Defence Research and Development Canada   Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

REAL-TIME SIMULATIONS USING QUARC AND RT-LAB AND DEVELOPMENT OF A HARDWARE-IN-THE-LOOP INDOOR FACILITY FOR ROBOT FORMATIONS

Alexandre Morris, Pierre Gosselin

NUMERICA TECHNOLOGIES INC.
3420 rue Lacoste
Québec, QC G2E 4P8 Canada

Project Manager: Pierre Gosselin
Contract number (in full): W7701-073141/001/QCV PO 91929NG

Scientific Authority: Camille Alain Rabbath (418) 844-4000 ext. 4756

Defence R&D Canada – Valcartier
Contract Report
DRDC Valcartier CR 2009-499
September 2009

Canada

REAL-TIME SIMULATIONS USING QUARC AND RT-LAB AND DEVELOPMENT OF A HARDWARE-IN-THE-LOOP INDOOR FACILITY FOR ROBOT FORMATIONS

Alexandre Morris
Numerica Technologies inc.

Pierre Gosselin
Numerica Technologies inc.

Numerica Technologies inc
3420 Lacoste
Québec
G2E 4P8

Project Manager: Pierre Gosselin
Contract Scientific Authority: Camille-Alain Rabbath, Telephone 4756

Contract number (in full): W7701-073141/001/QCV PO 91929NG

Defence R&D Canada – Valcartier

Contract Report

DRDC Valcartier CR 2009 - 499

September 2009

Author

Alexandre Morris

Approved by

Camille-Alain Rabbath

Scientific Authority

Approved for release by

Christian Carrier

Chief Scientist

REAL-TIME SIMULATIONS USING QUARC AND RT-LAB AND DEVELOPMENT OF A HARDWARE-IN-THE-LOOP INDOOR FACILITY FOR ROBOT FORMATIONS

PREPARED FOR

DEFENCE R&D CANADA - VALCARTIER

R&D POUR LA DÉFENSE CANADA - VALCARTIER

QUEBEC, QC, G3J 1X5
CANADA

Contract No.: W7701-073141/001/QCV - PO 91929NG

Camille Alain Rabbath, Scientific authority

Tel: 418-844-4000 ext. 4756

CR 2009-499

By

A. Morris, P. Gosselin

NUMERICA TECHNOLOGIES INC.

3420 rue Lacoste

Québec, QC G2E 4P8 Canada

September 2009

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

List of Acronyms and Abbreviations

ALTAV: Almost-Lighter-Than-Air Vehicle

ARM: Advanced RISC Machine

CPU: Central Processing Unit

C++: General purpose programming language

DAFD: Decentralized Abrupt Fault Detector

Ethernet: Ethernet is a physical layer for wired networking technology

FireWire: The FireWire port is a serial bus interface standard for high-speed communications

FPS: Frame per Second

GCC: GNU Compiler Collection

GNU: GNU is a Unix-like operating system

GPS: Global Positioning System

HIL: Hardware in the Loop

I/O: Input / Output

Linux ARM: Linux operating system for embedded computers

OS: Operating System

QNX: QNX is a Unix-like operating system for embedded computers

QuadRotor: QuadRotor is and aircraft that is lifted by four rotors

RedHawk: RedHawk is a Linux based real-time operating system

RISC: reduced instruction set computer

Shared Memory: Shared Memory is memory that can be accessed by more than one program

TCP: Transport Control Protocol, this protocol is a commonly used network protocol

UAV: Unmanned Aerial Vehicles

UDP: User Datagram Protocol, this protocol is a commonly used network protocol

VRPN: Virtual-Reality Peripheral Network. This communication protocol is used to exchange data between two programs executed on the same computer

Wi-Fi: Wi-FI is a physical layer for wireless networking technology

WMR: Wheeled Mobile Robot

Abstract

This report will present two commercial software environments used to distribute and execute real-time simulations: QuaRC and RT-Lab. Both QuaRC and RT-Lab allow the user to develop simulation models using Matlab/Simulink and include hardware, such as data acquisition boards, to connect to real vehicles and systems. In addition, QuaRC can be used to program embedded systems such as wheeled mobile robots and aerial vehicles.

This report will present formation flight models that have been modified in order to be compliant to QuaRC or RT-Lab. The simulations are composed of six to ten unmanned aerial vehicles, or UAVs, following a commanded trajectory while maintaining a prescribed trajectory. Models presented also include abrupt fault detection and formation shape morphing on operator's request. Vehicle models and dynamics are based on almost lighter-than-air (ALTAV) vehicles, unicycles and quadrotor vehicles. Low-level controllers used to stabilize these UAVs are feedback linearization controllers. Formation controller is of leader-to-follower type. Simulation results are displayed in real-time on a three-dimensional viewer (X-Plane).

The feedback linearization controller has been implemented on an embedded computer on board a wheeled mobile robot (QBot). An infrared camera system (OptiTrack camera setup) is used to measure the QBot's position and orientation. This information is then sent from the base station to the wheeled mobile robot's embedded computer using a wireless link in order to close the low-level controller's loop.

This report will then present major differences between QuaRC and RT-Lab as well as advantages and inconvenient of using either software solution.

Résumé

Ce rapport présentera deux logiciels servant à exécuter des simulations distribuées en temps réel; QuaRC et RT-Lab. Ces deux logiciels permettent de développer des modèles de simulations en utilisant Matlab/Simulink. Ces logiciels permettent également d'exécuter des simulations incluant du matériel tel que des cartes d'acquisition de données. De plus, QuaRC permet de programmer des systèmes embarqués à bord de robots mobiles et de véhicules aériens.

Ce rapport présentera des modèles de vols en formations qui ont été modifiés pour être compatibles avec RT-Lab ou QuaRC. Les simulations présentées sont composées de six à dix véhicules suivant une trajectoire commandée tout en maintenant la géométrie prescrite. Les modèles présentés montrent des capacités essentielles au vol en formation telles que la détection et le recouvrement de fautes abruptes ainsi que la modification de la géométrie de la formation sur la demande de l'utilisateur. Les modèles de véhicules utilisés pour exécuter ces simulations sont des véhicules presque plus légers que l'air (ALTAVs), des unicycles et des quadrotors. Leur contrôleur de bas niveau est une boucle de rétroaction linéarisée. Le contrôleur utilisé pour maintenir la formation est de type leader-suiveur. Les résultats de simulation sont affichés en temps réel dans un engin graphique à trois dimensions (X-Plane).

Le contrôleur par rétroaction linéarisée a également été implanté à sur l'ordinateur embarqué d'un robot mobile (QBot). Un système de pistage par caméra infrarouge (Système de caméra OptiTrack) est utilisé pour mesurer la position et l'orientation du QBot en temps réel. Ces informations sont par al suite relayées de la station de base au QBot via un lien sans-fil pour fermer la boucle du contrôleur de bas niveau

Ce rapport présentera par la suite les différences majeures entre QuaRC et RT-Lab ainsi que les avantages et inconvénients d’utiliser l’un ou l’autre des logiciels.

Executive Summary

A group of flying vehicles following a requested trajectory while maintaining certain geometry (formation) can be useful to execute several kinds of mission: automated aerial refueling, coordinated bombing, territorial surveillance, multi-vehicle heavy lift, and low-altitude cruising by fleets of missiles. A formation is composed of 2 kinds of vehicle: a leader and followers. The leader receives objectives and commands from an operator while followers attempt to keep their commanded relative position from neighbors. Note that the operator can either be on board the leader or on a ground station.

This report will present Real-Time simulation models where the user can control de leader's trajectory. The user can be considered as an operator who sends high level commands to the leader from a base station. Note that complying to real-time constraints is important in order to demonstrate the feasibility of the implementation of the formation controller and the low-level controller on actual hardware.

The control of a wheeled mobile robot (QBot) has also been developed using the QuaRC software. Controllers are executed on board the QBot's embedded computer. Showing that control algorithms are simple enough to be implemented in real-time on embedded computers.

This report will present two commercial software environments used to distribute and execute real-time simulations: QuaRC and RT-Lab. Both QuaRC and RT-Lab allow the user to develop simulation models using Matlab/Simulink and include hardware, such as data acquisition boards, to connect to real vehicles and systems. In addition, QuaRC can be used to program embedded systems such as wheeled mobile robots and aerial vehicles.

This report will provide formation flight models that have been modified in order to be compliant to QuaRC or RT-Lab. The simulations are composed of six to ten unmanned aerial vehicles, or UAVs, following a commanded trajectory while maintaining a prescribed trajectory. Models presented also include abrupt fault detection and formation shape morphing on operator's request. Vehicle models and dynamics are based on almost lighter-than-air (ALTAV) vehicles, unicycles and quadrotor vehicles. Low-level controllers used to stabilize these UAVs are feedback linearization controllers. Formation controller is of leader-to-follower type. Simulation results are displayed in real-time on a three-dimensional viewer (X-Plane).

This report will then present major differences between QuaRC and RT-Lab as well as advantages and inconvenient of using either software solution.

Sommaire

Un groupe de véhicules volants suivant une trajectoire prescrite tout en maintenant une géométrie (formation) peut être utile pour plusieurs types de mission : le bombardement coordonné, la surveillance territoriale, la levée multi véhiculaire de poids lourd, et le vol à basse altitude d'une flotte de missiles de croisière. Une formation est composée de deux types de véhicules : Un maître et des suiveurs. Le maître est le véhicule qui reçoit les objectifs et les commandes de haut niveau provenant d'un opérateur alors que les suiveurs sont des véhicules qui tentent de se positionner par rapport aux véhicules voisins pour maintenir une géométrie.

Ce rapport présentera des modèles de simulations en temps réel où l'utilisateur peut commander la trajectoire du maître de la formation. L'utilisateur peut être vu comme l'opérateur qui envoie des commandes de haut niveau au maître à partir d'une station de terrestre. Il est à noter que le respect des contraintes de temps réel est important afin de démontrer la faisabilité de l'implantation des contrôleurs de formation et de bas niveau.

Le contrôle d'un robot mobile (QBot) a également été développé en utilisant le logiciel QuaRC. Les contrôleurs de ce robot mobile sont exécutés sur l'ordinateur à bord. Montrant ainsi que les algorithmes de contrôle sont suffisamment simples pour être implantés en temps réel sur un ordinateur embarqué.

Ce rapport présentera deux logiciels servant à exécuter des simulations distribuées en temps réel; QuaRC et RT-Lab. Ces deux logiciels permettent de développer des modèles de simulations en utilisant Matlab/Simulink. Ces logiciels permettent également d'exécuter des simulations incluant du matériel tel que des cartes d'acquisition de données. De plus, QuaRC permet de programmer des systèmes embarqués à bord de robots mobiles et de véhicules aériens.

Ce rapport présentera des modèles de vols en formations qui ont été modifiés pour être compatibles avec RT-Lab ou QuaRC. Les simulations présentées sont composées de six à dix véhicules suivant une trajectoire commandée tout en maintenant la géométrie prescrite. Les modèles présentés montrent des capacités essentielles au vol en formation telles que la détection et le recouvrement de fautes abruptes ainsi que la modification de la géométrie de la formation sur la demande de l'utilisateur. Les modèles de véhicules utilisés pour exécuter ces simulations sont des véhicules presque plus légers que l'air (ALTAVs), des unicycles et des quadrotors. Leur contrôleur de bas niveau est une boucle de rétroaction linéarisée. Le contrôleur utilisé pour maintenir la formation est de type leader-suiveur. Les résultats de simulation sont affichés en temps réel dans un engin graphique à trois dimensions (X-Plane).

Ce rapport présentera par la suite les différences majeures entre QuaRC et RT-Lab ainsi que les avantages et inconvénients d'utiliser l'un ou l'autre des logiciels.

1. Introduction

This report presents real-time simulations of formation flight models. Figure 1 presents two types of formation geometry. The first formation is a V-Shape formation and the second is a string formation. Formations are composed of 2 types of vehicles, Leaders (L) and Followers (F). Arrows represents the information flow. In this report it is assumed that the information flow contains the position and the speed of previous vehicles. It is also assumed that an operator located in a base station and can send high level commands such as trajectories, velocities command and orientation commands. to the leader. Followers attempt to maintain the formation's geometry by relying on the information flow available.

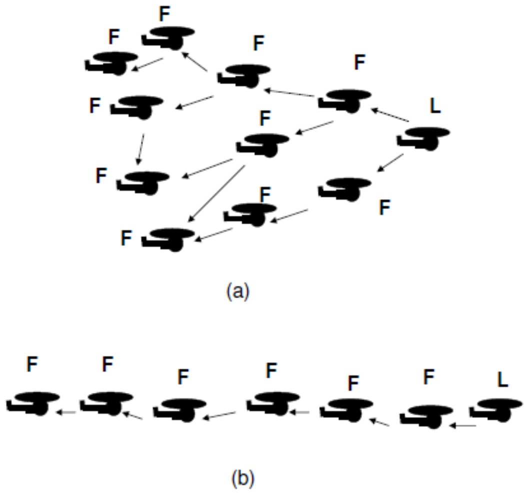


Figure 1: UAV Formation Example

Simulations presented in this report are based on ALTAV [6], unicycle [7] and quadrotor [8] models. Each UAV comprises onboard system and components such as actuators, control surfaces, engines, on board computers and wireless communication devices. It is assumed that the formation and low-level controllers (feedback linearization controller [9]) are executed onboard UAV computer and that the loop is closed by onboard sensors.

Real-Time simulations have been developed using QuaRC and RT-Lab. Both products allow developing models using Matlab/Simulink. Models can then be converted to C++ and compiled using the Real Time Workshop compiler. It is also important to note that these software solutions allow splitting models into smaller models in order to distribute the computation task on more than one computer.

The feedback linearization controller [9] has been implemented on an embedded computer onboard a wheeled mobile robot (QBot). An infrared camera system (OptiTrack camera setup [13]) is used to measure the QBot's position and orientation. This information is then sent from the base station to the WMR's embedded computer using a wireless link in order to close the low-level controller's loop. The controller is executed on board the QBot's embedded computer, showing that the controller is simple enough to be implemented in Real-Time on embedded computers.

This report will then present major differences between QuaRC and RT-Lab as well as advantages and inconvenient of using either software solutions.

2. Distributed Simulations Using RT-Lab

RT-Lab is software that allows developing simulation models that can be executed in real-time. Models are developed using Matlab/Simulink. Models can then be converted to C++ source code and compiled using RT-Lab's Real-Time-Workshop compiler. RT-Lab also offers to split models into smaller models in order to distribute the computation task to more than one computer or more than one CPU core. Each computation node (computer or core) are called a Target. For example, if a computer has 4 cores, it is possible to execute 4 different models (one on each core). This computer can then support 4 Targets.

Note that RT-Lab also offers target synchronization. Synchronization will ensure that simulation steps are executed periodically. Indeed, non synchronized simulation models will start a new computation step as soon as the previous one is finished. It is possible to synchronize targets via software or via hardware. Software synchronization is achieved using Target's CPU clock and hardware synchronization is achieved using an I/O board clock.

Note that this report will present software and hardware synchronized simulations. The hardware synchronization board is the NI6602 [14] board and has been installed on the computer called Target 1 (hardware specifications are available at the Annex 1).

Model separation is done entirely by RT-Lab. RT-Lab model separation is always composed of at least two targets: one target is called the console and the other is called the master. The console serves the purpose of giving an interface to the user with Real-Time Targets and is executed on the same computer that compiles models. For example, the joystick controller and data logging must be located in the console. The Target called the master is executed in Real-Time on a computer on which the OS is RedHawk (real-time linux) or QNX. Such targets will serve the purpose of executing models in real-time, acquire computation timings and send simulation results to the console.

Additional real-time targets can be added to the simulation. Those targets are optional, but can prove useful when simulation models require more computation power than one target can provide. Indeed, it is possible to distribute the computation task to additional targets called slaves and thus reducing computation requirements to the master target.

