima di 256 bytes.
- Integer: rappresenta numeri che possono essere scritti senza la parte frazionaria o decimale. Il range di valore varia tra -2147483648 e 2147483647
- String: Set di caratteri ASCII. Lunghezza massima di 256 caratteri.

Text over socket

La comunicazione attraverso Text over socket viene inizializzata dall' RTLS attraverso un messaggio di intestazione (header message).
La sequenza dei campi è la seguente:
<Appliance_ID>, SLMF, <SLMF_version>, <SLMF_vendor_version>,<Greeting> <CR> <LF>

Appliance_ID : Stringa da 1 a 10 caratteri, rappresenta l'ID del sistema RTLS
SLMF_version : Stringa da 1 a 10 caratteri, rappresenta la versione del formato SLMF usato.
SLMF_vendor_version : Stringa da 1 a 10 caratteri, la versione del fornitore che implementa il formato SLMF.
Greeting : Stringa da 1 a 256 caratteri, messaggio specifico del provider del sistema RTLS che inizia la comunicazione
CR : Per CR si intende Carriage Return
LF : Per LF si intende Line Feed

Successivamente vengono inviate le definizioni dei campi dei messaggi di localizzazione nel seguente formato:
FieldDefinition, <Name>, <Type> <CR> <LF>

Name (obbligatorio) : Stringa da a 1 a 256 caratteri, è il nome associato a uno o più messaggi di localizzazione
Type (obbligatorio) : Stringa da 1 a 256 caratteri, Il tipo di dati associato al campo.

Poi vengono inviate le definizioni dei messaggi di localizzazione strutturate nel modo seguente:

LocateMessageDefinition, <Source>, <Format>, <Field1>, <Field2>,
<Field3>, ... <CR> <LF>

Source (obbligatorio) : Stringa da 1 a 64 caratteri, indica l'ubicazione o la famiglia di prodotti software.
Format (obbligatorio) : Stringa da 1 a 64 caratteri, se la locateMessageDefinition contiene campi (fields) opzionali allora l'unione tra source e format servirà a indicare come i messaggi dovranno essere processati, se contiene solo i campi obbligatori allora questo valore sarà impostato come "DFT".
Field Names (obbligatorio) : I nomi dei campi usati nel messaggio definiti precedentemente

Il sistema RTLS può anche includere un'altra struttura per la conferma di attività strutturata nel modo seguente:

KeepAlive, <Period> <CR> <LF>

Period (obbligatorio) : Stringa da a 1 a 3600 caratteri, intervallo di tempo che indica per quanto può rimanere silenziosa la connessione.

Infine vengono inviati i messaggi relativi alle localizzazioni , fino alla chiusura della connessione nel seguente formato:

<Source>, <Format>, <Tag_ID_Format>, <Tag_ID>, <X>, <Y>, <Z>,
<Battery>, <Timestamp>, <Ext1>, <Ext2>, <Ext3> ... <CR> <LF>

I campi obbligatori sono:

Source : Come definito precedentemente con i vari locateMessageDefinition
Format : Come definito precedentemente con i vari locateMessageDefinition
Tag_ID_Format : HexBinary di lunghezza 1 byte che può assumere tre valori:

- 0x01 : per il formato ISO / IEC 15963
- 0x02 : per il formato IEEE EUI-48
- 0x03 : per il formato IEEE EUI-64

Tag_ID : HexBinary di lunghezza variabile in base all'ID

X, Y, Z : Tutti e tre Double che indicano le coordinate della posizione

Battery : Integer che può assumere i seguenti valori:

- 0 : In buono stato
- 1 : Richiesta sostituzione
- 3 : Stato della batteria sconosciuto

Timestamp : DateTime indica il tempo della localizzazione

Gli opzionali sono i seguenti:

Classification : Stringa da 1 a 64 caratteri, rappresenta un gruppo di tag
Zone : Stringa da 1 a 256 caratteri, indica la zona della localizzazione
Exciter_ID : Stringa da 1 a 64 caratteri, Id del device che ha comandato al tag di effettuare blink
Antenna_ID : Integer da 0 a 255, ID del dispositivo a radiofrequenza che ha inviato la localizzazione

Data : HexBinary da 1 a 123 bytes, contiene dati che possono essere decodificati dall'API del client.
Algorithm : Stringa da 1 a 64 caratteri, l'identificativo della formula usata per trovare la posizione, è definita dal fornitore.
A, B , C : Double , A e C sono radianti da 0 a 2π, B invece da 0 a π

XML e HTTP

La comunicazione attraverso HTTP invece è pensata per interfacciarsi facilmente con applicazioni industriali e con il cloud.

