UAVCAN-enabled indoor postioning solution for unmanned vehicles

Riccardo Miccini
Technical University of Denmark
Centre for Bachelor of Engineering Studies

June 21, 2016

Abstract

Usual abstract stuff.

The opinions expressed in this document are the author's own and do not reflect the view of UAVComponents and its personnel.

All brands, product names, logos, or other trademarks featured or referred to in this document are the property of their respective holders, and their use does not imply endorsement.

Contents

Chapter 1

Introduction

This introductory chapter will provide background information about Real-time Locating Systems (in short — and throughout the rest of this document — *RTLS*) and the current state of the art, along with a brief description of the partner company, the project description under the form of assignment formulation, and a brief overview of the implementation. The latter will be elaborated further in the following chapters.

1.1 Rationale

The importance of precise location

1.1.1 RTLS

From map-making to post tracking, localization has been a fundamental human endeavor which encompassed several ages and civilizations. Although GPS has proven to be a major leap is this regard, its current limitations causes it to be unsuitable for some purposes. First of all, the accuracy lays in the order of a few meters at best, mainly due to the relativistic implication of orbiting satellites [1] (that is without considering more advanced techniques such as *RTK*). Moreover, signal shadowing and multi-path reflections renders it particularly unreliable indoor or in dense urban areas. In order to fulfill the ever-growing demand for automation, more precise location systems have to be employed.

The term RTLS emerged from such needs, and characterizes systems capable of reckoning, tracking, and showing the position of moving objects and persons within a relative frame of reference, mainly a building or enclosed area. The applications benefiting from those systems mainly lay within the logistical/operational areas, and the interested parties range from manufacturing industries to health care operators (management of flows of goods in a warehouse or production plant; assets retrieval e.g. tools, medical equipment, cattle; surveillance and monitoring; personnel deployment and supervision). Consumer-related applications are also starting to arise in the fields of domotics and electronic devices.

The technologies driving these systems include ultrasounds, infrared radiation, or RSSI (Received signal strength indication) from various sources such as radio frequency modules, WiFi/WLAN, or even Bluetooth; however, each of them presents its own limitations which prevented them from being used in large scale. [2]

The more recent implementation of previously known technologies into positioning systems spawned a vast array of previously unforeseen applications, such as their use along with drones and other unmanned vehicles for maintenance and inspection of facilities, emergency situation management, and whenever human intervention may pose a safety threat.

1.1.2 UAVCAN

Although drones, copters, rovers, and the like are are experiencing a huge rise in popularity in both commercial and hobby use, integrators and final users often have to cope with a vast amount of different protocols and interfaces, some of which directly inherited from the RC modeling world, thus lacking fault tolerance and scalability. For years, the automotive industry has relied on CAN Bus as a means of communication between different microcontrollers and devices, due to its low cost and resilience against electro-magnetic resilience.

UAVCAN is a lightweight open source protocol aimed to provide fast and reliable communication within drone peripherals. [3] A UAVCAN network is composed of a series of nodes which are independent from the others and require no master/controller or host computer. The protocol sits on top of the transport and physical layers of CAN and relies on a library for easy integration with the user application code. Common standard high-level functions like node health monitoring, network discovery, time synchronization, and firmware update are predefined and taken care of by the library, while message definitions for most drone equipment (speed controllers, navigation data, gimbals, actuators) are standardized and publicly available, making it easy for manufacturers to engineer compliant devices and for integrators to interface with them.

1.2 Company description

UAVComponents ApS is a danish company which develops and supplies components, equipment, and tools for drones and UAVs integrators and hobbyists. Its main products are the *TSLRS* line of transceivers and the *Aeronav One* ground control station. Along with off-the-shelf hardware solutions, the company offers its expertise to clients seeking consultancy or ad-hoc projects. UAVComponents shares its offices and personnel with *Danish Aviation Systems*, a sister company also operating in the drones industry.