RT-Lab provides Simulink blocks that can be used for fast prototyping. For example, RT-Lab provides joystick blocks allowing the user to include a game controller to simulation models. RT-Lab also provides communication blocks allowing the user to exchange data in real-time between real-time targets. RT-Lab supports following communication protocols: Ethernet/UDP, FireWire and Shared Memory.

Real-Time models can be executed on computers with the following operating systems: RedHawk, QNX.

For more information about RT-Lab, refer to the Annex 2: RT-LAB Quick Start Guide.

RT-Lab computers are presented in Annex 1: Presentation of the Hardware (Table 4, Table 5 and Table 6).

2.1. Altav Convoy model

This model presents a 9-vehicle formation. The first vehicle is called the leader. Other vehicles are called followers. The formation geometry for this simulation is a string shape. For example: the follower 1 (UAV 2) will stay behind the leader, the follower 2 (UAV 3) will stay behind the follower 1 and so on. The leader follows trajectory determined by the user. The user can control the leaders' heading, speed and altitude using a joystick. ALTAV's low level controller is the feedback linearization controller [9]. The formation controller used is detailed in [11]. This model also allows the user to trigger an abrupt fault on the UAV 2 at any time during the simulation. UAV 3 will detect the anomaly using the DAFD [10] and will recover the fault by following the leader of the formation instead of following the UAV 2. Simulation results can be viewed in a three dimensional viewer in Real-Time using X-Plane 8.4 [4] software. Note that this simulation model is executed at a period of 0.01 s and is software synchronized.

The ALTAV model presented in this report is based on the original ALTAV model [6] available at the DRDC-Valcartier Precision Weapon section.

Following model has been delivered to the Precision Weapons Section at DRDC-Valcartier:

- RT-Lab:ALTAVconvoyTeamFDRSummer2007joystick(DAFD)

2.1.1. Objectives

The ALTAV model was already an RT-Lab compliant model. It as however been modified in order to allow the user to control the leader of the formation with a joystick more easily. Previously, the joystick was controlling directly the X-Y trajectory corresponding to the X-Y values of the joystick output. Now the user controls the velocity and the heading vehicles allowing a smoother control of the leader.

2.1.2. Block Diagram

This model is composed of two targets; APSEQUANSER and Target 1 computers. APSEQUANSER is a non Real-Time windows target that allows the user to control the leader with a joystick and trigger an abrupt fault on the UAV 2. The console also logs simulation results and computation time. The Target 1 executes UAVs 1 to 9 models, low-level controllers (feedback linearization controller [9]) and formation controllers [11] for each vehicles. The Target 1 also receives trajectory commands and fault trigger from the console using Ethernet/UDP. The UAV 3 model also includes an abrupt fault detector (DAFD [10]) that can detect abrupt fault on the UAV 2. Note that the fault DAFD only uses the UAV 2 X and Y position. Simulation results are also displayed on a three dimensional viewer (X-Plane 8.4 [4]) that is executed on the computer APSERTVIEW. Note that simulation results are transferred in Real-Time from the Target 1 to APSERTVIEW using Ethernet/UDP.

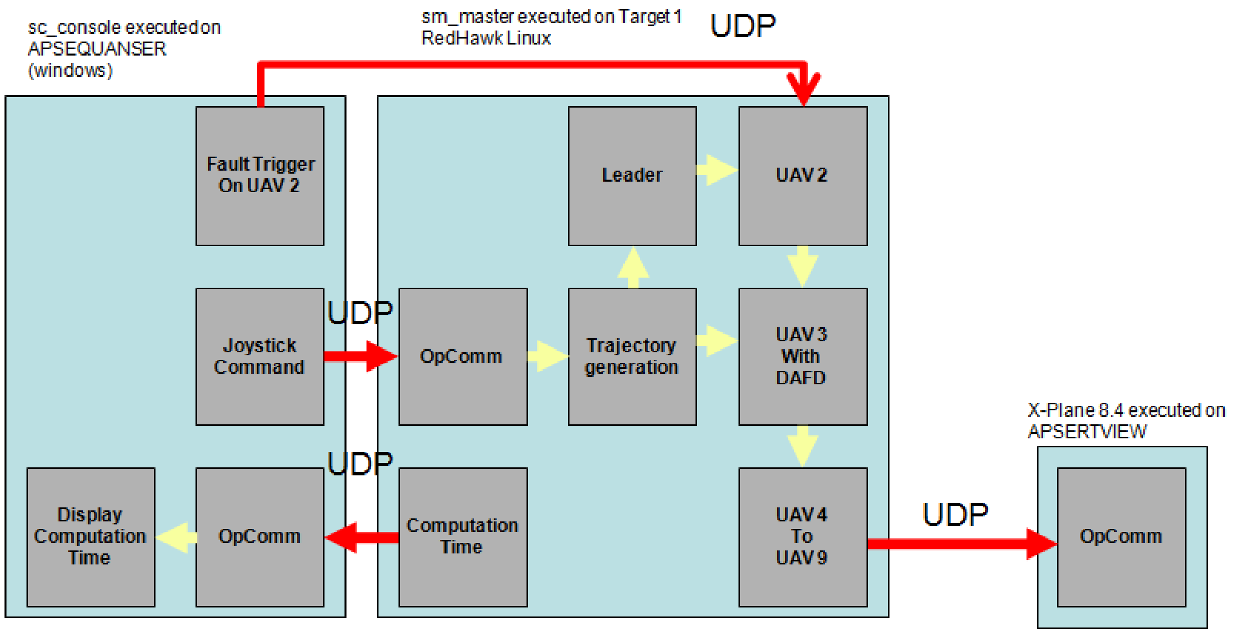


Figure 2: RT-Lab **ALTAV** Model

2.1.3. Joystick Control

The Figure 3 shows the joystick controller developed for the ALTAV. A hysteresis has been added to the altitude and the heading. This hysteresis will allow the user to keep the same heading and altitude without requiring the joystick to be perfectly centered. The altitude will increase if the user pull the joystick toward him or decrease if the user pushes the joystick.

The trajectory is then calculated as follows:

$$X_d = \int_0^T V(t)\cos(\theta(t))dt$$

$$Y_d = \int_0^T V(t)\sin(\theta(t))dt$$

Where: V(t) is the requested velocity (Joystick's throttle command), $\theta(t)$ the requested heading and T is the current simulation time. Note that the requested heading $\theta(t)$ will increase if the joystick is steered to the left and decrease if the joystick is steered to the right. The requested heading will remain still otherwise.

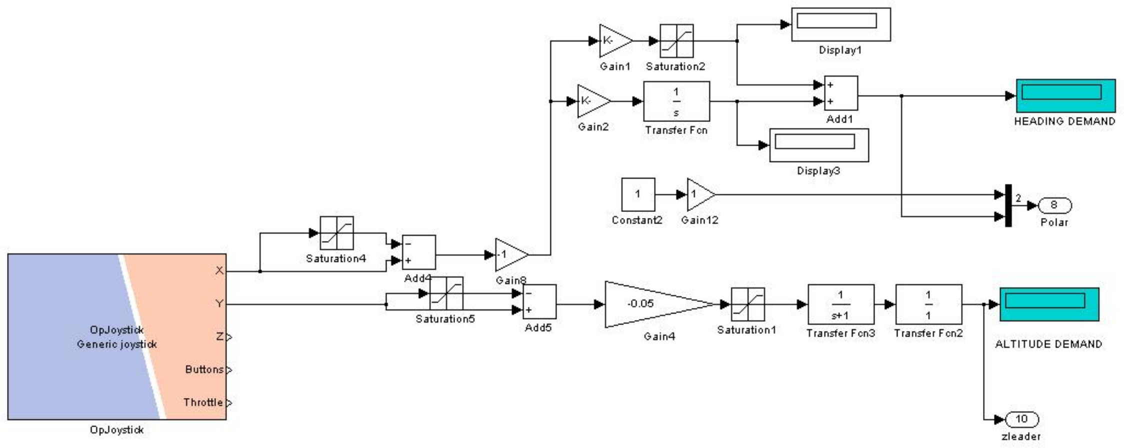


Figure 3: **ALTAV** Joystick Controller

2.2. RT-Lab QuadRotor Models

This model presents a 6-vehicle formation maintaining a V-Shape geometry. The first vehicle is called the leader. Other vehicles are called followers. Followers will follow their immediate predecessors. As shown on Figure 4, UAVs 1 and 2 follow the Leader, UAVs 3 and 4 follow the UAV 1 and the UAV 5 follows the UAV 2. The user can control the leaders' heading, speed and altitude using a joystick. The user can also trigger an abrupt fault on the follower 2. Follower 5 will then detect the fault (using the DAFD [10]) and follow the leader instead of following the Follower 2. The low level controller is a feedback linearization controller [9] and the formation controller described in [11] is used to maintain the geometry of the formation. Simulation results are sent to X-Plane 8.4 [4] in real-time using Ethernet/UDP. Note that this simulation model is executed at a period of 0.001 s.

Note: The QuadRotor model presented in this report is based on the original QuadRotor model [8].

The following models have been delivered to the Precision Weapons Section at DRDC-Valcartier:

- Formation_6QuadRotors_Fault_fault_detect.mdl (Synchronized by software)
- Formation_6QuadRotors_Fault_joystick.mdl (Synchronized by software)
- Formation_6QuadRotors_Fault_joystick_HWSynchro.mdl (Synchronized by hardware using the NI6602 board)

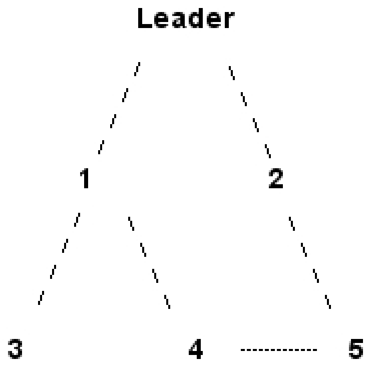


Figure 4: QuadRotor formation topology (lines indicate information flow)

2.2.1. Objectives

- Transform the original Simulink model to a RT-Lab compliant model
- Send the data in real-time to X-Plane (3d environment viewer)
- Allow the user to use the joystick to control the formation
- Allow the user to trigger a fault on the follower 2 (The fault will be detected by the Follower 5)
- Obtain computation time of the model on RT-Lab

2.2.2. Block Diagram

This block diagram presents how the model is distributed. The simulation was distributed on one Real-Time target and one Non-Real-time Target (Console). The model named sc_console has the task to acquire joystick commands, allow the user to trigger a fault on the UAV 2 and log the simulation statistics (Simulation Step Size, Communication statistics). Each of the 6 UAVs is simulated on the model sm_master. The computer named Target 1 is executing the model sm_master. The computer named APSEQUANSER executes the model called sc_console. The computer named APSERTVIEW displays simulation results in Real-Time on X-Plane 8.4 [4]. Note that simulation results are sent to X-Plane from the Target 1 using an Ethernet/UDP link. For more information computers hardware, see the Annex 1: Presentation of the Hardware.

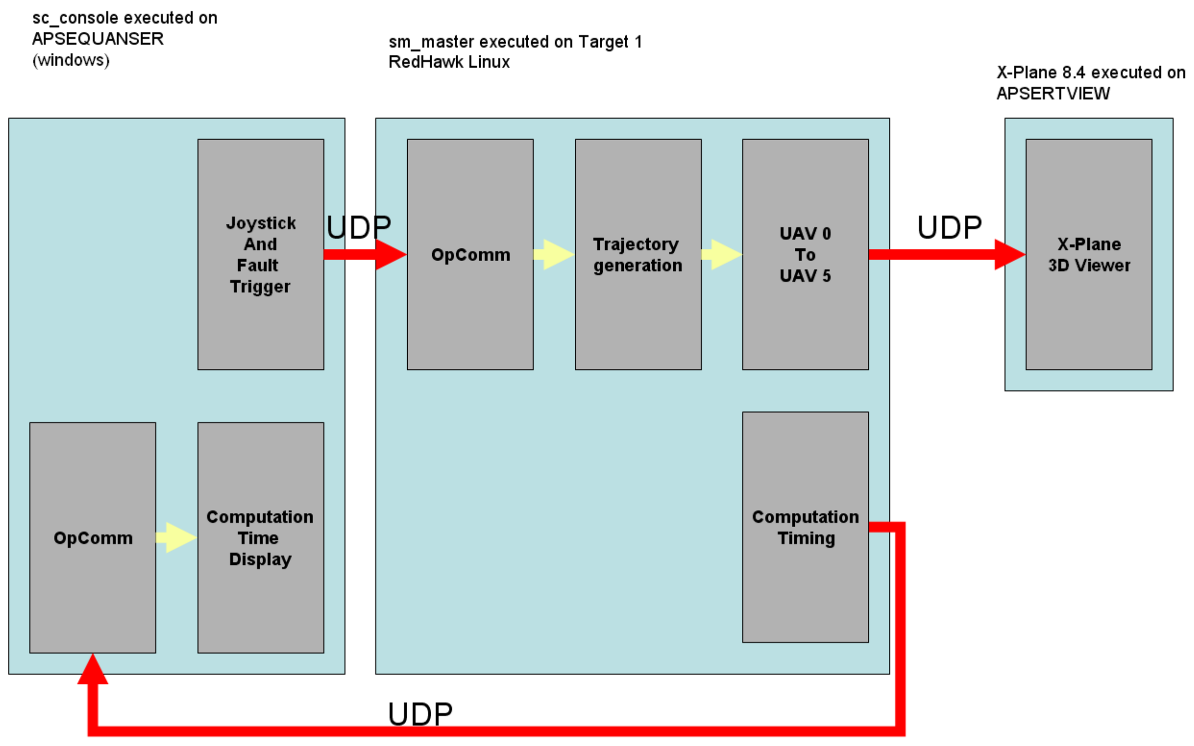


Figure 5: RT-Lab QuadRotor model

2.2.3. Joystick Control

The Figure 6 shows the joystick controller for the QuadRotor. A hysteresis has been added to the altitude and the heading. This hysteresis will allow the user to keep the same heading and altitude without requiring the joystick to be perfectly centered. The altitude will increase if the user pull the joystick toward him or decrease if the user pushes the joystick. The heading will increase or decrease if the user steer the joystick. The requested velocity will be 1.1 m/s if the formation is flying in a straight line or 2 m/s if the formation is turning. This special velocity command will ensure that the followers will keep tracking.

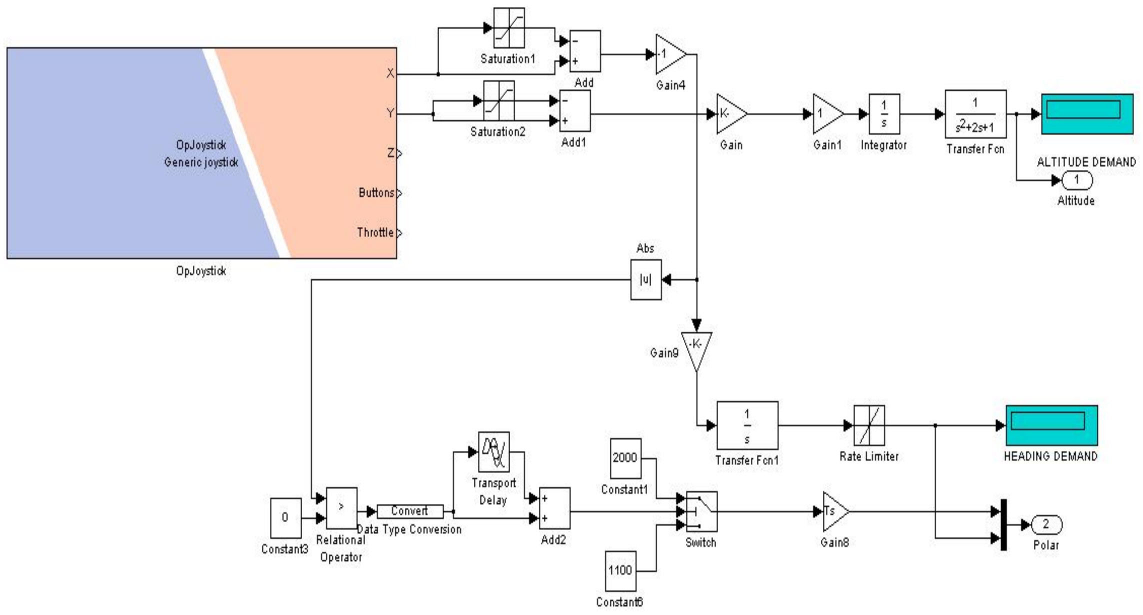


Figure 6: QuadRotor Joystick Controller

The trajectory is then calculated as follow:

$$X_d = \int_0^T V(t)\cos(\theta(t))dt$$

$$Y_d = \int_0^T V(t)\sin(\theta(t))dt$$

Where: V(t) is the requested velocity (Joystick's throttle command), $\theta(t)$ the requested heading and T is the current simulation time. Note that the requested heading $\theta(t)$ will increase if the joystick is steered to the left and decrease if the joystick is steered to the right. The requested heading will remain still otherwise.

