Cécile Lauriac, Ievgenii Meshcheriakov, RJJ Foster
nanotron Technologies © 2013

Collision Avoidance Systems (CAS) using nanotron's *swarm* radios

In open pit and underground mines, building sites and train-yards, there is a permanent risk of serious accidents due to the large machinery assets, lack of visibility, user fatigue and controlling the proximity between the machines and workers. New developments in radio technology are providing solutions for supporting safety equipment to provide a safer environment for workers whilst also reducing insurance costs. Nanotron has provided an enabling technology for such collision avoidance systems (CAS) based on the concept of wireless *swarms*.

The concept of wireless *swarms*

Inspired from bees' behaviour, this concept sees each individual moving around an area as a member of a *swarm* network. Inside this network, individuals are able to communicate between their peers and to broadcast precise information such as their relative distances. With the nanotron *swarm*, individuals are represented by so called *swarm* radios. These radios are small autonomous electronic devices powered by a small lithium battery. Fitted with an antenna and a transceiver, they are able to interact within a *swarm* group thanks to a fast and robust over-the-air communication radio using Chirp Spread Spectrum (CSS) technology.

These radios are fully programmable and customizable from a computer or embedded uP, which acts as the host controller. A two-way serial communication or USB connection is used between radio and host controller. In a later example considered in this paper the communication is facilitated using a Bluetooth radio, where the *swarm* host application is running on a mobile phone.

Using a *swarm* specific set of commands, the host can control a swarm radio through the application interface (API) and set the control parameters to relevant values according to the specific environment where they will be used.

A PC can be used as the host controller for development testing and system evaluation purposes. With the *swarm* set of API commands, developers can easily develop their own user-interfaces (UI's). These UI's can be integrated in many different environments, such as smart phones or dashboards, adapted to the special requirements for the specific application.

The concept of CAS applications

Knowing the respective distances between objects is the key requirement of any Collision Avoidance System (CAS). The *swarm* concept of measuring distances between radios is therefore very well-suited for implementing such applications for protecting people, assets and vehicles within a defined area or site.

How does it work?

All objects that can collide in a given site-area have to be fitted with *swarm* radios. The radios are able to detect automatically their neighbours (1) to know proximity distance if another object is coming too close, (2) using their capacity to measure distances between each others within a *swarm* group. (3) Finally, if the system detects a potential collision, it triggers some visual or audio alarm to inform the individuals concerned of the imminent danger.
As these three steps make up a periodic CAS cycle, it is relevant to describe them in more detail.

1. Organizing the *swarm* group by detecting neighbours

Within a *swarm* network, the radios need firstly to make themselves visible to the other members by broadcasting their respective node ID. Setting their broadcast interval to 0.5 second, using the *SetBroadcastInterval=0.5* API command, will guarantee the radios good visibility inside the group. However, increasing this interval will free resources in the communication channel without affecting the proper running of the application in its steady state. To show the importance of selecting the correct broadcast interval, the default value in this example is initially set to a value of 3 seconds.

Other *swarm* radios are then instructed to automatically detect other radios coming nearby. Thanks to the *GetNodeIDList* command, the host application is able to know which radios are identified as neighbours of the one it controls. When a *swarm* radio is turned off, it does not broadcast its node ID anymore and after some seconds of inactivity, it will be automatically removed from the node ID list of the other *swarm* radios.

2. Detecting potential collisions by measuring distances

Getting the list of neighbours, the host application now has the required information to begin the ranging process. Going through this list, the application ranges to each node ID present in the list by doing subsequent calls to the *RangeTo* <*nodeID*> API command. Each valid distance result is stored by the application in a table containing each detected node ID and its range.

For this example, to reach the best possible accuracy, ranging results can be filtered and averaged over several values.