The team is composed of roughly 8 people, almost all of whom are involved in Research&Development. Besides a solid know-how of piloted and autonomous aerial vehicles, the core competencies include the electronic design of analogic and MCU-based solutions, and software development — both for embedded and computer systems. The company relies on external support for more skill-intensive tasks such as mechanical and RF design. Manufacture is also mostly outsourced, especially for high volume yields.

As the vast majority of the employees is quite young, the company can boast a very dynamic and informal atmosphere, which in turn resulted in flexible working hours and a high degree of collaboration among all the colleagues.

1.3 Problem formulation

The aim of the project is to design and develop a UAVCAN-compatible indoor positioning system for aerial unmanned veicles (UAVs) based on the *DecaWave DWM1000* devices. Using these radio modules, an algorithm shall be devised and implemented in order to estimate the UAV position in a 2D coordinate system. The resulting information is to be broadcasted through the UAVCAN bus — a message definition shall be chosen amongst the pool of standard definitions, or designed from scratch.

The performance of the system ought to be in line with DecaWave's claims in term of maximum accuracy and range, and the device has to be suitable for battery operation. Moreover, the system shall comply with the UAVCAN specifications and support its major standard functionalities.

The project is carried out in collaboration with *UAVComponents ApS*, which will lend its expertise in hardware design and provide the necessary development tools.

1.4 System description

The system is composed of one (or more) *tags* and up to four *anchors*.

The anchors are to be spread across the area of operation. Their coordinates within the frame of reference shall be know beforehand. The tag device is located inside the drone/vehicle and is connected through the UAVCAN bus.

Both type of devices share the same hardware design (mainly for development sake), but are flashed with different firmwares. It comprises:

- ARM Cortex-M4 microcontroller
- DecaWave DWM1000 module
- Inertial measurement unit
- Barometric sensor
- Power supply unit
- CAN transceiver and connectors
- USB interface
- SWD connector for debugging and flashing

The boards can be powered through either the USB or the UAVCAN connectors.

The software infrastructure governing both firmwares is built upon *ChibiOS*, a real-time operating system for embedded devices. Each device — regardless of its role — can be configured through a command line interface accessible through the USB.

Each anchor runs the same software routine, but with different parameters (coordinates and transmission delay). Their role is to respond to the polling message sent by the tags.

The tag firmware is responsible for initiating new communications and processing it (in a way that is thoroughly described in the Theoretical principles).

Whenever a new response message is parsed and the distance between the tag and that anchor calculated, the position estimation is updated. Lastly, the location broadcaster transmit the current estimation to the UAVCAN bus, in an asynchronous fashion.

Chapter 2

Problem analysis

The following sections aim to elaborate on the Problem formulation and create a more solid base to use as a guideline and template during the actual development. The adopted framework is based upon the one presented during the course of the studies [4] and bear similarities with the *Unified Software Development Process.*

This analysis does not aspire to be exhaustive, but rather satisfactory in the level of detail required to establish the project.

2.1 Requirements specification

2.1.1 Functional requirements

As per definition, they “define specific behavior or functions”, and are the basis for the tests.

2.1.2 Non functional requirements

Other charateristics and qualities of the system.

2.1.3 Use cases

This section describes how the actors can interact with the system. For each there should be: brief description, actors, preconditions, main flow, post conditions, remarks / alternative flows.

2.2 Theoretical principles

This section will present the theoretical aspects of the technologies employed in the project, and provides a knowledge base for better understanding the problem and the employed solution.

2.2.1 Ultra-wideband

Ultra-wideband (UWB in short) is a radio technology that use a very low energy level for short-range, high-bandwidth communications over a large portion of the radio spectrum.

Although this technique has been know for for decades, is a relatively new entry in the RTLS field [5], with the first affordable development kits having entered the market in the last few years. It gained immediate popularity due to its advantages: blablabla

2.2.2 Ranging algorithm

There can be several possible two-way ranging schemes, blabla

Their role is to respond to the polling message sent by the tags after predetermined delay. This response message constains the polling message reception timestamp and its own transmission timestamp.