2.2.4. Results

The Figure 7 and the Figure 8 show the computation time obtained in RT-Lab. All these values are in µs.

These 5 plots represent:

- Time spent by the RedHawk target to receive data
- Time spent by the RedHawk target to send data
- Model calculation time (Computing only)
- Effective step size (Computing + synchronization)
- Total step size (Computing + overhead + synchronization)

The Figure 7 shows the computation time for the QuadRotor model executed with a hardware synchronization. We can see that the communication timing is set to 0 which means this measure is considerably small. The total step size is around 1000 µs, which is normal because the sampling time is set to 1 ms. The minimum and maximum total step size are 998 µs to 1002 µs which mean that the jitter is ± 2 µs. The model calculation time and the effective step size are varying from 60 µs to 85 µs which means the step size could be reduced to 90 µs without risking having overruns in the simulation.

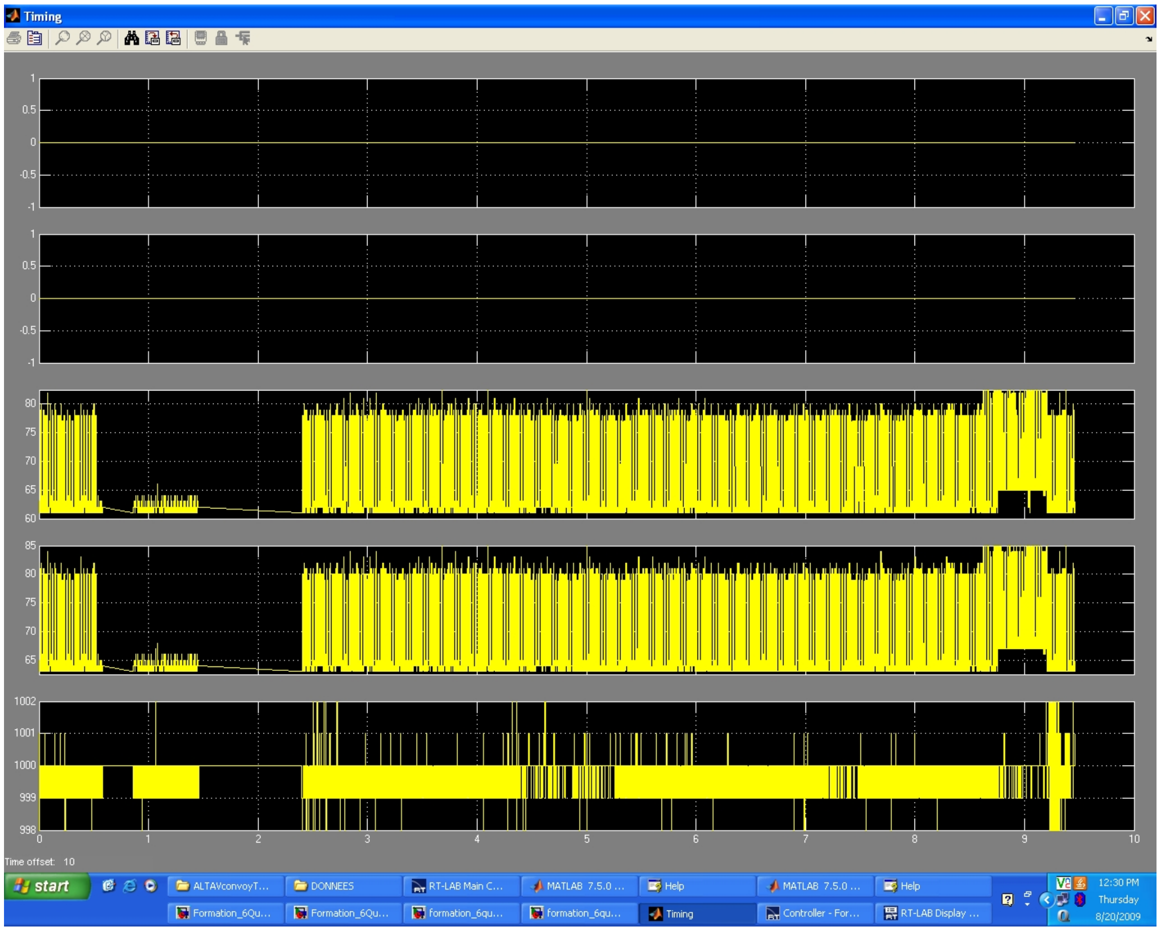


Figure 7: QuadRotor model with hardware synchronization computation timing

Figure 8 shows the computation time for the QuadRotor model executed with a software synchronization. We can see that the communication timing is set to 0 which means this measure is considerably small. The total step size is around 1000 µs, which is normal because the sampling time is set to 1 ms. The minimum and maximum total step size are 998 µs to 1001 µs which mean that the jitter is ± 2 µs. The model calculation time and the effective step size are varying from 60 µs to 85 µs which means the step size could be reduced to 90 µs without risking having overruns in the simulation.

Note that the total step size seems to be more stable with the hardware synchronization.

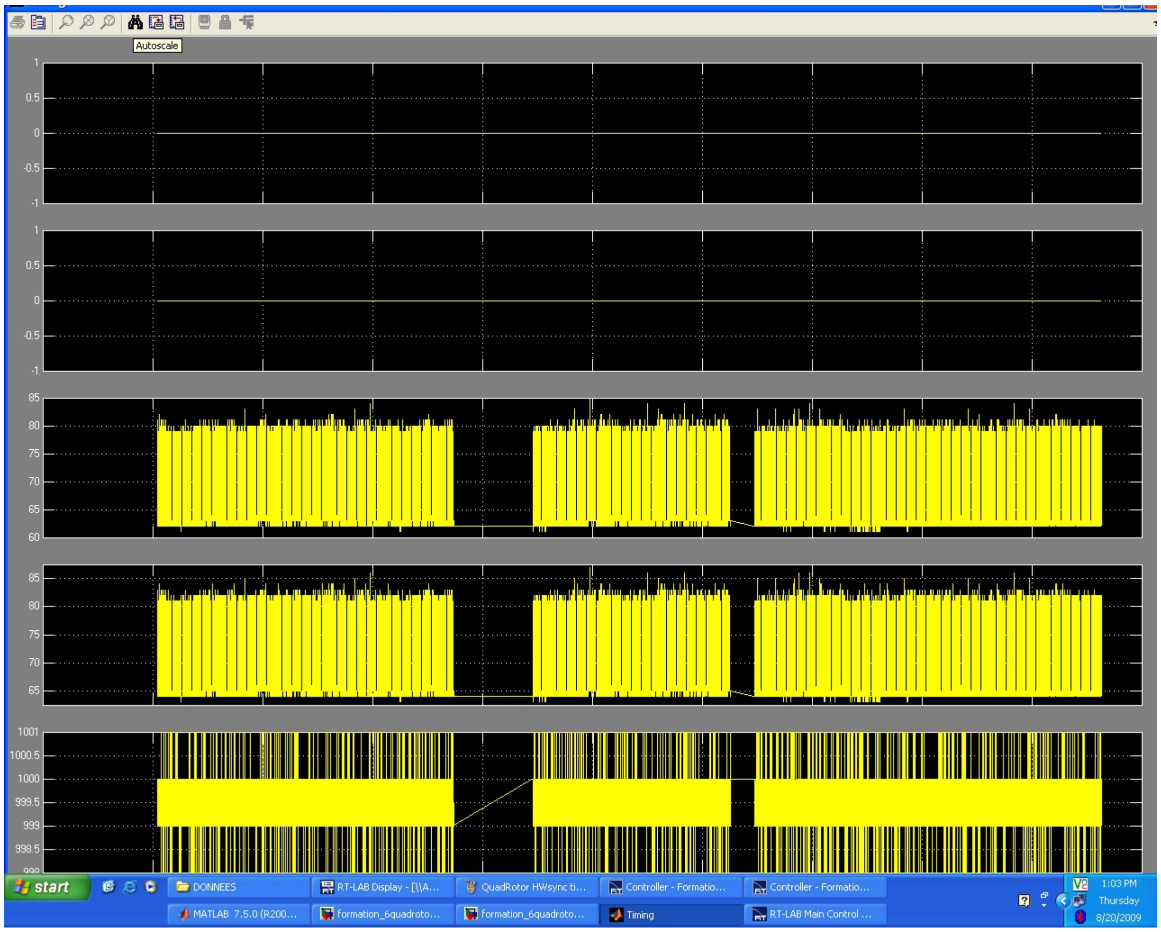


Figure 8: QuadRotor model with software synchronization computation timing

2.3. RT-Lab AFGsummer2007

This model presents a 6-vehicle formation. The first vehicle is called the leader. Other vehicles are called followers. Followers will follow their immediate predecessors in order to maintain a V-Shape geometry. The user can control the leaders’ heading, speed and altitude using a joystick. The user can also order the formation to perform a shape morphing. Indeed, the user can order vehicles to increase or decrease distance between neighbors. Low-level controller are feedback linearization controller [9] and the formation controller described in [11] has been used to maintain the formation geometry. Simulation results are displayed in Real-Time using a three dimensional viewer (X-Plane 8.4 [4]). Simulation results are sent from a Real-Time target to the X-Plane computer using an Ethernet/UDP link. Note that this simulation model is executed at a period of 0.001 s.

The unicycle model presented in this report is based on the original unicycle model [7].

The following models have been delivered to the Precision Weapons Section at DRDC-Valcartier:

- RT-Lab:AFG_summer2007joystick (Software synchronized)
- RT-Lab:AFG_summer2007joystick2CPUs (Software synchronized)
- RT-Lab:AFG_summer2007joystickHWsynchro (Hardware synchronized)

2.3.1. Objectives

The objectives of that model are:

- Ensure that models were still working in RT-Lab.
- Obtain the computation timings
- Show the simulation results in real-time on X-Plane

Note: The model can send simulation data to X-Plane in real-time. This feature is not shown in this document.

2.3.2. Block Diagram

The Figure 9 presents a block diagram of the distributed unicycle simulation. The simulation was distributed on two Real-Time targets and one Non-Real-time Target (Console). The model named sc_console has the task to generate a trajectory from the joystick command and log simulation statistics (Simulation Step Size, Communication statistics). The leader and UAVs 1 to 5 are simulated on the model sm_master. UAVs 6 to 9 are simulated on the model ss_slave. The computer named Target 1 is executing the model sm_master and the computer named Target 2 is executing the model ss_slave. The computer named APSEQUANSER executes the model called sc_console. The computer named APSERTVIEW is used to display simulation results in Real-Time using X-Plane 8.4 [4]. For more information computers hardware, see the Annex 1: Presentation of the Hardware.

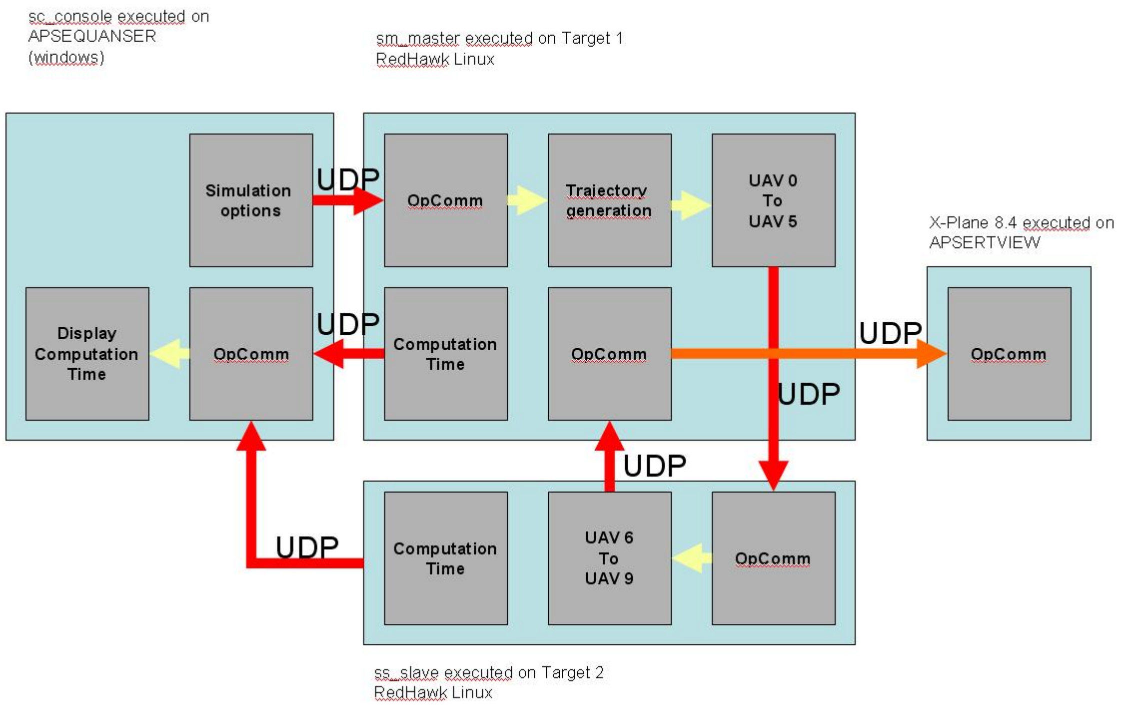


Figure 9: RT-Lab AFG summer 2007 distributed on 2 CPU block diagram

The Figure 10 presents a block diagram of the unicycle simulation. The simulation was distributed on one Real-Time target and one Non-Real-Time Target (Console). The model named sc_console has the task to generate a trajectory from the joystick command and log simulation statistics (Simulation Step Size, Communication statistics). The leader and UAVs 1 to 8 are simulated on the model sm_master. The computer named Target 1 is executing the model sm_master. The computer named APSEQUANSER executes the model called sc_console. The computer named APSERTVIEW is used to display simulation results in Real-Time using X-Plane 8.4 [4]. For more information computers hardware, see the Annex 1: Presentation of the Hardware.

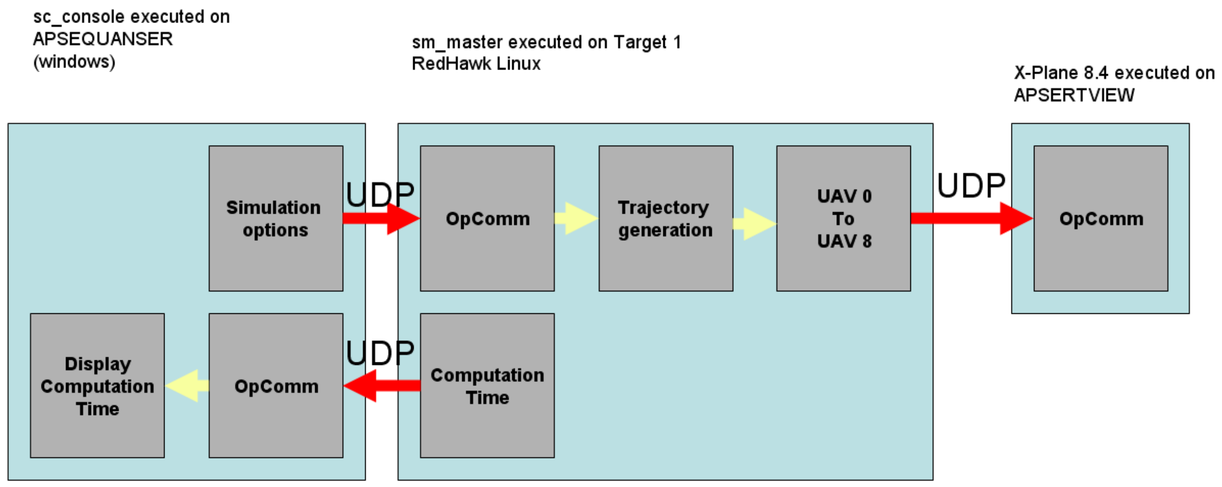


Figure 10: RT-Lab AFG summer 2007 block diagram

2.3.3. Results

The Figure 11 shows the computation timing for the AFGSummer 2007 model with software synchronization.