In questo caso il sistema RTLS è un client, il Server è l'applicazione enterprise o cloud con cui si comunica. Il sistema RTLS invia una chiamata HTTP (nello specifico di tipo POST) contenente i messaggi accumulati in un certo periodo tempo, quest'ultimo configurato all'interno del sistema in base all'ambiente di esecuzione.

Questo connettore viene chiamato SLMF-HTTP connector e può essere interrotto o riavviato. Richiede i seguenti parametri di configurazione obbligatori:

- Url del servizio HTTP
- Porta del servizio HTTP

In aggiunta possono essere anche specificati i seguenti parametri:

- Numero massimo di messaggi
- Periodo di accumulazione
- Opzioni di recupero in caso di errore
    - il numero di tentativi dopo un errore
    - il numero di messaggi salvati prima della perdita di informazione

I messaggi sono scritti usando il formato XML con il seguente schema XML:

```
<? xml version = ”1.0” encoding = ”UTF-8”?>
<xs: schema xmlns: xs = ”http://www.w3.org/2001/XMLSchema”>
        <xs: element name = ”src”>
                <xs: simpleType>
                        <xs: restriction base = ”xs: string”>
                                <xs: minLength value = ”1” />
                                <xs: maxLength value = ”64” />
                        </ xs: restriction>
                </ xs: simpleType>
        </ xs: element>
        <xs: element name = ”fmt”>
                <xs: simpleType>
                        <xs: restriction base = ”xs: string”>
```

```
                    <xs: minLength value = ”1” />
                    <xs: maxLength value = ”64” />
                </ xs: restriction>
            </ xs: simpleType>
        </ xs: element>
        <xs: element name = ”idfmt”>
            <xs: simpleType>
                <xs: restriction base = ”xs: hexBinary”>
                    <xs: length value = ”1” />
                </ xs: restriction>
            </ xs: simpleType>
        </ xs: element>
        <xs: element name = ”tid”>
            <xs: simpleType>
                <xs: restriction base = ”xs: hexBinary”>
                    <xs: minLength value = ”6” />
                    <xs: maxLength value = ”8” />
                </ xs: restriction>
            </ xs: simpleType>
        </ xs: element>
        <xs: element name = ”x” type = ”xs: double” />
        <xs: element name = ”y” type = ”xs: double” />
        <xs: element name = ”z” type = ”xs: double” />
        <xs: element name = ”bat”>
            <xs: simpleType>
                <xs: restriction base = ”xs: int”>
                    <xs: enumeration value = ”0” />
                    <xs: enumeration value = ”1” />
                    <xs: enumeration value = ”3” />
                </ xs: restriction>
            </ xs: simpleType>
        </ xs: element>
        <xs: element name = ”t” type = ”xs: dateTime” />
        <xs: element name = ”cls”>
            <xs: simpleType>
                <xs: restriction base = ”xs: string”>
                    <xs: minLength value = ”1” />
                    <xs: maxLength value = ”64” />
                </ xs: restriction>
            </ xs: simpleType>
        </ xs: element>
        <xs: element name = ”zon”>
```

```
        <xs: simpleType>
                <xs: restriction base = ”xs: string”>
                        <xs: minLength value = ”1” />
                        <xs: maxLength value = ”256” />
                </ xs: restriction>
        </ xs: simpleType>
</ xs: element>
<xs: element name = ”exc”>
        <xs: simpleType>
                <xs: restriction base = ”xs: string”>
                        <xs: minLength value = ”1” />
                        <xs: maxLength value = ”64” />
                </ xs: restriction>
        </ xs: simpleType>
</ xs: element>
<xs: element name = ”ant”>
        <xs: simpleType>
                <xs: restriction base = ”xs: int”>
                        <xs: minInclusive value = ”0” />
                        <xs: maxInclusive value = ”255” />
                </ xs: restriction>
        </ xs: simpleType>
</ xs: element>
<xs: element name = ”dat”>
        <xs: simpleType>
                <xs: restriction base = ”xs: hexBinary”>
                        <xs: minLength value = ”1” />
                        <xs: maxLength value = ”123” />
                </ xs: restriction>
        </ xs: simpleType>
</ xs: element>
<xs: element name = ”alg”>
        <xs: simpleType>
                <xs: restriction base = ”xs: string”>
                        <xs: minLength value = ”1” />
                        <xs: maxLength value = ”64” />
                </ xs: restriction>
        </ xs: simpleType>
</ xs: element>
<xs: element name = ”SLMF”>
        <xs: complexType>
                <xs: sequence>
```

```
                    <xs: element ref = ”src” minOccurs = ”1” maxOccurs = ”1” />
                    <xs: element ref = ”fmt” minOccurs = ”1” maxOccurs = ”1” />
                    <xs: element ref = "idfmt" minOccurs = "1" maxOccurs = "1"
                    />
                    <xs: element ref = ”tid” minOccurs = ”1” maxOccurs = ”1” />
                    <xs: element ref = ”x” minOccurs = ”1” maxOccurs = ”1” />
                    <xs: element ref = ”y” minOccurs = ”1” maxOccurs = ”1” />
                    <xs: element ref = ”z” minOccurs = ”1” maxOccurs = ”1” />
                    <xs: element ref = ”bat” minOccurs = ”1” maxOccurs = ”1” />
                    <xs: element ref = ”t” minOccurs = ”1” maxOccurs = ”1” />
                    <xs: choice>
                        <xs: element ref = ”cls” minOccurs = ”0” maxOccurs
                        = ”1” />
                        <xs: element ref = ”zon” minOccurs = ”0”
                        maxOccurs = ”1” />
                        <xs: element ref = ”exc” minOccurs = ”0”
                        maxOccurs = ”1” />
                        xs: element ref = ”ant” minOccurs = ”0” maxOccurs
                        = ”1” />
                        <xs: element ref = ”dat” minOccurs = ”0”
                        maxOccurs = ”1” />
                        <xs: element ref = ”alg” minOccurs = ”0” maxOccurs
                        = ”1” />
                    </ xs: choice>
                </ xs: sequence>
            </ xs: complexType>
        </ xs: element>
        <xs: element name = ”Push_ Events”>
            <xs: complexType>
                <xs: sequence>
                    <xs: element ref = ”SLMF” minOccurs = ”1” maxOccurs =
                    ”1024” />
                    </ xs: sequence>
            </ xs: complexType>
        </ xs: element>
</ xs: schema>
```