2.2.3 Position estimation

Explanation of the triangulation algorithms.

2.2.4 CAN Bus

Brief explanation of CAN protocol, its history, and principles of operation. May also be opted out and substituted with notions of ARM Cortex-M architecture.

2.3 Risk management

Risk assessment and management are well-established disciplines amongst many fields of engineering and other areas of human endeavor, and represent a way of minimizing the probability and impact of adverse conditions in a given context.

In the Overview subsection, possible sources of hindrance are categorized and listed. Preventive measures are then suggested in the Prevention and mitigation subsection.

2.3.1 Description

This section acts as a legend, providing the necessary information for interpreting the subsequent risk assessment. Risks are evaluated according to their composite risk index (*CRI*):

$$\text{CRI} = \text{Severity} \times \text{Probability of occurrence} \tag{2.1}$$

The CRI represents the degree of importance of each entry. Entries with a high (ie. red-leaning) CRI shall have maximum priority.

Traditionally, this kind of analysis involves assigning a numeric value from an arbitrarily chosen range to each variable — e.g. 1–5 for the severity and 0–100% for the probability. However, educated estimations require solid data and a tracked history of previous occurrences, all of which were either not available or difficult to obtain with the necessary degree of confidence. [6] Therefore, the final value of the CRI has been neglected in favor of a color-mapped matrix composed of four sub-ranges. Whilst selecting these sub-ranges, higher “granularity” has been given to the more adverse scenarios.

		Probability of occurrence			
		Negligible (1)	Low (2)	Medium (3)	High (4)
Severity	Small (1)	🟩	🟩	🟨	🟨
	Moderate (2)	🟩	🟨	🟨	🟧
	Large (3)	🟨	🟨	🟧	🟥
	Critical (4)	🟨	🟧	🟥	🟥

Table 2.1: Risk impact matrix

2.3.2 Overview

Follows the lists of threats (divided by category) with associated likelihood, impact, and overall CRI. The risk index is accompanied with the affected resource(s), chosen amongst:

- Product **F**unctionality
- Project **C**ost
- Development **T**ime

Whenever possible or necessary, a brief explanation of the predicted consequences is given.

Hardware

The estimations are based on company personnel’s knowledge and previous experience with the given components and communication protocols governing them. For the purpose in this analysis the board components have been divided in primary (microcontroller, DWM1000 module, CAN transciever) and secondary (power supply unit, other sensors, USB interface), based on their importance for the fulfillment of the project.

Software

The estimations are based on personal previous development experience, online research, and relevant documentation. The table sub-categories loosely reflect the different software layers described in the Software section.

2.3.3 Prevention and mitigation

After having discerned and evaluated as many threats as possible, a number of preemptive measures will be suggested. Four approaces blablabla

Due to the similarities amongst related entries, the risk prevention methods will be discussed on a sub-category basis.

ID	Risk	Prob.	Sev.	CRI	Res.	Contingencies/Notes
SD	*Schematics design issues (wrong/missing parts, connections...)*					
SD.1	Primary components	3	4	🔴	TC	New PCB manufacturing
SD.2	Secondary components	2	3	🟡	TF	
PL	*PCB layout issues (wrong footprints, overlapping traces...)*					
PL.1	Primary components	3	3	🟠	TC	New PCB manufacturing
PL.2	Secondary components	1	2	🟢	TF	
SC	*Supply chain issues*					
SC.1	Components lead time	1	3	🟡	T	Wait until restocked
SC.2	PCB lead time	3	3	🟠	T	Negotiation/bargaining
SC.3	Defective PCB	1	4	🟡	T	New PCB delivery
PU	*Performance/usability issues*					
PU.1	MCU doesn't boot up	1	4	🟡	T	
PU.2	MCU not detected by SWD	2	4	🟠	T	Tedious troubleshooting
PU.3	High packet loss (USB, CAN)	1	2	🟢	F	
PU.4	DWM1000 calibration off	2	2	🟡	F	