These 5 plots represent:

- Time spent by the RedHawk target to receive data
- Time spent by the RedHawk target to send data
- Model calculation time (Computing only)
- Effective step size (Computing + synchronization)
- Total step size (Computing + overhead + synchronization)

We can see that the communication timing is set to 0 which means this measure is considerably small. The total step size is around 1000 µs, which is normal because the sampling time is set to 1 ms. The minimum and maximum total step size are 998 µs to 1001 µs which mean that the jitter is ± 2 µs. The model calculation time and the effective step size are varying from 40 µs to 80 µs which means the step size could be reduced to 85 µs without risking having overruns in the simulation.

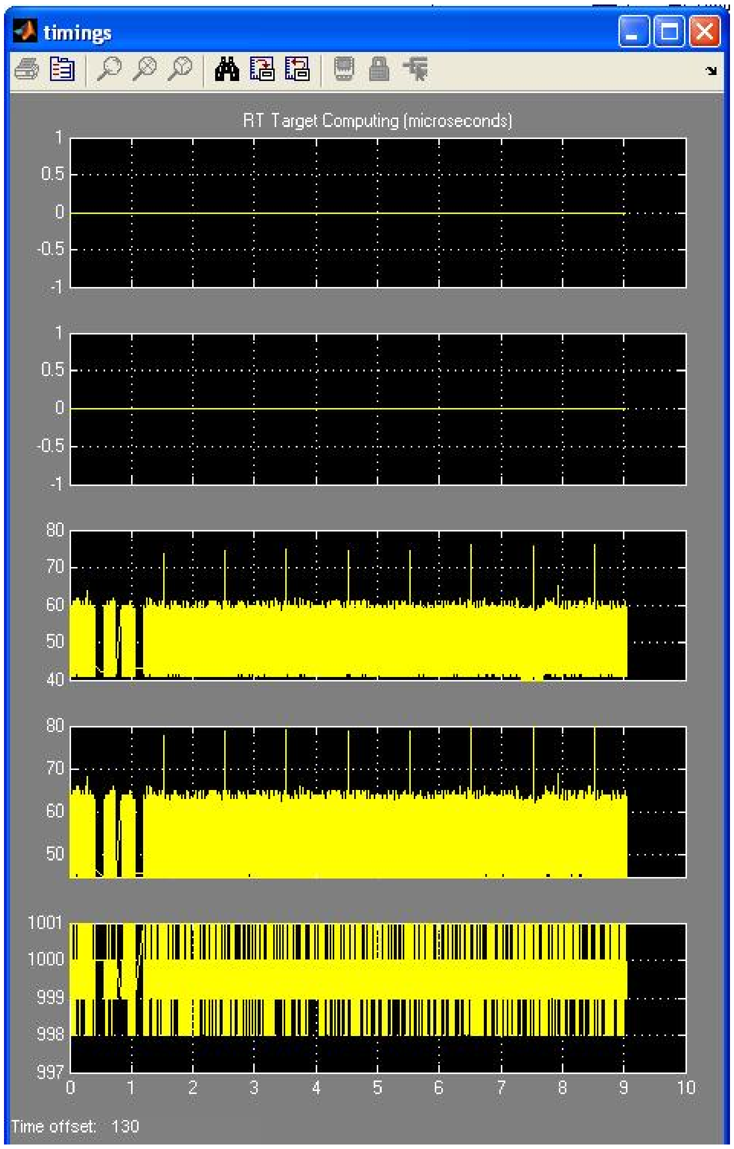


Figure 11: RT-Lab timings for AFGSummer2007

Figure 12 shows the computation timing for the AFGSummer 2007 model with hardware synchronization.

We can see that the communication timing is set to 0 which means this measure is considerably small. The total step size is around 1000 µs, which is normal because the sampling time is set to 1 ms. The minimum and maximum total step size are 988 µs to 1010 µs which mean that the jitter is ± 12 µs. The model calculation time and the effective step size are varying from 40 µs to 80 µs which means the step size could be reduced to 85 µs without risking having overruns in the simulation.

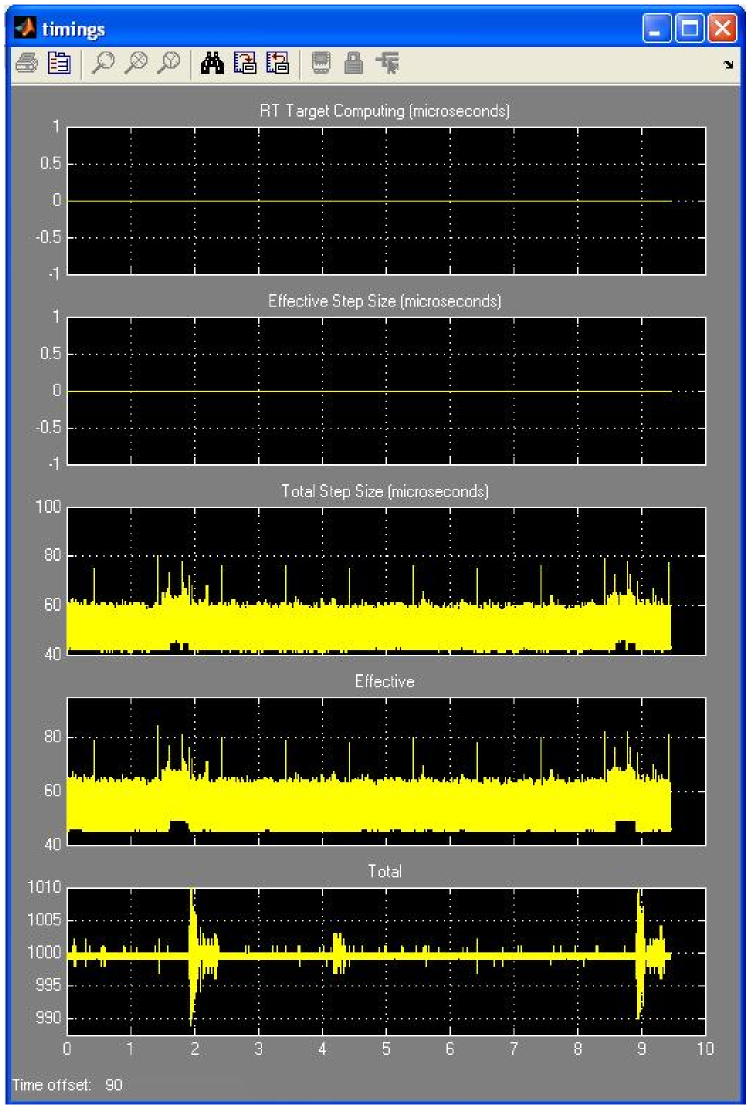


Figure 12: RT-Lab timings for AFGSummer2007 with hardware synchronization

Figure 13 shows the computation timing for the AFGSummer 2007 model distributed on 2 computers with software synchronization. On the left side, the SM_Master timings are shown. On the right side, the SS_Slave timings are shown.

The computing time of the master target reaches 70 µs and the slave reaches 18 µs. The effective step size of the master reaches 75 µs and the slave's is the same as the effective step size since the slave is synchronizing with the master.

The minimum and maximum total step size of the slave is 987 µs to 1013 µs which mean that the jitter is ± 13 µs. The minimum and maximum total step size of the maser is 990 µs to 1010 µs which mean that the jitter is ± 10 µs.

These data shows that the step size could be lowered to 80 µs. The simulation step size could be lowered when distributing models.

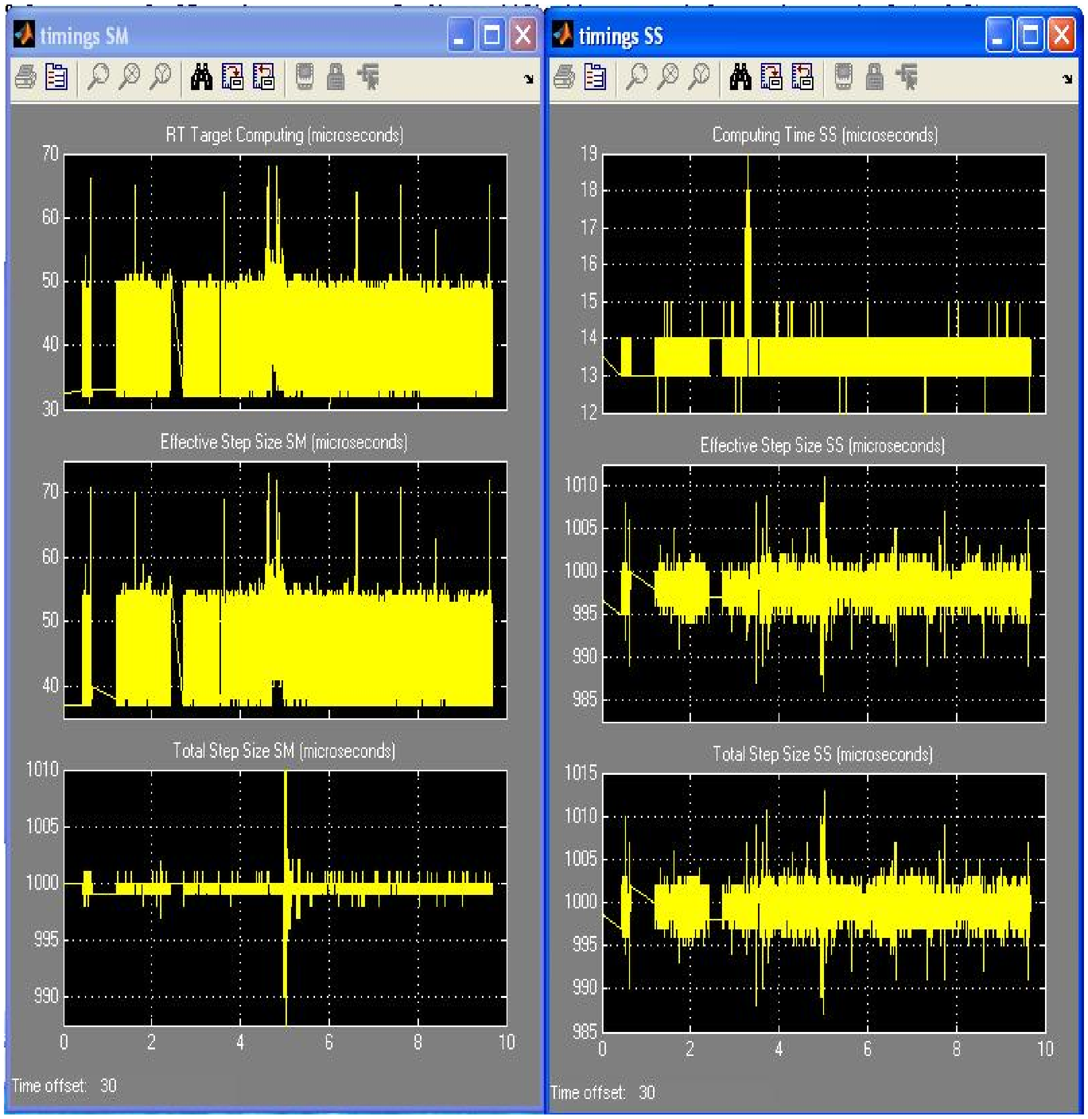


Figure 13: RT-Lab timings for AFGSummer2007 distributed on 2 CPUs

3. QuaRC Models

The QuaRC software allows developing simulation models using Matlab/Simulink. Models can then be converted to C++ and compiled using the QuaRC Real-Time-Workshop compiler. Each QuaRC Models can be executed in real-time or in a regular Simulink simulation. Note that a QuaRC Target is simply a computer on which QuaRC is installed. Note that more than one model can be executed on the same Target (or same computer). Indeed, a computer with one core can execute 2 models simultaneously in Real-Time.

QuaRC provides Simulink blocks that can be used for fast prototyping. For example, QuaRC provides joystick blocks allowing the user to include game controllers to models. QuaRC also provides communication blocks allowing data to be exchanged between QuaRC Targets. QuaRC communication blocks can also be used to exchange data between QuaRC targets and non QuaRC targets.

Note that users must separate manually models and configure communication blocks in order to distribute models. Indeed, unlike RT-Lab, QuaRC users must configure communication parameters manually such as IP addresses and port numbers in order to exchange data between models.

In RT-Lab, Targets can be console, master or slaves (refer to the section 2 for more information). In QuaRC, there is no such a distinction. Indeed, QuaRC models do not require a Non Real-Time model to be executed in order to change simulation parameters. Simulation parameters can be changed directly in the Simulink model during the execution.

Real-Time models can be executed on computers with the following operating systems:

- Windows
- QNX
- Embedded QuaRC Target (Linux ARM)

3.1. QuaRC QuadRotor Model

This model presents QuadRotors executing a V-Shape formation flight. The formation is composed of 10 QuadRotors. The user can control the leader's velocity, heading and altitude using a joystick. Followers will attempt to follow the leader and maintain the prescribed geometry. The user can also trigger an abrupt fault on the UAV 3. The UAV 5 will then detect the fault on the UAV 2 using the DAFD [10] and follow the leader instead of following the UAV 3. Low level controllers used to stabilize vehicles are feedback linearization controllers [9]. The formation controller used to maintain the formation geometry is given in [11]. Note that this simulation model is executed at a period of 0.001 s.

This model has been delivered to the DRDC-Valcartier Precision Weapons Section.

This simulation model is based on the original Simulink QuadRotor model [8].

3.1.1. Objectives

The objectives of that model are:

- Simulate the QuadRotor model on a Windows Real-Time Target
- Allow the user to use the joystick to control the leader of the formation
- Show that the fault on the UAV 2 can be detected an recovered by the formation using the DAFD [10] on board the UAV 5
- Send the data in Real-Time to X-Plane (3d environment viewer)

3.1.2. Block Diagram

The Figure 14 presents the QuaRC QuadRotor block diagram. A joystick is connected to the computer APSEQUANSER. The user can use the joystick to control the trajectory of the leader. Followers will attempt to follow the leader and hold a V-Shape geometry as shown on the Figure 15. The computer APSEQUANSER will then send simulation results to the computer APSERTVIEW using an Ethernet/UDP link, where UAVs are displayed in Real-Time on X-Plane 8.4 [4] .For more information computers hardware, see the Annex 1: Presentation of the Hardware.