Abbiamo quindi deciso di utilizzare la versione con HTTP perchè più facile da integrare in sistemi già esistenti.

Proof of concept

Andremo di seguito ad analizzare la soluzione sperimentale creata da noi.

Descrizione generale

Per l'infrastruttura abbiamo utilizzato un'architettura client/server tra le ancore e un software eseguito da sistema multipiattaforma.
La comunicazione tra essi avviene in rete locale tramite una connessione WiFi, dove i range collezionati dalle varie ancore vengono poi processati dal server che calcola le posizioni.
Il server poi invia le localizzazioni calcolate utilizzando un slmf http connector, scritto appositamente, ad un endpoint http configurato.

Ancore WiFi

Per i dispositivi che svolgono il ruolo di ancora, abbiamo deciso di utilizzare un microcontroller ESP8266, in particolare un modello denominato NodeMCU 1.0 poichè presenta già parte della circuiteria necessaria integrata e soprattutto connettività WiFi integrata e senza bisogno di circuiti esterni.
Un altro punto a favore che ha giocato nella scelta di questo dispositivo è la compatibilità con l'ecosistema Arduino, infatti l'API ufficiale per programmarlo implementata dal produttore è basata su quella ufficiale Arduino, denominata ESP8266 core for Arduino.
Oltre all' API standard Arduino il produttore ha implementato comode librerie per l'interfacciamento al modulo WiFi interno.
La libreria offre semplici funzioni e macro per la connettività, ad esempio per connettersi basta semplicemente eseguire una chiamata alla funzione

WiFi.begin("network-name", "pass-to-network");

e per verificare che si è connessi basta utilizzare la funzione *WiFi.status()* e controllare con delle macro il risultato ottenuto, ad esempio in questo caso:

WiFi.status() != WL_CONNECTED

La WiFi ci è utile per comunicare con un server che fa sempre parte dell'infrastruttura e che in seguito invierà le localizzazioni all'endpoint specificato. Ogni ancora invierà i range calcolati a questo server con una richiesta POST inviando il range calcolato, il proprio short address e quello del tag.