Table 2.2: Risk assessment: Hardware-related threats

ID	Risk	Prob.	Sev.	CRI	Res.	Contingencies/Notes
DE	*Development environment issues*					
DE.1	No compiler available	1	4	🟡	C	Switch MCU vendor/arch. or host systen
DE.2	No flashing tools available	2	4	🟠	C	Switch MCU vendor/arch. or host systen
DE.3	Loss of data	2	3	🟡	T	
LL	*Low-level support issues (peripherals, data busses...)*					
LL.1	No MCU peripherals drivers	2	3	🟡	TF	Implement from scratch
LL.2	No MCU peripherals docs.	1	4	🟡	TC	Switch MCU vendor/arch.
LL.2	No comm. with ext. devices	2	4	🟠	TF	
LL.4	RTOS not supported	2	1	🟢	TF	Implement port; single thread paradigm
MW	*Middleware issues (libraries, device APIs...)*					
MW.1	Restrictive/costly license	1	3	🟡	CF	Purchase; Implement from scratch
MW.2	Lack of platform support	3	2	🟡	TF	Implement port
MW.3	Poor documentation	2	3	🟡	TF	
MW.4	Unstable/buggy code	1	2	🟢	F	
UA	*User application-level issues*					
UA.1	Reached computational limit	1	3	🟡	F	
UA.2	Middlewares conflict	2	3	🟡	TF	
UA.3	Poor ranging accuracy	2	3	🟡	F	
UA.5	Unimplemented use cases	2	4	🟠	T	Increased time-to-market
UA.6	Unmet requirements	2	4	🟠	T	Increased time-to-market

Table 2.3: Risk assessment: Software-related threats

ID	Description	Approach
DE	*Development environment issues*	
LL	*Low-level support issues (peripherals, data busses. . .)*	
MW	*Middleware issues (libraries, device APIs. . .)*	
UA	*User application-level issues*	

Table 2.4: Risk management: threats preventions and mitigation proposal

2.4 Milestone plan

Here is the project milestone plan as formulated during the *Inception* and *Elaboration* phases. This version of the plan lists a series of achievements deemed necessary to accomplish the major functionalities. It is arranged in a progressive way, with a time axis roughly flowing vertically. Each task depends from its sub-tasks and — more indirectly — on the previous ones sharing its hierarchy level.

Although it has been used — together with the previously discussed Use cases — as blabla, casini vari

Please refer to the Possible improvements for the final version of the milestone plan based on the actual project development.

This section should be based on the following (sorted out) milestone plan. It should also refer to the conclusions and appendices for remarks and detailed logs.

1. get the modules to work (Arduino, mbed)
    (a) simple tx/rx
    (b) ranging
    (c) localization
2. schematics design
3. software development
    (a) implement basic peripherals
    (b) port working localization code to ARM
    (c) integrate libUAVCAN
4. integration and validation tests
5. documentation
    (a) final report
    (b) product datasheet
    (c) production files

Chapter 3

Design and implementation

3.1 Hardware

3.1.1 Setup

Information about the environment: KiCad, libraries, supply voltages, connectivity, and tags / beacons layout. Information about the physical tools: dev board, debugger probe, scopes etc ...

3.1.2 decaWave module

Information about the decaWave IC and module, specs, modes of usage.

These UWB radio modules are designed for easy integration of two-way ranging schemes.

3.1.3 Microcontroller

Information about the MCU used, its charateristics, and the reasons behind its adoption.

3.1.4 Sensors

Other relevant sensors mounted on the board. The board may feature a barometric sensor (for altitude) and an IMU.

3.2 Software

3.2.1 Setup

Development environment

Description of the IDE and compiler.

Project structure

Explain code subdivision (HAL, modules, processes, other user app code, and libraries).

Other tools

Sublime Text, git, GitHub, debug server, UAVCAN tools.

3.2.2 Hardware abstraction layer

SPI

Used by the decaWave module.

EEPROM

Settings storage.

UART

Communication with PC, exclusively during development.

Timers (SCT, MRT, RIT, SysTick)

PWM signals, time tracking.