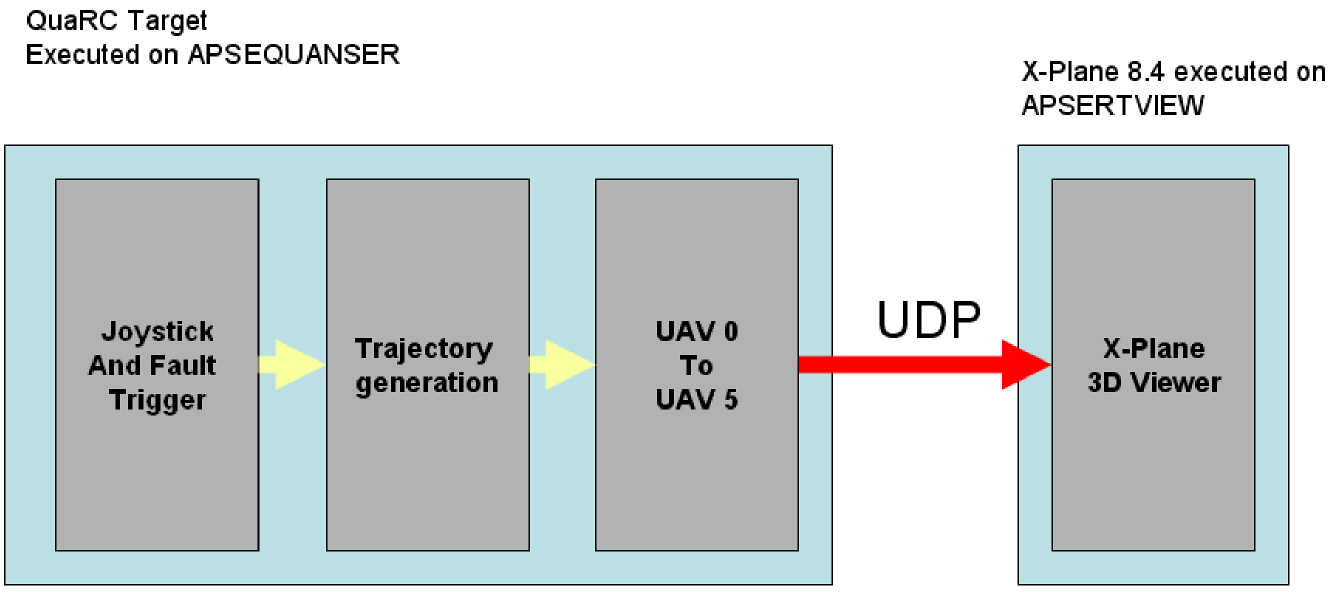


Figure 14: QuadRotor Block Diagram

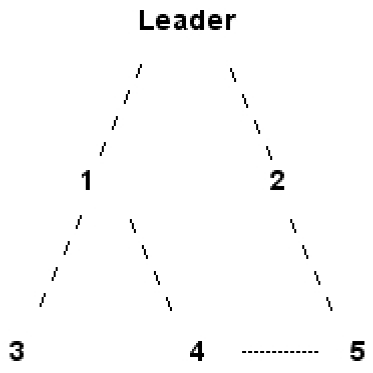


Figure 15: QuadRotor fleet topology (lines indicate information flow)

3.1.3. Plots

Figure 16 to Figure 18 show the 2D trajectory of simulated QuadRotors at different simulation instants. The black dashed line represents the leader's requested trajectory. The blue dotted line represents the trajectory of the UAV 3 (follower 2) and the blue dashed-dotted line represents the UAV 5 trajectory. An abrupt fault has occurred on this UAV after 79 second. On the Figure 17, we can see that the UAV 5 has detected the fault and maintain his relative position. On the Figure 18, we see that the UAV 5 now follows the Leader. The UAV 5 detected a fault on the UAV 2 using only UAV 2's X-Y position and speed and the DAFD [10]. It is important to note here that the trajectory was generated by a joystick and that the formation shape has been maintained using the formation controller of [11]. A fault has been trigged on the UAV 3 at a random time by the user and has been detected in real-time during the simulation using the DAFD [10] (abrupt fault detector). The formation has then adapted its topology to recover from the fault.

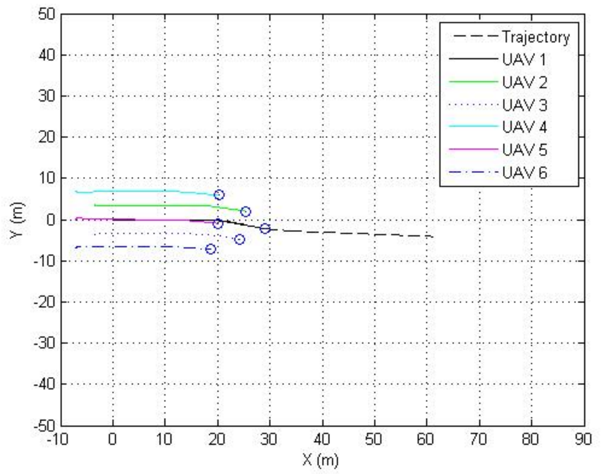


Figure 16: QuadRotor X-Y Trajectory at 79 second of simulation

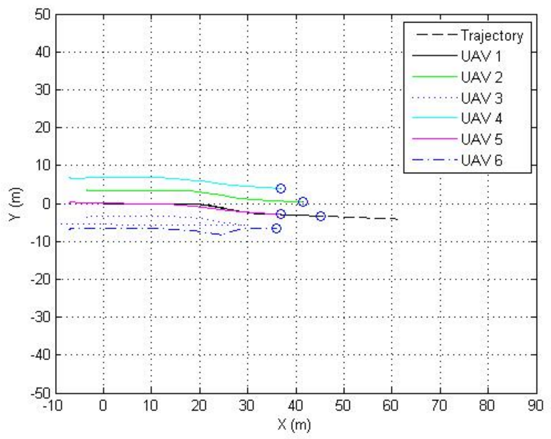


Figure 17: QuadRotor X-Y Trajectory at 118 second of simulation

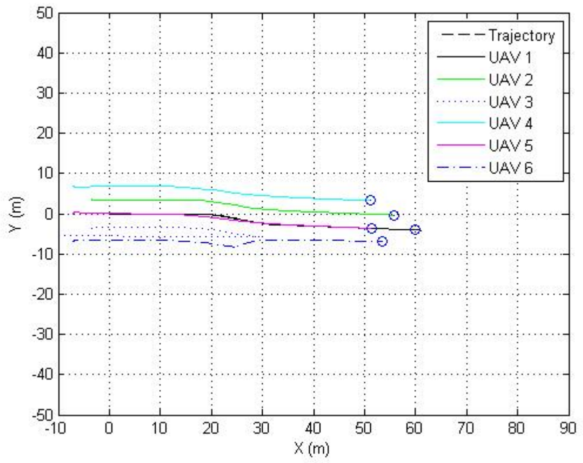


Figure 18: QuadRotor X-Y Trajectory at 158 second of simulation

3.2. QuaRC Distributed QuadRotor model

This model presents six QuadRotor executing a V-Shape formation flight. The user can control the leader’s velocity, heading and altitude using a joystick. Followers will attempt to follow the leader and maintain the formation geometry. This simulation is constituted of three Real-Time models executed on 2 Real-Time targets. The model is distributed as follow: The console allows the user to control the leader’s trajectory, trigger a fault on the UAV 3. The console also sends simulation data to X-Plane 8.4 [4] in Real-Time using an Ethernet/UDP link. In this simulation, the leader is executed on the first target and followers are executed on the second target. The UAV 5 can detect abrupt faults on the UAV 2 using the DAFD [10] and recover by following the leader instead of following the UAV 2. The QuadRotor low-level controller is the feedback linearization controller in [9]. The V-shape geometry is assured by the formation controller [11]. Note that this simulation model is executed at a period of 0.001 s.

This model has been delivered to the DRDC-Valcartier Precision Weapons Section.

3.2.1. Objectives

- Distribute the computing power
- Separate the leader model from the followers
- Send the data in Real-Time to X-Plane (3d environment viewer)

3.2.2. Block Diagram

This block diagram presents how models and targets are organized to execute the distributed QuadRotor simulation. Note that the vehicles are performing a V-Shape formation as shown on the Figure 15. The simulation was distributed on 3 targets. The model named console has the task to generate the leader's commanded trajectory and send simulation results to X-Plane. The leader of the formation is executed on the model named Target 1 and followers are executed on the model named Target 2. The computer named APSENLEVHI is executing the console and the Target 2 model. The computer named APSEQUANSER is executing the Target 1 model. Simulation results are sent in Real-Time using an Ethernet/UDP link from the console model to the three dimensional viewer: X-Plane 8.4 [4].

For more information computers hardware, see the Annex 1: Presentation of the Hardware.

Note that in this configuration, there are three Real-Time models (Console, Target1 and Target2) executed on two Real-Time Targets (APSENLECHEVI and APSEQUANSER). X-Plane 8.4 is executed on a third computer called APSERTVIEW.

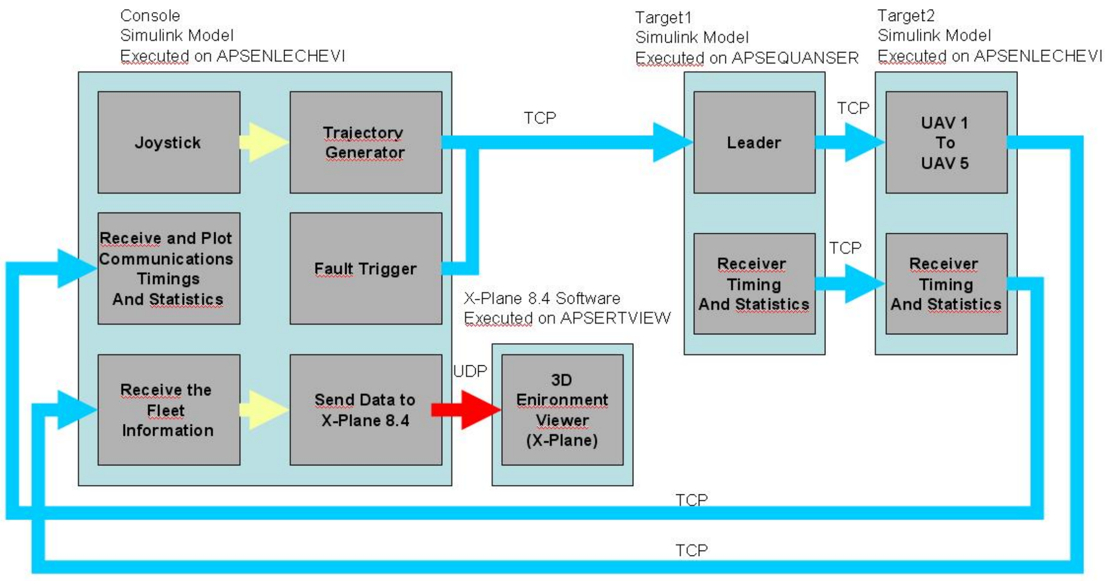


Figure 19: QuadRotor with the leader and followers on different targets

3.2.3. Plots

Figure 20 to Figure 22 show the 2D trajectory of simulated QuadRotors at different simulation instants. The blue dotted line represents the trajectory of the UAV 3 (follower 2) and the blue dashed-dotted line represents the UAV 5 trajectory. An abrupt fault has occurred on this UAV after 94 second. On the Figure 21, we can see that the UAV 5 has detected the fault and maintain his position. On the Figure 22, we see that the UAV 5 now follows the Leader. The UAV 5 detected a fault on the UAV 2 using the DAFD [10] on board the UAV 5. Note that the DAFD uses only the UAV 2 X-Y position and speed. Note also that when the abrupt fault on the UAV 2 is detected by the UAV 5, it begins to follow the leader instead of following the UAV 2. On the Figure 21 and the Figure 22, we see that the UAV 5 catching up to the leader.

Note that the trajectory of the UAV 3 has been omitted for Figure 21 and Figure 22 because of its erratic motion.

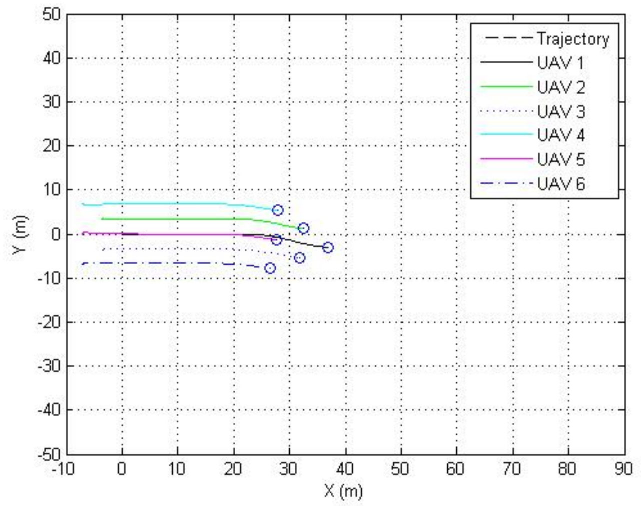


Figure 20: QuadRotor leader executed on a separate target X-Y Trajectory at 94 seconds of simulation

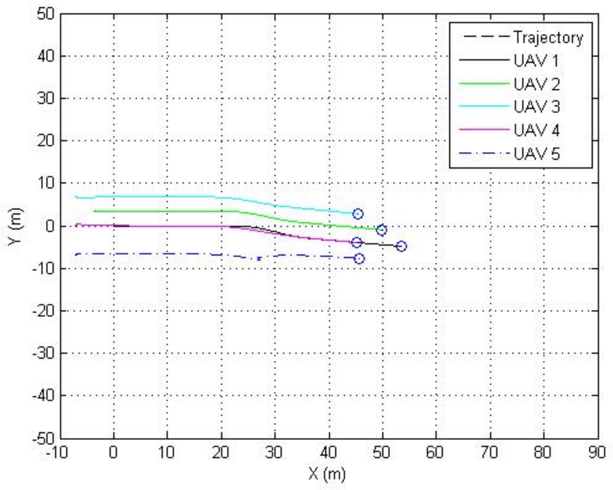


Figure 21: QuadRotor leader executed on a separate target X-Y Trajectory at 141 seconds of simulation

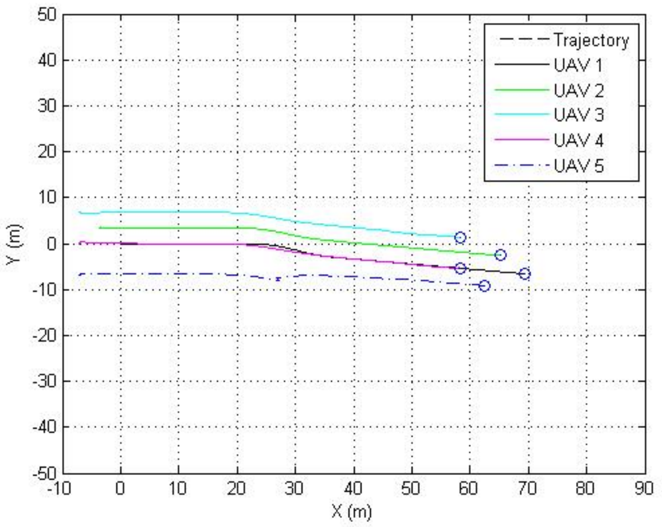


Figure 22: QuadRotor leader executed on a separate target X-Y Trajectory at 158 seconds of simulation

4. Wheeled Mobile Robot Control Using QuaRC and Windows Real-Time Targets

The Quanser QBot is a wheeled mobile robot provided by the Quanser Company [1] and shown on the Figure 23. The QBot as on board an embedded computer (GumStix [12]) that executes Linux ARM Operating System. This robot can be programmed using Matlab Simulink and Real-Time Workshop/QuaRC. Simulink models are written in C++ by Real-Time Workshop/QuaRC. The C++ source code is then sent to the QBot using a Wi-Fi link. The embedded QBot computer then compiles the source code using GCC. The QBot can the execute the compiled code in Real-Time and can communicate with other QuaRC targets using Wi-Fi. Its onboard computer can be used to process data and control actuators. The QBot can execute tasks without requiring a base station. Indeed, controllers, sensors data processing and other tasks can be executed on the on board embedded computer. The QBot can also be part of a Real-Time simulation. Indeed, simulations can rely on the QBot hardware data. For instance, a virtual vehicle can follow the QBot using its internal sensors.

4.1. Presentation of the QBot

The Figure 23 presents the QBot. The QBot is a Wheeled Mobile Robot provided by Quanser [5].