La libreria ufficiale permette di effettuare queste chiamate in vari modi, prima di tutti specificando manualmente le richieste effettuate in rete già inclusa nella libreria del WiFi o se non si ha bisogno di particolari impostazioni specifiche utilizzando delle astrazioni presenti nella libreria ESP8226HTTPClient.h .

Abbiamo deciso di utilizzare queste ultime poiché abbiamo utilizzato delle normali richieste HTTP.

Utilizzando queste librerie sono stati modificati gli sketch precedentemente discussi nella sezione riguardante il prototipo.

Il tag è rimasto pressoché uguale, i cambiamenti principali sono stati fatti al codice relativo alle ancore.
Nella fase di setup è stata aggiunta la logica relativa alla connessione Wifi, prerequisito necessario all'avviamento del dispositivo, il device prova a connettersi con le credenziali specificate e all'avvenuta connessione visualizza utilizzando la linea seriale l'indirizzo IP del dispositivo nella relativa rete.
Nel loop principale invece è rimasta invariata la componente di ranging ma è stata modificata la componente relativa alla notifica del range.
Le ancore non informano più una ancora 'Main' ma inviano il risultato di range ottenuto direttamente al server che verrà utilizzato per la gestione dei dati ricevuti e il calcolo della posizione dei tag. L'ancora denominata Main rimane però il principale intermediario tra il tag e l'infrastruttura poichè è quella addetta alla ricezione dei blink, assegnazione dello short address al tag e l'iniziazione dei ranging necessari alla localizzazione con le ancore successive.
Ovviamente è stato rimosso l'algoritmo di localizzazione dai device nell'ottica, appunto, di spostare il calcolo sul server per una più facile gestione dei processi e velocità computazionale.
Questo cambiamento ha migliorato il flusso di scambio tra il tag e le varie ancore riducendo gli errori, poiché sono stati eliminati i report tra le varie ancore e quindi si è ridotta la possibilità di errori di sovrapposizione di messaggi in aria.

Server di localizzazione

Il server[9] adibito alla ricezione dei range ed alla notifica dell'endpoint come specificato dallo standard ISO/IEC 24730-1:2014 è stato sviluppato utilizzando il linguaggio di programmazione Javascript per via della necessità di avere uno sviluppo veloce e con l'ausilio di un ecosistema maturo e moderno come quello creato dal runtime Node.js e del suo package manager NPM corredato di moltissime librerie open source.

[9] "GitHub - F-Army/RTLS-infrastructure-PoC: Proof of concept RTLS ...." 1 Dec. 2018, https://github.com/F-Army/RTLS-infrastructure-PoC. Accessed 15 Jan. 2019.

Per la creazione dell' API è stato utilizzato il framework Express.js che permette una veloce e modulare implementazione delle routes, con la possibilità di creazione di vari middleware per l'analisi e la processazione delle richieste, così da rendere l'API estendibile facilmente in futuro o per particolari esigenze dovute a specifiche.
L'API è suddivisibile in due sotto categorie:

- Interna all'infrastruttura
- Per la configurazione esterna di parametri

API Interna all'infrastruttura : Questa parte dell'API è utilizzata per il report delle ancore dei vari messaggi e nello specifico è la parte designata alla ricezione dei range e della loro processazione.

Configurazione esterna : Questa componente è utilizzabile per inserire elementi all'interno dell'infrastruttura ed identificarli, nell'implementazione specifica viene permessa l'aggiunta di nuovi device con la specifica del loro short address e per le ancore anche della loro posizione nella rappresentazione in coordinate cartesiane del luogo fisico.

Per quanto riguarda la componente di invio dei messaggi di localizzazione è stata utilizzata una libreria sviluppata da noi chiamata SLMF-HTTP-Connector che andremo ad analizzare di seguito.

SLMF-HTTP-Connector

L'implementazione dell' SLMF-HTTP-Connector[10] di cui si è discusso nella descrizione generale è stata valutata ed infine si è optato per la creazione di una libreria da poter così integrare in altri applicativi o da poter rendere standalone con poche righe di codice, incentivando così uno sviluppo modulare.
Nella prima fase di sviluppo si è utilizzato javascript per via delle innumerevoli librerie open source presenti nell'ecosistema di packaging NPM del runtime javascript Node.js e la facile integrazione di esse.
Durante lo sviluppo ci siamo però resi conto che javascript al crescere della codebase tende a crescere nella complessità di mantenimento, abbiamo così valutato come poter ovviare a questo problema.
Dopo un'accurata analisi si è optato per la conversione del codice della libreria utilizzando il linguaggio Typescript, un superset di Javascript sviluppato da Microsoft che aggiunge al linguaggio feature molto simili a linguaggi più *enterprise like* come C# o Java e che compila il suo codice sorgente in Javascript mantenendo così una piena compatibilità al 100%.