3. Prevent collisions by triggering an alarm

In the last step, these results have to be evaluated and compared to a definite safety distance D. This distance can be chosen by the user through the CAS application. The optimum value depends on the maximum speed of the vehicles on the site considered. In order to decrease the collision risk, this safety distance is then multiplied by different factors depending on the velocity of objects. (as shown on Table 1)

Vehicle	**To Vehicle**	**To Asset**	**To Person**
D=20m	**3***D = 60m	**1.5***D=30m	**2***D = 40m
Max Speed	50km/h	-	10km/h
Safe Time	2.2s	2.2s	2.4s

Table 1: Example of a safety distance guaranteeing almost the same safe time in the 3 different collisions' cases using the predefined factors.

If the calculated safety distance is violated, the CAS application triggers a visual and audio alarm to inform the individuals, of an imminent danger.

Choosing the best CAS cycle frequency

As part of the CAS cycle, these 3 steps are repeated at a periodic frequency, which can also be chosen through the application interface. The choice of this frequency has to be carefully chosen to insure the lowest possible collision's risk. As all *swarm* radios are sharing the same air interface in an asynchronous mode to communicate so collisions and congestions in the air may occur. That is why these potential congestions have to be taken into account, such as the numbers of *swarm* radios in the group. In order to reduce the possibility of congestion and increase the probability of a successful ranging cycles, the number of CAS cycles has to be maximized, without saturating the communication channel.

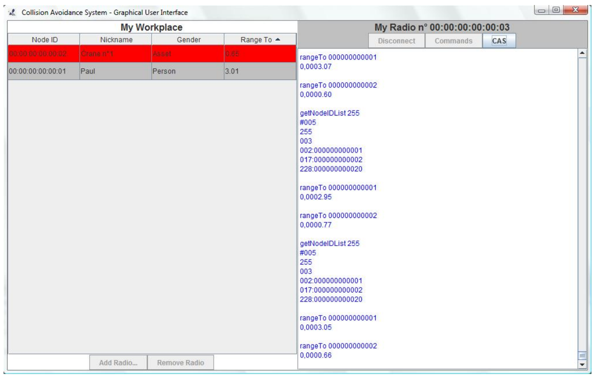
A whole CAS cycle takes about 2.2 milliseconds to be performed. This execution time has to be multiplied by the number of *swarm* radios running a CAS application. As a rule of thumb, this whole result should not exceed 17% of the available safe time before a collision occurs. This means that at some point, there is a trade-off between the number of *swarm* radios and the CAS cycle period. Care must be taken to have the correct compromise between these parameters to ensure a safe working environment.

As a way to create greater channel capacity, it is also possible in the swarm API host application to set levels of hierarchy and create star-networks. With this star-network one can arrange for certain nodes (for example nodes carried by pedestrians) not to range with each other. In this type of application the vehicles and pedestrians would range to each other (preventing collisions), likewise vehicles would also range with other vehicles, but pedestrians would not range with other pedestrians, since this does not pose a threat to safety.

Two running implementation examples

To illustrate the CAS application, nanotron created two demos.

In the *swarm* development environment a PC is used via USB interface as the host processor controlling the swarm radio. In a production environment the PC would be replaced by a host micro with a serial interface controlling the swarm radio. For this example, the CAS application is displayed in a window on the PC as a simple user interface which is split vertically in two parts. The left part lists all radios detected on the site in a table, in which rows are highlighted when a potential collision situation is detected. The right side focuses only on the *swarm* radio connected to the host. It contains a menu with the whole set of *swarm* API commands to allow the different parameters of the connected *swarm* radio to be defined as required. It also provides a CAS section, where safety distance and frequency can be set and where the CAS cycling process can be started.