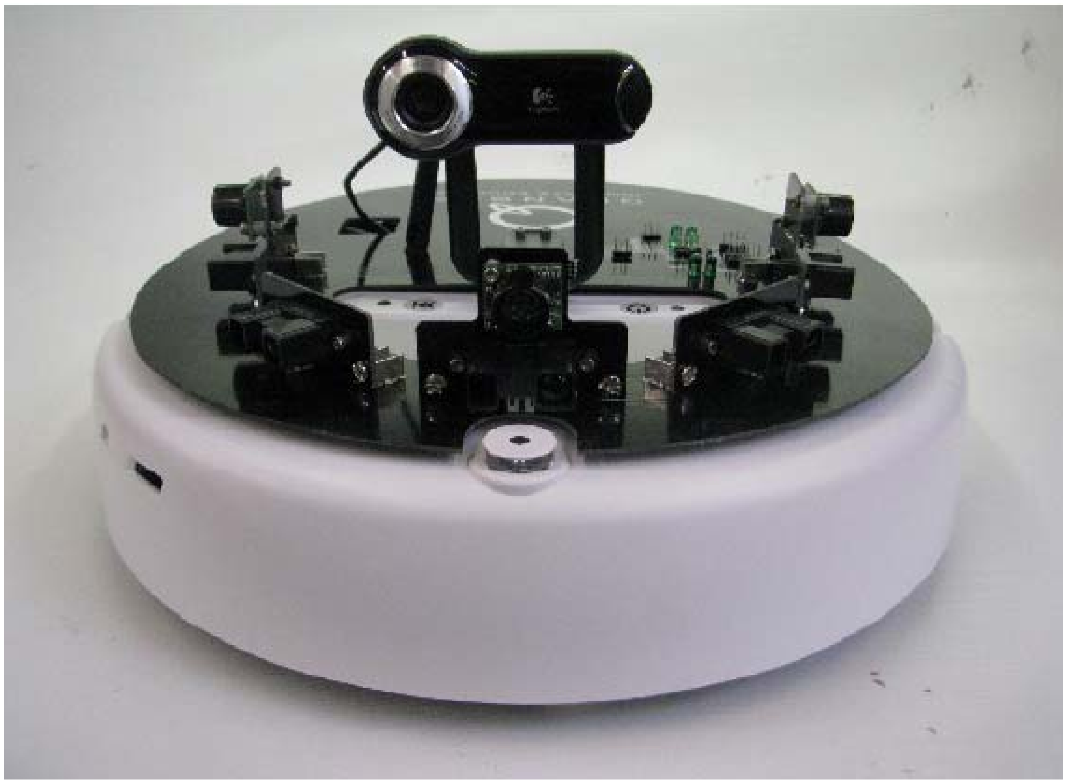


Figure 23: Presentation of the QBot

QBot's embedded computer specifications are listed below:

Computer Type	GumStix verdex pro XL6P [12]
Processor	Marvell® PXA270 with XScale™ @ 600 Mhz
Cores	1
Memory	128 MB ram, 32 MB Flash
Operating System	Linux 2.6.21
Web site	http://www.gumstix.net/Hardware/view/Hardware-Specifications/Verdex-Pro-Specifications/112.html

Table 1: QBot computer specifications

The QBot has the following hardware on board:

- 5 IR Sensors
- Battery Capacity Reading from the HIL Read Write Block
- X-Y-Z Magnetometers
- 3 Sonar
- 7 Analog Inputs
- 7 Digital I/O
- 8 PWM Outputs
- 22 5V outputs
- 23 ground
- 1 Webcam
- 1 Wi-Fi interface (802.11)
- 3 Bumper Sensors
- 3 Wheel drop sensors
- 1 Wall sensor
- 4 Cliff sensors
- 1 Omnidirectional; Ir Receiver
- Power Button
- Play Button
- Advance Button
- Battery Voltage, Battery Current, Battery Temperature, Battery Charge and Battery Capacity sensor
- Velocity and Radius sensor
- Roomba drive (2 motors for wheels)
- 1 Speaker

For more information about the QBot hardware and QBot blocks, refer to the QuaRC iRobot documentation provided with the purchase of the QBot.

4.2. Visual Feedback Camera System and Hardware Setup

QBot models have been developed and tested at the DRDC Valcartier hardware in the loop laboratory. This laboratory composed of a QuaRC computer, a X-Plane computer and an OptiTrack infrared camera tracking system. The QuaRC computer (APSEPORCOOP) is used as a base station for the QBot and for Real-Time simulation purposes. The X-Plane computer (APSERTCLUSTER) executes X-Plane 8.4 [4] and display simulations results in Real-Time in a three dimensional environment. The OptiTrack system is composed of 7 OptiTrack V100 infrared cameras [2] connected to the computer APMVSWAT01LAB via USB. The OptiTrack Tracking Tools software [2] is executed on APMVSWAT01LAB in order to process visual data. This infrared camera system is used to measure the trackable’s position and orientation in Real-Time with a refresh rate of 100 Hz. Trackable’s are objects formed by infrared reflectors. As you can see on the Figure 24, infrared reflectors (white balls mounted on wooden sticks) has been mounted on a QBot in order to be able to track it in Real-Time with the OptiTrack system. The Tracking Tools software sends the QBot’s position and orientation to a software called VRPN [3] (Virtual-Reality Peripheral Network) Streamer (Also executed on the APMVSWAT01LAB). This software will send the QBot’s position and orientation to the QuaRC computer via an Ethernet/UDP link. The QuaRC computer then relays the visual feedback to the QBot’s on board computer via a Wi-Fi link. The Figure 26 presents a block diagram of the hardware setup used to control one QBot using visual feedback. One can see the OptiTrack system as a local indoor GPS.

In order to use the OptiTrack system, refer to the Annex 4: Experimental Setup Quick Start Guide.

For complete computer specifications (Table 11, Table 12 and Table 13), refer to the Annex 1: Presentation of the Hardware.

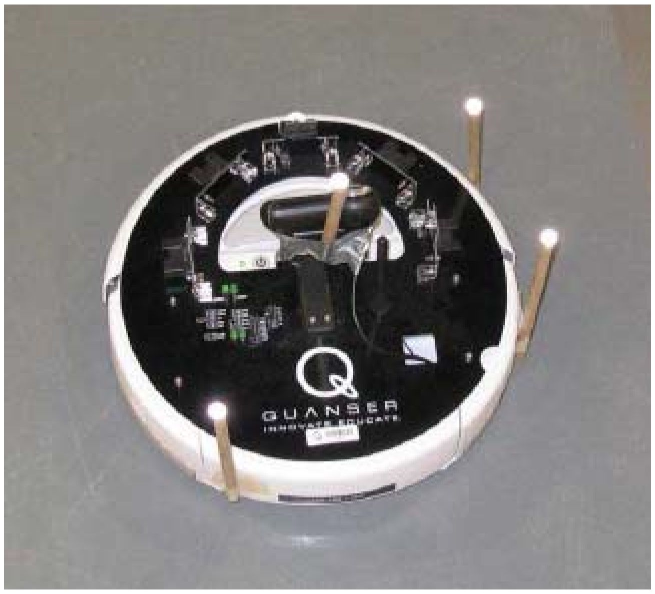


Figure 24: QBot with mounted Infrared Reflectors

4.2.1. VRPN UDP Streamer

The VRPN [3] UDP Streamer is a software that is used to transfer visual data from the Tracking Tools software [13] to the QuaRC computer using an Ethernet/UDP link. The Tracking Tools software sends the QBot position and orientation to the VRPN Streamer using the VRPN port #3883. The VRPN streamer then sends the QBot’s position and orientation to the QuaRC computer using an Ethernet/UDP link. The QuaRC computer will then relay visual feedback to the QBot using a Wi-Fi link. The QBot will then use the visual feedback to follow the commanded trajectory.

The Tracking Tools software and the VRPN Streamer are both executed on the computer called APMVSWAT01LAB.

In order to use the OptiTrack system and the VRPN Streamer, refer to the Annex 4: Experimental Setup Quick Start Guide.

Note that the VRPN Streamer executable and the C++ source code have been delivered to the DRDC-Valcartier Precision Weapons section.

4.2.1.1. QuaRC UDP Receiver

Receiving UDP packets using QuaRC communication blocks is very useful. It allows QuaRC models to communication with non QuaRC targets. Indeed, the QuaRC UDP receiver is used to receive visual data from the VRPN Streamer.

To receive an UDP packet on the QuaRC computer, QuaRC intermediate server communication blocks can be used. In the Stream answer block, udp://131.132.57.12:19000 (udp is the communication protocol, 131.132.57.12 is the IP address of the QuaRC computer and 19000 is the port on which the computer receives the message). The Figure 25 presents the CamUDP Simulink model. As you can see on the Figure 25, the stream write error is left unconnected because there is no need to send data to the VRPN UDP Streamer. You can also see on the Figure 25 the output FPS (Frame per Second). This is a measure of the mean refresh rate received from cameras. The FPS information can be used to verify that visual data is received on the QuaRC computer.

The CamUDP Simulink model has been delivered to the Precision Weapons section of DRDC Valcartier.

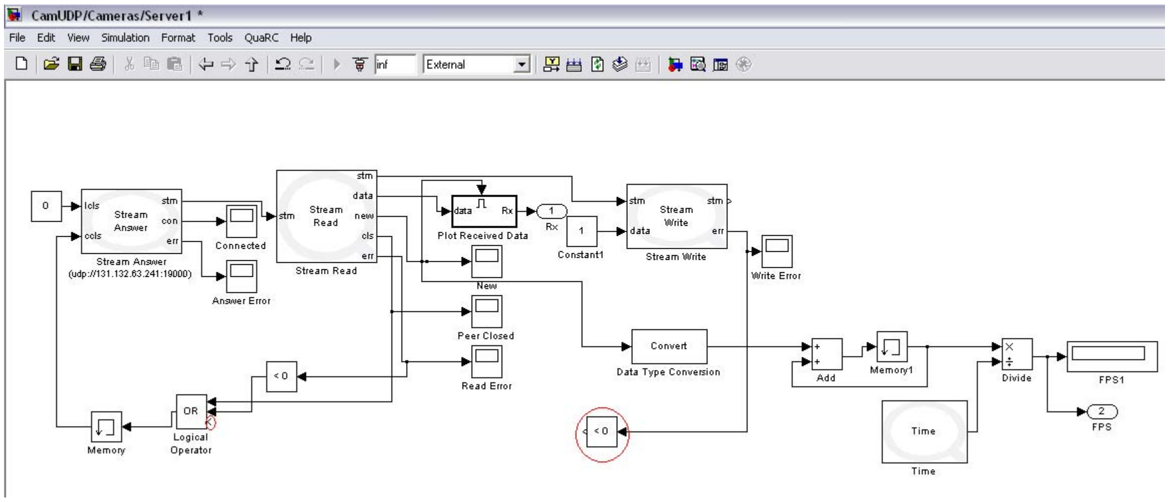


Figure 25: QuaRC UDP receiver

4.3. QuaRC QBot: Unicycle_on_QbotFollowTrajectory_Cameras_direct_drive model

This model allows the QBot to follow a specific trajectory using camera feedback (position and orientation, the QBot speed is estimated using the time derivative of the QBot's position on board the QBot's computer). The low level position controller is a feedback linearization controller [9] also executed on board the QBot's computer. The trajectory commanded to the QBot is a 2 dimensional trajectory (X – Y trajectory). Note that Simulink models are executed at a period of 0.01 s on both QBot and the QuaRC computer and that the Simulink block that controls QBot's wheels is executed at a period of 0.03 s.

Note that Simulink models developed for this experimentation has been delivered to the DRDC-Valcartier precision weapons section.

4.3.1. Objectives

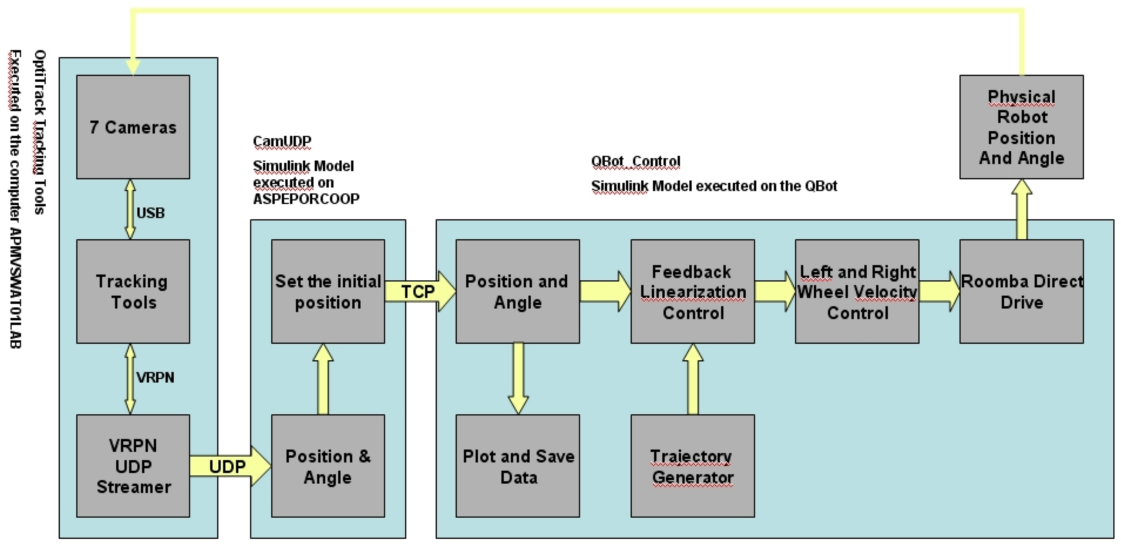
The objectives of that model are:

- Determining the performance of the feedback linearization (controller developed for the unicycle model) [9]
- Determine the position error on the QBot when following a trajectory.
- Use the OptiTrack Infrared Camera system to obtain the QBot's position in real-time and close the low-level controller's loop

4.3.2. Block Diagram

As you can see, the OptiTrack infrared camera system is used to measure the QBot's position and orientation in Real-Time. This information is then sent to the QuaRC computer (on the CamUDP model). The QuaRC computer then relays this information and the requested trajectory to the QBot using a wireless link. The QBot's embedded computer executes the feedback linearization controller [9] in order to follow the commanded trajectory.

For more information on the hardware, see the Annex 1: Presentation of the Hardware.



OptiTrack Tracking Tools
Executed on the computer APMVSMAT01LAB
7 Cameras
USB
Tracking Tools
VRPN
VRPN UDP Streamer
UDP
CamUDP
Simulink Model executed on ASPEPORCOOP
Set the initial position
Position & Angle
TCP
QBot_Control
Simulink Model executed on the QBot
Position and Angle
Feedback Linearization Control
Left and Right Wheel Velocity Control
Roomba Direct Drive
Physical Robot Position And Angle
Plot and Save Data
Trajectory Generator

Figure 26: QBot follows trajectory Overall Block diagram

4.3.3. QBot's actuators and control

The Figure 27 presents QBot actuators; left and right wheels. Note that the front wheel is not motorized. Left and Right wheels angular velocity (ω_l and ω_r) can be controlled individually. Wheels have a radius r = 0.035 m. The distance between wheels d = 0.262 m (center to center). The maximum wheel angular velocity is 14.2857 rad/s for both motorized wheel.

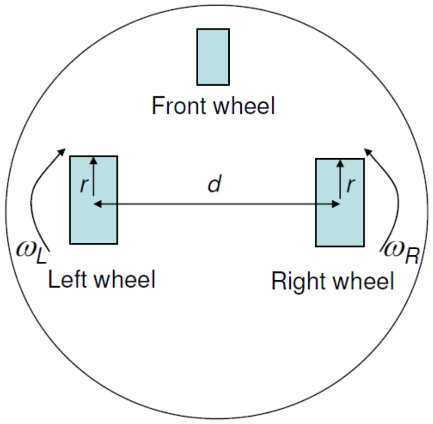


Figure 27: Presentation of the QBot's actuators

Since the feedback linearization controller [9] outputs velocity and theta commands and the QBot can be controlled by differential wheels control, a Simulink block has been designed to convert velocity and angle command to left and right wheels angular velocity commands:

$$\omega_L = \frac{V_c}{r} - \frac{d\omega_C}{2r}$$, left wheel angular velocity

$$\omega_R = \frac{V_c}{r} + \frac{d\omega_C}{2r}$$, right wheel angular velocity

Where V_c is the commanded speed output from the feedback linearization controller. The QBot commanded angular velocity ω_c is calculated from the commanded angle output of the feedback linearization controller θ_c and the actual QBot's angle θ. Ts is Simulink block that controls wheels angular velocity: Ts = 0.03 s.

$$\omega_C = \frac{\theta_C - \theta}{Ts}$$

4.3.3.1. Trajectory of the QBot and Initial Positions

The trajectory commanded to the QBot is a 2 dimensional trajectory (X – Y trajectory). The trajectory is generated as follow:

$$x(t) = 0.04t$$

$$y(t) = 0.75\sin(\frac{pi \times t}{60})$$

Where: x(t) (meter) is the position on the X axis , y(t) (meter) is the position on the Y axis and t is the simulation time (second).