[10] "GitHub - F-Army/SLMF-HTTP-connector: Typescript implementation of …." https://github.com/F-Army/SLMF-HTTP-connector. Accessed 15 Jan. 2019.

L’architettura del connector è stata pensata e sviluppata sin dalle prime fasi con un’ottica Object Oriented così da essere modulare e facilmente testabile.
Partendo dalle specifiche dello standard ISO/IEC 24730-1:2014 la libreria è stata divisa nei seguenti moduli:

- Accumulator
- ConnectorLoop
- ConnectorSettings
- LocationMessage
- SlmfHttpConnector

Accumulator : E’ il componente di tipo generico (e quindi utilizzabile con vari tipi di strutture dato) che permette l’accumulazione di dati fino ad una determinata quantità specificata durante la creazione. Nella visione complessiva della libreria è quel componente che accumula i messaggi di localizzazione da inviare successivamente all’endpoint richiesto.

ConnectorLoop : Il ConnectorLoop è un modulo che permette di creare una routine generica e di eseguirla ciclicamente in un determinato intervallo di tempo, oltre a questo può essere avviato e fermato in qualsiasi momento. Nell’ottica complessiva della libreria è quel componente che permette di inviare ciclicamente messaggi di localizzazione accumulati all’endpoint richiesto.

ConnectorSettings : Questo componente implementa i parametri di configurazione richiesti dallo standard che sono:

- Periodo di accumulazione
- Massimo numero di messaggi accumulati
- Numero di tentativi massimi
- Massimo numero di messaggi da inviare in massa
- Url dell’endpoint a cui eseguire la richiesta
- Porta dell’endpoint a cui eseguire la richiesta

Questi parametri vanno specificati alla creazione dell’oggetto connectorSettings e per la loro validazione è stata utilizzata la libreria open source Joi (sviluppata internamente dal team tecnico di Walmart e resa open source), che semplifica la validazione dei campi senza intaccare la leggibilità del codice.

LocationMessage : E’ la componente che raccoglie i parametri delle varie localizzazioni discusse nello standard e mette a disposizione delle funzioni per la loro processazione e manipolazione. Per la validazione anche in questo è stata utilizzata la libreria Joi.
Questo componente permette la conversione e la restituzione di una stringa in formato XML dei parametri contenuti. Per la conversione è stata utilizzata la libreria xml2js.

SlmfHttpConnector : Collante principale della libreria che utilizza gli altri componenti discussi precedentemente per creare un connector http che implementa quello teorico dello standard.

Oltre agli altri componenti sono state utilizzate due librerie:

- Axios: per effettuare le richieste HTTP all'endpoint esterno a cui notificare i messaggi di localizzazione in XML
- Axios-retry: per effettuare i vari tentativi massimi impostati dallo sviluppatore

Il codice è stato sviluppato utilizzando la metodologia Test-Driven-Development per ottenere un pacchetto più stabile e affidabile. Per eseguire i test è stato utilizzato il framework di testing Jest (sviluppato inizialmente dal team di Facebook).

Il codice sorgente della libreria è disponibile al seguente link:
https://github.com/F-Army/SLMF-HTTP-connector/

Per quanto riguarda le release da utilizzare in un progetto, esse possono essere scaricate dal repository NPM[11].

[11] "slmf-http-connector - npm." 15 Nov. 2018, https://www.npmjs.com/package/slmf-http-connector. Accessed 15 Jan. 2019.