Collision Avoidance System - Graphical User Interface
My Workplace
My Radio n° 00:00:00:00:00:03
Disconnect
Commands
CAS
Add Radio...
Remove Radio

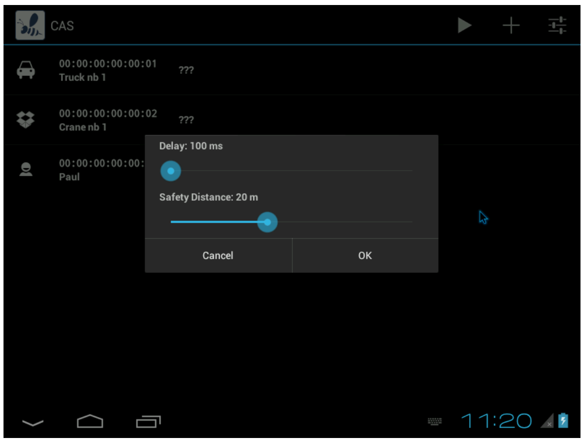
Node ID	Nickname	Gender	Range To
00:00:00:00:00:02	Crane n°1	Asset	0.65
00:00:00:00:00:01	Paul	Person	3.01

```
rangeTo 000000000001
0,0003.07

rangeTo 000000000002
0,0000.60

getNodeIDList 255
#005
255
003
002:000000000001
017:000000000002
228:000000000020

rangeTo 000000000001
0,0002.95

rangeTo 000000000002
0,0000.77

getNodeIDList 255
#005
255
003
002:000000000001
017:000000000002
228:000000000020

rangeTo 000000000001
0,0003.05

rangeTo 000000000002
0,0000.66
```

Figure 1: CAS PC application development screen.

Initially, the application would be used to configure the *swarm* radios, to understand the CAS concept and to test the various options with the radios before beginning trials on a site. Another interesting example for this development project was to replace the serial communication between host and swarm board with a Bluetooth link to increase the portability of the swarm module during testing.

Nanotron developed a second CAS application based on Bluetooth communication with a smart phone and mobile swarm devices. In this example the smart phone is running an Android application to display all of the *swarm* radios in the vicinity on test site. The user can then choose the safety distance required and the update frequency required. The application connects to a *swarm mini* radio, with its serial API interface connected to a serial Bluetooth module. The host application running on the smart phone is a Java application under Android, and was also providing a user-interface via the display on the phone.


CAS

00:00:00:00:00:01 ???
Truck nb 1

00:00:00:00:00:02 ???
Crane nb 1

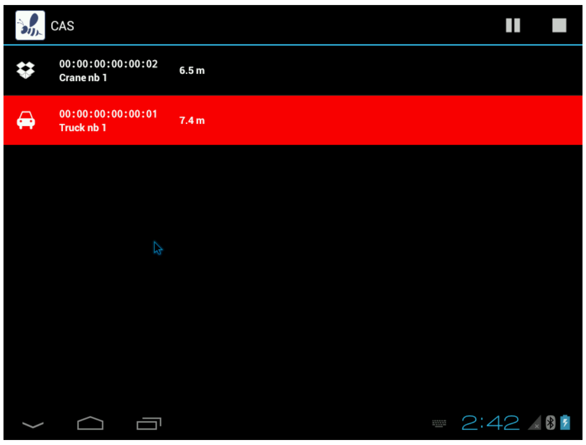
00:00:00:00:00:
Paul

Delay: 100 ms

Safety Distance: 20 m

Figure 2: Settings dialog of the Android application.

The user has access to the CAS application displayed on the smart phone which can be used to show ranging data, trigger alarms and highlight possible collision situations. This also provides the user a simple portable application.


CAS

00:00:00:00:00:02 6.5 m
Crane nb 1

00:00:00:00:00:01 7.4 m
Truck nb 1

Figure 3: Application screen on an Android phone.

For the sake of simplicity in this example, access to the control parameters of the *swarm* radios was not used. But this could be easily added if needed. Likewise dynamic broadcast intervals and safety zones are also possible.

To a large scale CAS application

At this time, both host applications were made to connect to one *swarm* radio at a time. To achieve a large-scale CAS application dealing with multi-connected nodes at the same time, another application would need to be developed to coordinate the simultaneous use of several of these demos working concurrently.

Contact the authors through info@nanotron.com by making reference to this white paper.