4.3.3.2. *Plots*

The Figure 28 shows the commanded and the actual trajectories of the QBot on the X-Y plane. The red line represents the commanded X trajectory and the black line represents the QBot's trajectory.

The Figure 29 presents the QBot X trajectory error, the Y trajectory error and the total distance error from the commanded trajectory versus time. In the transition state, the QBot has a position error of 5 cm from the commanded trajectory. In steady state, the QBot has a position error of 2 cm. Note that the closed loop runs at a period of 0.01 s except differential wheel commands (ω_L and ω_R) that are executed at a period of 0.03 s. Note also that the QBot model and dynamic has been approximated to a simple unicycle[7] model for the feedback linearizing controller design and that performances are still satisfactory.

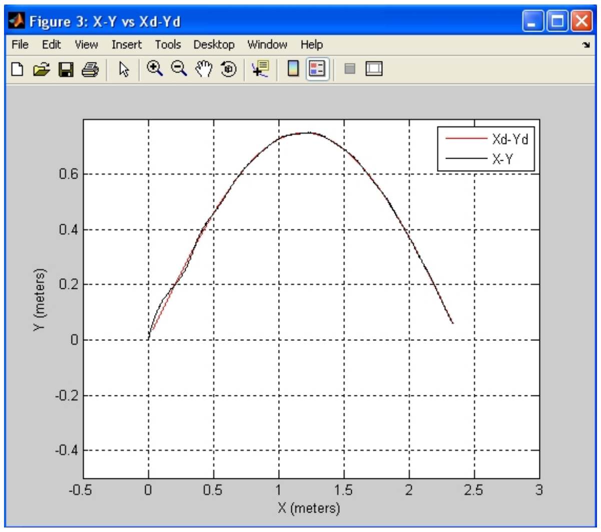


Figure 28: X and Y position versus Requested X and Y (m)

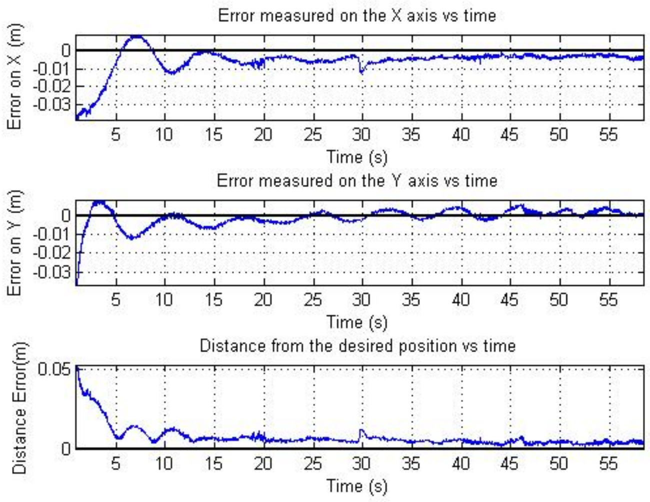


Figure 29: Distance of the QBot from the commanded trajectory (m)

5. Comparison Between QuaRC and RT-Lab simulation results

This section will present a comparison between QuaRC and RT-Lab. Both software products allow developing Real-Time distributed simulations. Simulation models are developed using Matlab/Simulink and then converted to C++ source code using Matlab/Real-Time Workshop. Source code is then compiled and executed in Real-Time. In addition, both products allow separating models in order to distribute computation tasks on more than one computer.

It is important to note that both software products can be used to include hardware to simulations such as Analog to Digital converters, Digital to Analog converters, embedded processors and more.

5.1. RT-Lab and QuaRC QuadRotor validity test models

The RT-Lab QuadRotor validity test model and the QuaRC QuadRotor validity test model have been used to compare simulation results obtained with QuaRC. These models have been delivered to the DRDC-Valcartier Precision Weapon section.

These models present six QuadRotor executing a V-Shape formation flight. The user can control the leader's velocity, heading and altitude using a joystick. Followers will attempt to follow the leader and maintain the formation's geometry. In these simulations, the leader, UAVs 2 and 5 are executed on the first target and UAVs 1,3 and 4 are executed on the second target. The user can trigger an abrupt fault on the UAV 2 at any simulation moment. The UAV 5 can detect abrupt faults that occur on the UAV 2 using the DAFD [10] and recover by following the leader instead of following the UAV 2. QuadRotor's low level controller is the feedback linearization controller [9] and the formation geometry is preserved by means of the formation controller detailed in [11]. Simulation results are displayed on X-Plane 8.4 [4] which is executed on the computer called APSERTVIEW.

Note that these simulations are based on the QuadRotor model [8].

5.1.1. Objectives

The objectives of that model were:

- Explore and understand how to distribute models using QuaRC.
- Compare simulation results obtained with RT-Lab, QuaRC and a regular simulation
- Display the simulation results in real-time on X-Plane

5.1.2. Block Diagram

Figure 30 presents the block diagram of the distributed QuaRC QuadRotor simulation. Note that the topology of the formation is a V-Shape formation as shown on Figure 32. The simulation uses 3 Real-Time targets. The model named Console has the task to generate a trajectory from the joystick command, allow the user to trigger a fault on the UAV 2 and send simulation results in Real-Time to X-Plane[4]. The leader, UAVs 2 and 5 are simulated on the model Target 1. UAVs 1,3 and 4 are simulated on the model Target 2. The computer named APSEQUANSER is executing the model Target 1 and the computer named APSENLECHEVI is executing the console and the Target 2 model. The console model sends simulation results in Real-Time to the computer APSERTVIEW using an Ethernet/UDP link. This computer displays simulation results using the three dimensional viewer X-Plane 8.4 [4].

For more information on the hardware, see the Annex 1: Presentation of the Hardware. (See Table 8, Table 9 and Table 10).

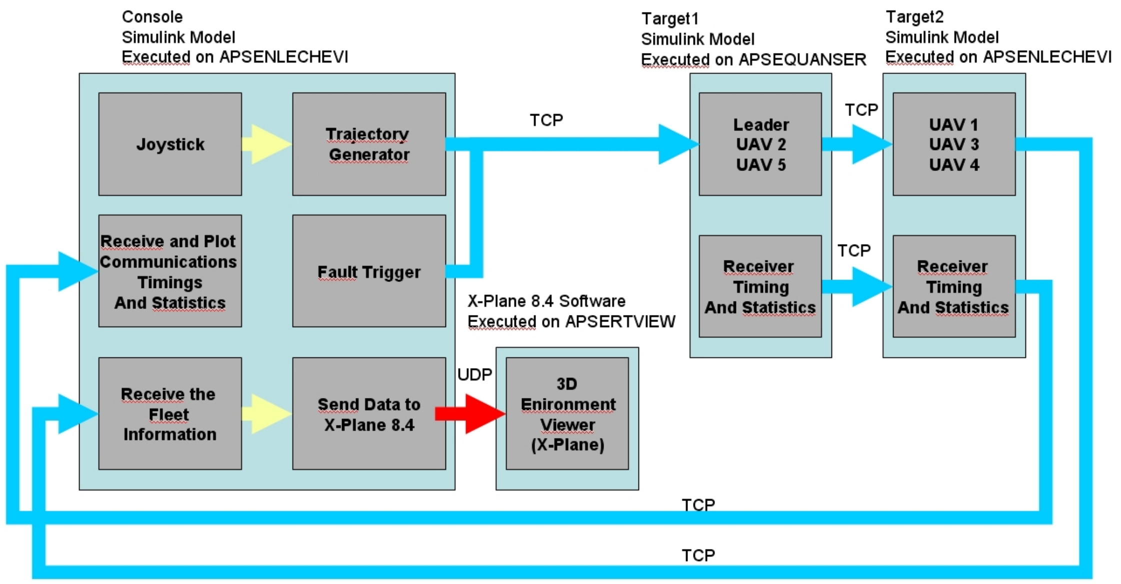


Figure 30: QuaRC Distributed QuadRotor simulation

The Figure 31 presents the block diagram of the distributed RT-Lab QuadRotor simulation. Note that the topology of the formation is a V-Shape formation as shown on the Figure 32. The simulation uses 2 Real-Time targets. The model named sc_console (Executed on APSERTVIEW) has the task to generate a trajectory from the joystick command, allow the user to trigger a fault on the UAV 2. The leader, UAVs 2 and 5 are simulated on the Target 1. UAVs 1,3 and 4 are simulated on Target 2. The computer named APSERTVIEW is used to display simulation results in Real-Time on X-Plane 8.4 [4].

For more information on the hardware, see the Annex 1: Presentation of the Hardware. (See Table 4, Table 5, Table 6 and Table 7).

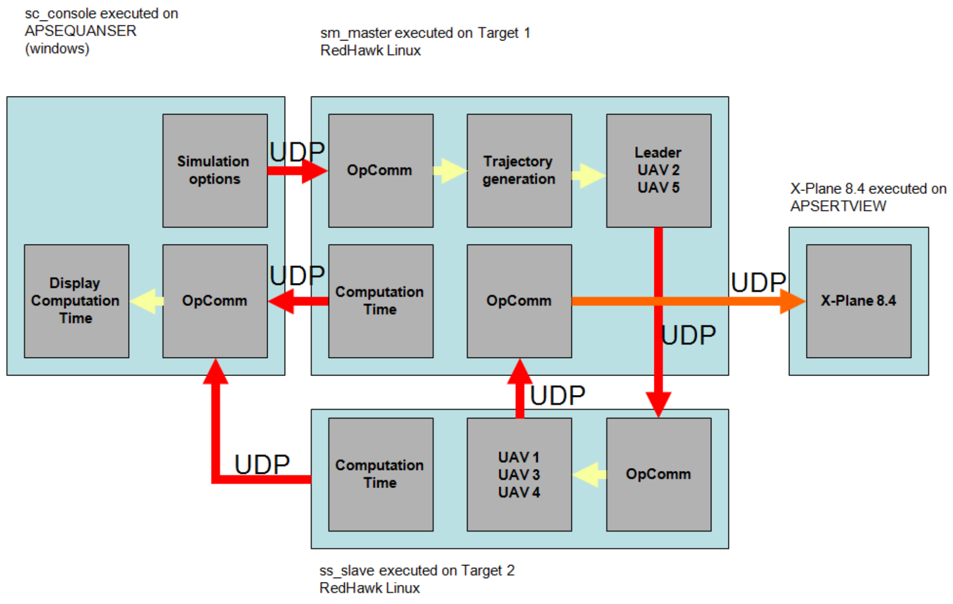


Figure 31: RT-Lab Distributed QuadRotor simulation

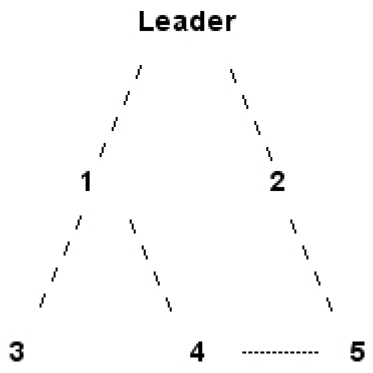


Figure 32: QuadRotor Formation topology (lines indicate information flow)

5.1.3. Results

Simulations have been done using the original Simulink QuadRotor model [8], the QuaRC distributed QuadRotor model and the RT-Lab distributed model. Note that on both QuaRC and RT-Lab models, UAVs 0, 2 and 5 are executed on the first target and UAVs 1,3 and 5 are executed on a second target. On the Figure 33 the X-Y trajectory of the UAV 4 is plotted for the regular Simulink Simulation (blue dashed line), the QuaRC distributed simulation (green dashed line) and the RT-Lab distributed simulation (red line).

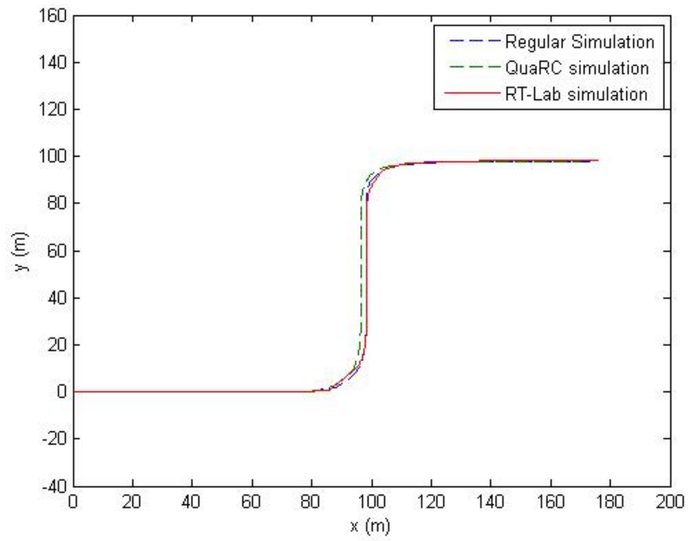


Figure 33: Simulation results for the UAV 4 in regular simulation, RT-Lab simulation and QuaRC Simulation

On the Figure 34 the X-Y trajectory of the UAV 5 is plotted for the regular Simulink Simulation (blue dashed line), the QuaRC distributed simulation (green dashed line) and the RT-Lab distributed simulation (red line). Note that the trajectory of the UAV 5 has an abrupt change around X = 100 m and Y = 40 m. It is due to the fact that the UAV 2 had a fault at this moment and that the UAV 5 has detected it and then starts to follow the leader instead of following the UAV 2.

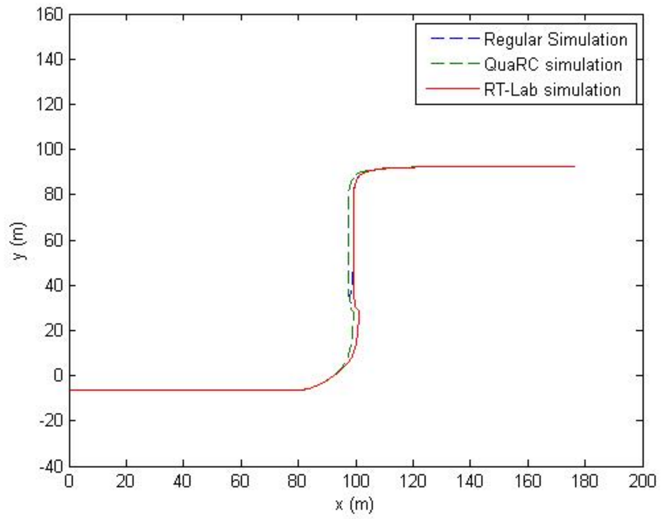


Figure 34: Simulation results for the UAV 5 in regular simulation, RT-Lab simulation and QuaRC Simulation

As you can see, on QuaRC, plots have been translated to the left compared to curves from the regular model and curves from the RT-Lab simulation. It can be explained by the way models are started using QuaRC. In QuaRC, the simulation is split in 3 different models. To begin the simulation, models must be started sequentially causing a delay between the moment models are started and the moment models are enabled (See the Annex 7:Synchronizing QuaRC Models). This delay affects the way the trajectory is generated and thus slightly affecting simulation results.

6. Major Differences Between QuaRC and RT-Lab

This section presents a comparison chart between QuaRC and RT-Lab. Both software products allow developing a simulation model that can be executed in real-time. Model are developed using Matlab/Simulink, they are then converted to C++ and compiled using the Real-Time-Workshop compiler. Both products allow the user to distribute the computation task over multiple computers and cores.