Panoramica del mercato RTLS

Durante lo sviluppo del progetto di questa tesi il mercato relativo al campo dei sistemi RTLS ha visto l'avvio e la crescita di alcune aziende che producono o utilizzano questo tipo di tecnologia.
Il caso più rilevante rispetto alle nostre ricerche è Sewio[12] che durante questo periodo ha rilasciato sul mercato prodotti di alta qualità ed ha installato i suoi sistemi andando a soddisfare gli use case di grandi brand e aziende operanti nel settore dei servizi.
Un caso tutto nostrano è quello di Enel[13] che ha utilizzato questa tecnologia per il controllo delle presenze e accessi in una loro infrastruttura, così da aumentare la sicurezza per i lavoratori e nei casi di emergenza. Altri clienti famosi sono Skoda, Volkswagen, Bosch e IBM.
Un altro fattore chiave di questo mercato è la nascita di alternative ai chip creati dalla Decawave. Degno di nota è un chip che può essere utilizzato sia in maniera complementare sia sostitutiva a seconda del business case, creato dalla Texas Instruments chiamato mmWave Radar Sensor[14] che permette di localizzare e contare le persone in una stanza senza l'utilizzo di tag.
Un altro indizio della richiesta di questa tecnologia e l'utilizzo aumentato nel tempo è anche dovuto alla creazione di terze parti di nuovi moduli e kit di sviluppo utilizzanti il chip DW1000 discusso in questa tesi.
Questi sono chiari segni di un nuovo settore nascente e interessante da studiare e valutare nei prossimi anni.

[12] "Sewio." 8 Jan. 2019, https://www.sewio.net/. Accessed 15 Jan. 2019.
[13] "Employee Tracking to Improve Safety using RTLS in ENEL | Sewio RTLS." https://www.sewio.net/customer-projects/enel/. Accessed 15 Jan. 2019.
[14] "Millimeter Wave (mmWave) Sensors | Overview | TI.com." http://www.ti.com/sensors/mmwave/overview.html. Accessed 15 Jan. 2019.

Conclusioni e possibili sviluppi futuri

Dalle ricerche e sperimentazioni effettuate è stato constatato che la tecnologia UWB è un ottimo candidato per la creazione di sistemi di localizzazione indoor con una ottima precisione (nello specifico anche 5-10 cm di errore massimo) e l'attuale panoramica hardware potrebbe permettere uno sviluppo ad un prezzo competitivo e affine a tecnologie simili, soprattutto in vista della precisione dei chip esistenti.
Anche lo sviluppo di standard gioca a favore di questa tecnologia per via delle possibili interazioni tra sistemi creati con l'ottica di essere compatibili.
Le maggiori difficoltà riscontrate durante lo sviluppo e la prototipazione sono dovute all'algoritmo da noi utilizzato, il Two Way Ranging.
La scelta di tale algoritmo è stata dettata dall'esigenza di avere un buon compromesso tra la difficoltà dello sviluppo hardware (grazie anche a dei moduli hardware di facile interfacciamento già esistenti) e lo sviluppo software di algoritmi già affrontati in maniera teorica da altre fonti di ricerca.
Seppur questo algoritmo non necessita di particolari attenzioni durante lo sviluppo dell'hardware o di particolari calcoli matematici complessi, tuttavia presenta gravi limitazioni per uno sviluppo di sistemi scalabili sia nella funzionalità che nella manutenzione. Il massimo numero di tag localizzabili in un sistema che fa completamente uso del suddetto algoritmo si attesta a poche decine e la complessità cresce considerevolmente poiché per ottenere le distanze necessarie alla localizzazione c'è bisogno di un numero elevato di scambio di dati tra i dispositivi usati come tag e le ancore fisse usate per la localizzazione. Oltre al numero limitato di device utilizzabili e la loro difficoltà di coordinazione, un'altra problematica risiede nell' elevato consumo energetico dei tag stessi dovuto alla necessità di ricevere messaggi, il che comporta una durata della batteria che varia da qualche decina di ore a qualche settimana in casi di elevata ottimizzazione dei consumi attraverso un'intelligente design dell'hardware e del software che lo gestisce eliminando le trasmissioni e ricezioni non necessarie (ad esempio quando il dispositivo non è in movimento).
Tuttavia per quanto riguarda applicazioni che necessitano solo di ranging e non di localizzazione è un ottimo candidato per sistemi che hanno bisogno di un'elevata precisione.