Real-time Clock

Time tracking.

Watchdog

Necessary safety feature.

Internal temperature sensor

Safety and system self-test.

CRC and Flash signature engines

Savety and system self-test.

3.2.3 Software modules

decaWave module

Abstraction layer to its registers and functionalities. May consist in the APIs provided by the IC supplier.

Processes manager

Executes different tasks at required time intervals, monitors their return values, collects statistics. Explain differences between normal tasks and idle task.

Configuration manager

Stores and retrieves settings of arbitrary type. Needed by pretty much anything. Will be implemented on top of EEPROM since it's the only available permanent, user-writtable memory (Flash is too, but on a page-basis).

Command line interface

Used for providing runtime infos and changing settings without the need of a debugger probe. Will read from UART.

Time tracking

Provides time tracking and management capabilities (virtual timeouts and stop-watches, delays). Needed by UAVCAN and processes manager. May be POSIX-compliant, if strictly necessary. May be implemented through the RTC and timers peripherals or decaWave timestamps system.

System self-test

Verifies correct state of operation and data integrity. Needed by UAVCAN (every node shall broadcast its state).

3.2.4 UAVCAN

Protocol

Description and explanation of the protocol (only uppermost layers).

Library

Explanation of the different implementations and functionalities of the reference one.

Platform driver Description of driver interface, microcontroller CAN interface, and necessary adaptation.

3.2.5 Processes

(There may be more processes).

UAVCAN Node

Broadcasting of node information, server for handling configuration changes, firmware updates, and so on.

Sensor reading

As per title.

Position estimator and publisher

As per title.

Idle task

Updates watchdog, blinks LEDs.

Chapter 4

Testing

4.1 Acceptance tracing

This section should be an overview of the tests conducted. A table shall correlate the requirements with the tests that prove their fulfillment, as well as the outcome (passed, not passed, partially passed, feature not implemented).

4.2 Acceptance tests

Detailed view of the hardware-related tests. Should show the following: purpose (which requirements it tests), preconditions, test actions wth relative expected results.

4.3 Unit tests

This shall contain a similar set of information, based on the software unit tests. May not be necessary.

Chapter 5

Conclusions

5.1 Product assessment

Conclusions based on the product, e.g. if it fulfilled all the requirements, if it yielded the desired performances etc ...

5.2 Process assessment

Conclusions based on the development, e.g. how lean it has been, any useful tool discovered, any cumbersome aspect of the workflow that coud have been improved ...

5.3 Possible improvements

Suggestions on improvements based on missing features, company and supervisors feedbacks, etc ...

Bibliography

[1] 2016 — Richard W. Pogge — Real-World Relativity: The GPS Navigation System `http://www.astronomy.ohio-state.edu/~pogge/Ast162/Unit5/gps.html`

[2] 2009 — Joost Koppers — Location Based Services: A general model for the choice between positioning techniques (Dutch) Radboud University Nijmegen

[3] 2014 — Pavel Kirienko — UAVCAN Specification `http://uavcan.org/Specification/1._Introduction/`

[4] 2013 — DTU Course Base — Object oriented software engineering compendium `http://www.kurser.dtu.dk/2013-2014/62432.aspx`

[5] 2005 — Faranak Nekoogar — Ultra-Wideband Communications: Fundamentals and Applications ISBN-10: 0-13-146326-8

[6] 2004 — David Hillson & David Hulett — Assessing Risk Probability: Alternative Approaches PMI Global Congress Proceedings

Appendix A

Glossary

MCU Microcontroller Unit

NXP A semiconductor designer and manufacturer company

RTLS Real-time locating system

UAV Unmanned aerial vehicle, commonly known as drone

term definition

another term another definition

Appendix B

User manual

This section should explain how to set up and use the device, the way a user manual normally would.

Appendix C

Further material

Further content to be found in the hand-in support, with explanation of its folder structure. It may include:

- Source code
- Repository commit logs
- Matlab scripts
- Data and pictures from tests
- Production files
- Bill of materials