Per quanto riguarda possibili sviluppi futuri di ricerca, vale la pena analizzare gli altri algoritmi utilizzabili per l'indoor positioning, soprattutto da ricerche effettuate in particolare gli algoritmi PdoA e TdoA.
Entrambe le soluzioni dovrebbero riportare vantaggi e svantaggi simili, i vantaggi riguardano soprattutto la non necessità dei dispositivi di tag di mettersi in ricezione, il che permette di passare da un' autonomia di al massimo poche settimane ad una di

qualche anno in base alle necessità di localizzazione. Entrambe riducono drasticamente la complessità software per quanto riguarda la coordinazione delle trasmissioni. Gli svantaggi, invece, riguardano la maggiore complessità e costo di sviluppo dell'hardware dovuto alla necessità di creare circuiti specifici per entrambi gli algoritmi. Nel caso del PdoA si ha bisogno di un array di antenne posizionate in modo specifico con lo scopo di calcolare l'angolo di arrivo, invece, nel caso del TdoA si ha bisogno di una sincronizzazione esterna molto precisa del clock dei moduli UWB e quindi di una infrastruttura apposita che generi centralmente il clock e lo distribuisca ai vari dispositivi con tutte le problematiche hardware che ne derivano.

Infine un'altra possibilità è quella di ricercare una nuova tipologia di algoritmi che non presentino queste problematiche e allo stesso tempo permettano un risparmio energetico di simile entità[15] [16].

[15] "High-accuracy TDOA-based Localization without Time Synchronization." https://ieeexplore.ieee.org/iel5/71/4359390/06287503.pdf. Accessed 15 Jan. 2019.
[16] "Synchronization-Free and Low Power TDOA for Radio Based Indoor ...." 16 Nov. 2018, https://www.researchgate.net/publication/327890864_Synchronization-Free_and_Low_Power_TDOA_for_Radio_Based_Indoor_Positioning. Accessed 15 Jan. 2019.

Bibliografia

[1] "APS003 DW1000 RTLS Introduction - Decawave." https://www.decawave.com/wp-content/uploads/2018/10/APS003_DW1000-RTLS-Introduction_v1.1.pdf. Accessed 15 Jan. 2019.

[2] "ISO/IEC 24730-62:2013 - Information technology -- Real time locating ...." https://www.iso.org/standard/60379.html. Accessed 15 Jan. 2019.

[3] "dw1000 user manual - Decawave." https://www.decawave.com/wp-content/uploads/2018/09/dw100020user20manual_0.pdf. Accessed 15 Jan. 2019.

[4] "DW1000 Datasheet - Decawave." https://www.decawave.com/sites/default/files/resources/dw1000-datasheet-v2.09.pdf. Accessed 15 Jan. 2019.

[5] "DWM1000 Datasheet - Decawave." https://www.decawave.com/sites/default/files/dwm1000-datasheet.pdf. Accessed 15 Jan. 2019.

[6] "GitHub - thotro/arduino-dw1000: A library that offers functionality to ...." https://github.com/thotro/arduino-dw1000. Accessed 15 Jan. 2019.

[7] "GitHub - F-Army/arduino-dw1000-ng: Arduino driver and library to use ...." https://github.com/F-Army/arduino-dw1000-ng. Accessed 15 Jan. 2019.

[8] "ISO/IEC 24730-1:2014 - Information technology -- Real-time locating ...." https://www.iso.org/standard/59801.html. Accessed 15 Jan. 2019.

[9] "GitHub - F-Army/RTLS-infrastructure-PoC: Proof of concept RTLS ...." 1 Dec. 2018, https://github.com/F-Army/RTLS-infrastructure-PoC. Accessed 15 Jan. 2019.

[10] "GitHub - F-Army/SLMF-HTTP-connector: Typescript implementation of ...." https://github.com/F-Army/SLMF-HTTP-connector. Accessed 15 Jan. 2019.

[11] "slmf-http-connector - npm." 15 Nov. 2018, https://www.npmjs.com/package/slmf-http-connector. Accessed 15 Jan. 2019.

[12] "Sewio." 8 Jan. 2019, https://www.sewio.net/. Accessed 15 Jan. 2019.

[13] "Employee Tracking to Improve Safety using RTLS in ENEL | Sewio RTLS." https://www.sewio.net/customer-projects/enel/. Accessed 15 Jan. 2019.

[14] "Millimeter Wave (mmWave) Sensors | Overview | TI.com." http://www.ti.com/sensors/mmwave/overview.html. Accessed 15 Jan. 2019.

[15] "High-accuracy TDOA-based Localization without Time Synchronization." https://ieeexplore.ieee.org/iel5/71/4359390/06287503.pdf. Accessed 15 Jan. 2019.

[16] "Synchronization-Free and Low Power TDOA for Radio Based Indoor ...." 16 Nov. 2018, https://www.researchgate.net/publication/327890864_Synchronization-Free_and_Low_Power_TDOA_for_Radio_Based_Indoor_Positioning. Accessed 15 Jan. 2019.