6.1. Major differences between QuaRC and RT-Lab

	QuaRC	RT-Lab
Compiler	QuaRC requires changing the compiler in RTW for each model that was not created on a computer that has QuaRC installed. You also have to change the model configuration parameters to support the external mode.	With RT-Lab, this is completely transparent to the user.
Scripts	With QuaRC, you have to execute a .m parameter files manually before starting the model in order to initialize constants.	With RT-Lab, you can associate a .m parameter file to a model, when you open the model, the .m script is executed automatically.
Malfunctions	When you do not do the operations in the right order or assign 2 models to a same URI, Simulink will freeze.	You must always reset the target before loading a new model, sometimes, the target has a model and you can't get rid of it, you have to reset

		the target (Reset RedHawk Computers).
Initial state	In QuaRC, you don’t need to reload the model in order to restart at the initial state, simply reconnect to the targets and click play.	In RT-Lab, you have to reload the model to restart it to the initial state.
On-the-fly changes	In QuaRC, you can change a constant on the fly when executing a model.	In RT-Lab, you can modify a constant on the fly in the console (Non-Real Time Model) only.
Model separation and IP addresses	In QuaRC, you must separate the targets manually. In addition, the user must configure communications blocks (IP addresses and port numbers) in order to exchange data between targets.	In RT-Lab, target separation is done automatically and IP addresses are handled by the software. You have to specify to RT-Lab how to separate targets.

Table 2: Major differences between QuaRC and RT-Lab

	QuaRC	RT-Lab
Communications	When you use multiple targets on QuaRC, you have to configure communication blocks manually (client/server protocol and port number) for each link between targets. In addition with QuaRC, handling communications takes much time, but they can customize for certain needs.	In RT-Lab, opComm block handles communications between targets. RT-Lab communication blocks are easy to use and user friendly.
External Communication	QuaRC Targets can communicate with Non-QuaRC software or hardware by using the TCP, UDP, SHMEM and serial protocol which is very useful for developing an HIL system.	With RT-Lab, the communication using the Ethernet port to non RT-Lab Targets must be programmed using S-Functions. This constraint does not allow fast prototyping when including non RT-Lab targets.
Model	To have	In RT-Lab,

Performance and Monitoring	access to the communication statistics, models have to be modified in order to respect the QuaRC computation time block requirements (refer to the Matlab Help about the Computation time block or the QuaRC Computation Time demo)	network statistics can be found in the opSimulationInfo block. In the block, you have access to the time needed to send and receive packets.
Model Performance and Monitoring	QuaRC will most of the time require to modify the existing model to be able to obtain the computation time. This process may require a lot of time.	In RT-Lab, the OpSimulationInfo gives the models calculation time, communication statistics and effective sample time. This is transparent to the user. It is as simple as adding a block in the model.
Simulation Architecture	In QuaRC, you can run a model which has no console.	In RT-Lab, all models must have a console.
User Interface and Compilation	In QuaRC, you can execute	In RT-Lab, you have to go

	your model directly in Matlab. The model is built in the Matlab's console.	through the RT-Lab interface to build and execute the code.
Data analysis	In QuaRC real-time results are accessible on any target.	In RT-Lab real-time results are accessible only in the console (Non-Real Time target).

Table 3: Major differences between QuaRC and RT-Lab

6.2. QuaRC Strong Points

- QuaRC has a very complete communication block set that allows communicating with non QuaRC targets. For example, QuaRC communication blocks are used to receive data from the OptiTrack computer on an Ethernet/UDP link.
- Integrating hardware with QuaRC is made easy with the QuaRC Communications Library which is available in the Simulink library browser.
- Quanser also provides hardware boards or robots (ground, air) that are QuaRC compatible. Hardware can be integrated to a QuaRC HIL system very quickly.
- Targets can be windows targets, linux_arm targets, linux_x86, and qnx_x86. QNX
- Constants can be changed during the real-time execution on any target

6.3. QuaRC Weak Points

- Using Model and Target URI can be confusing
- Handling communications can be confusing
- A model that is not properly configured can crash Matlab when executed
- Computation timings may be hard to obtain when using windows targets

6.4. RT-Lab Strong Points

- Handling communications is user friendly
- Handling targets is user friendly
- Building and loading is user friendly
- Obtaining Computation timings is very easy
- RT-Lab provides hardware boards that allow to design a HIL system

6.5. RT-Lab Weak Points

- Communication blocks do not allow exchanging data with non RT-Lab targets using the Ethernet port, an S-Function must be programmed in order to do so.
- Constants can’t be changed on real-time targets, but it is indeed possible to initialize constants in the RT-Lab console and change them without needing to recompile
- Real-Time targets can’t be widows target

7. Conclusion

This report presented real-time formation flight simulation models. Formations presented are composed of ALTAVs, unicycles and quadrotors. Formation geometries used for these models are V-Shape and String-Shape types. Simulation models include vehicles dynamics, low-level controller and high-level formation controller. It was demonstrated through simulations that homogenous formations remained stable. It was also demonstrated that both formation controller (leader-follower type) and low-level controller (feedback linearization controller) can be ported to several aerial platforms and executed in real-time, and thus are realizable. Simulation models were developed using Matlab/Simulink then ported to QuaRC and RT-Lab compliant models. Both QuaRC and RT-Lab environments allow compiling, distributing and executing Simulink models in real-time.

The feedback linearization controller was implemented onboard a wheeled mobile robot (Quanser QBot) embedded processor. It was shown that this low-level controller can be executed on a computer with limited computational capabilities and memory, and yet, it was demonstrated that the QBot can follow an X-Y trajectory with satisfactory performances. The robot's position is measured using an infrared camera setup and the position information is sent to the robot's embedded computer where the low-level controller is executed. Note that the visual data is sent from the base station to the QBot using a wireless link. This experiment showed that the controller was robust to measurement noise on the position, on the heading and on the speed estimation.

QuaRC and RT-Lab were compared. Both software solutions enable to develop models using Matlab/Simulink, compile, distribute and execute models in Real-Time. Simulink models are automatically converted to C++ code, which is then compiled for specified Targets. In general, RT-Lab is easier to use than QuaRC, although it provides less flexibility than QuaRC. Indeed, QuaRC provides several options such as the possibility to choose and configure communication options.

Finally, this report presented quick start guides for the RT-Lab and QuaRC setups that are used to execute distributed real-time simulations and the QuaRC hardware-In-the-loop setup that is used to perform experiments using wheeled mobiles robots, QuaRC and the infrared tracking camera system.

Future work will present mixed simulation models and wheeled mobile robot experiments. In other words, the development of an experiment including simulated vehicles and real wheeled mobile robots is planned. In addition, the development of a formation composed entirely of wheeled mobile robots is also planned. Furthermore, there will be experiments on decentralized fault, failure and anomaly detection. Ultimately, the formation will use onboard sensors to navigate and thus removing the dependency on the infrared camera system.

The formation control can have multiple civil and military applications such as automated aerial refueling, coordinated bombing, territorial surveillance, multi-vehicle heavy lift. Formation control can also provide unmanned support to manned vehicles for surveillance or combat missions.

References

[1] QuaRC QBot User's Guide, Quanser Inc., 119 Spy Court, Markham Ontario, Canada.

[2] NaturalPoint OptiTrack optical motion capture solutions, NaturalPoint Inc., 2317 Corvallis, OR, USA.

[3] Russell M. Taylor II, Thomas C. Hudson, Adam Seeger, Hans Weber, Jeffrey Juliano, Aron T. Helser, "VRPN: A Device-Independent, Network-Transparent VR Peripheral System," *Proceedings of the ACM Symposium on Virtual Reality Software & Technology 2001*, VRST 2001. Banff Centre, Canada, November 15-17, 2001.

[4] Laminar Research, X-Planes 3D viewer environment, web site: http://www.x-plane.com/

[5] Quanser, Embedded Matlab/Simulink Solutions and Mechatronics, web site : http://www.quanser.com/

[6] E. Earon, "Almost-lighter-than-air vehicle fleet simulation", *Technical Report V. 1.1*, Quanser Inc., 2006.

[7] N. Léchevin, C.A. Rabbath, and P. Sicard, "Stable Morphing of Unicycle Formations in Translational Motion", in *Proceedings of the 2006 American Control Conference*, Minneapolis, MN, pp. 4231-4236.

[8] P. Castillo, R. Lozano, and A.E. Dzul, *Modelling and Control of Mini-Flying Machines*, Springer, 2005.

[9] N. Léchevin and C.A. Rabbath, "Sampled-data Control of a Class of Nonlinear Flat Systems with Application to Unicycle Trajectory Tracking", *Journal of Dynamic Systems, Measurement, and Control*, September 2006 , Volume 128, Issue 3, pp. 722-728.

[10] N. Léchevin and C.A. Rabbath, "Robust Decentralized Fault Detection in Leader-to-Follower Formations of Uncertain, Linearly Parameterized Systems", *AIAA Journal of Guidance, Control and Dynamics*, Vol. 30, No.5, September-October 2007, 1528-1535.

[11] N. Léchevin, C.A. Rabbath and P. Sicard, "Trajectory Tracking of Leader-Follower Formations Characterized by Constant Line-of-Sight Angles", *Automatica*, Vol. 42, No. 12, December 2006, 2131-2141.

[12] GumStix, web:http://www.gumstix.com/

[13] OptiTrack Natural Point, web: http://www.naturalpoint.com/optitrack/

[14] National Instruments, web: http://www.ni.com/

Appendix

REF 1: Quanser QBot User Manual
The QBot User Manual is provided with the purchase of the QBot

REF 2: OptiTrack Website
http://www.naturalpoint.com/optitrack/ [2]

REF 3: Natural Point Rigid Body Toolkit
http://www.naturalpoint.com/optitrack/support/downloads-archive.html [2]

REF 4: Natural Point, Point Cloud Calibration Tools
http://www.naturalpoint.com/optitrack/support/downloads-archive.html [2]

REF 5: Natural Point Tracking Tools
http://www.naturalpoint.com/optitrack/support/downloads.html#software [2]

REF 6: Natural Point Tutorial Videos
http://www.naturalpoint.com/optitrack/products/videos.html#tt [2]

REF 7: VRPN UDP Streamer
The VRPN UDP Streamer source code and visual studio project has been delivered to DRDC-Valcartier Precision Weapons Section.

REF 8: QuaRC QBot: Unicycle_on_QbotFollowTrajectory_Cameras_direct_drive
This model has been delivered to DRDC-Valcartier Precision Weapons Section.

REF 9: QuaRC Windows: QuadRotor_Fault_Detect_Joystick_3onTarget1_3onTarget2
This model has been delivered to DRDC-Valcartier Precision Weapons Section.

REF 10: RT-Lab:AFG_summer2007joystick
This model has been delivered to DRDC-Valcartier Precision Weapons Section.

REF 11: RT-Lab:AFG_summer2007joystick2CPUs
This model has been delivered to DRDC-Valcartier Precision Weapons Section.

REF 12: RT-Lab:AFG_summer2007joystickHWsynchro
This model has been delivered to DRDC-Valcartier Precision Weapons Section.

REF 13: RT-Lab:ALTAVconvoyTeamFDRSummer2007(DAFD)HWSynchro
This model has been delivered to DRDC-Valcartier Precision Weapons Section.

REF 14: RT-Lab:ALTAVconvoyTeamFDRSummer2007joystick(DAFD)
This model has been delivered to DRDC-Valcartier Precision Weapons Section.

REF 15: RT-Lab: QuadRotor_validity_test
This model has been delivered to DRDC-Valcartier Precision Weapons Section.

Annex 1: Presentation of the Hardware

This annex presents computers that compose the RT-Lab distributed simulation setup, the QuaRC distributed simulation setup and the QuaRC Hardware In the Loop setup.

1. RT-Lab Setup

The Figure 35 presents the RT-Lab setup block diagram. This setup is used to distribute Real-Time simulations using RT-Lab. This setup is composed of one non Real-Time Target (APSEQUANSER executing Windows XP OS) and two Real-Time Targets (Target 1 and Target 2 executing RedHawk Linux OS). The non Real-Time Target allows the user to visualize simulation results and modify simulation parameters during the model's execution. RedHawk targets execute simulation models in Real-Time. Note that RedHawk targets can be synchronized via software (using Ethernet/UDP or FireWire) or via an hardware synchronization board. A fourth computer (APSERTVIEW) executes X-Plane 8.4. This software displays simulations results in Real-Time in a 3 dimensional environment in Real-Time.

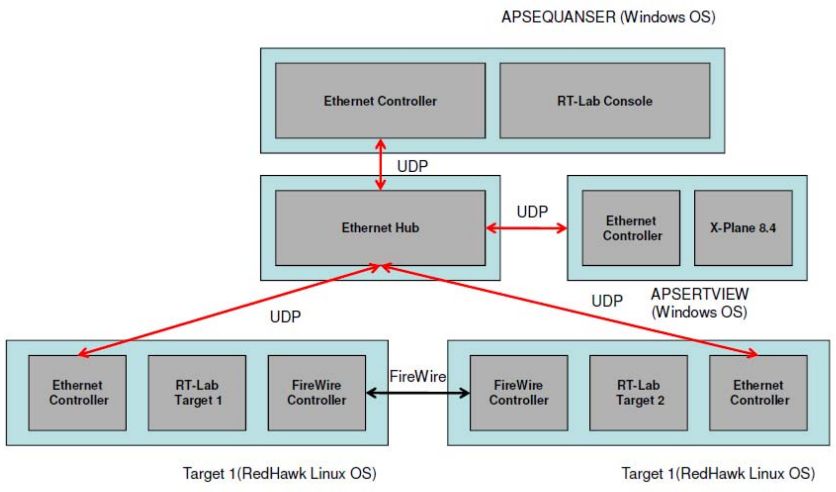


Figure 35: RT-Lab Hardware Setup

1.1. RT-Lab Windows Target (Non Real Time Target)

Computer Name	APSEQUANSER
Computer Type	DELL OPTIPLEX GX280
Processor	Pentium® 4 3.00GHz
Cores	1
Memory	1.5 GB Ram
Video Card	Intel® 82915G/GV/910GL Express
Operating System	Microsoft Windows XP Professional SP3
Matlab Version	R2007b
QuaRC Version	1.2
RT-Lab version	8.2 beta 4

Table 4: APSEQUANSER computer specifications

1.2. RT-Lab RedHawk Linux Target (Real-Time Target)

Computer Name	Target1
Computer Type	Dell Precision 530
Processor	Intel P4/Xeon Extended MCE 2.2 GHz
Cores	1
Memory	1 GB Ram
Operating System	RedHawk Linux 2.3
Hardware Synchronization card	National Instrument NI-6602

Table 5: Target 1 computer specifications

Computer Name	Target 2
Computer Type	Dell Precision 530
Processor	Intel P4/Xeon Extended MCE 2.2 GHz
Cores	1
Memory	1 GB Ram
Operating System	RedHawk Linux 2.3
Hardware Synchronization card	No

Table 6: Target 2 computer specifications

1.3. X-Plane computer (Non RT-Lab Target)

Computer Name	APSERTVIEW
Computer Type	DELL XPS 600
Processor	Intel ® Pentium® D CPU 3.46 GHz
Cores	4
Memory	2 GB Ram
Video Card	nVIDIA GeForce 7900 GTX
Operating System	Microsoft Windows XP Professional SP2
Matlab Version	R2007b
QuaRC Version	1.2
X-Plane version	8.4

Table 7: APSERTVIEW computer specifications

2. QuaRC Windows Setup

The Figure 36 presents the QuaRC distributed hardware setup. As you can see, three Windows XP computer are connected together using an Ethernet connection. Each computer can be used as a target and/or to build models. The computer APSERTVIEW executes X-Plane 8.4. X-Plane is a three dimensional viewer that displays simulation results in Real-Time. Any QuaRC target can send data in Real-Time to X-Plane using an Ethernet/UDP link. QuaRC models can exchange data using an Ethernet/TCP link.

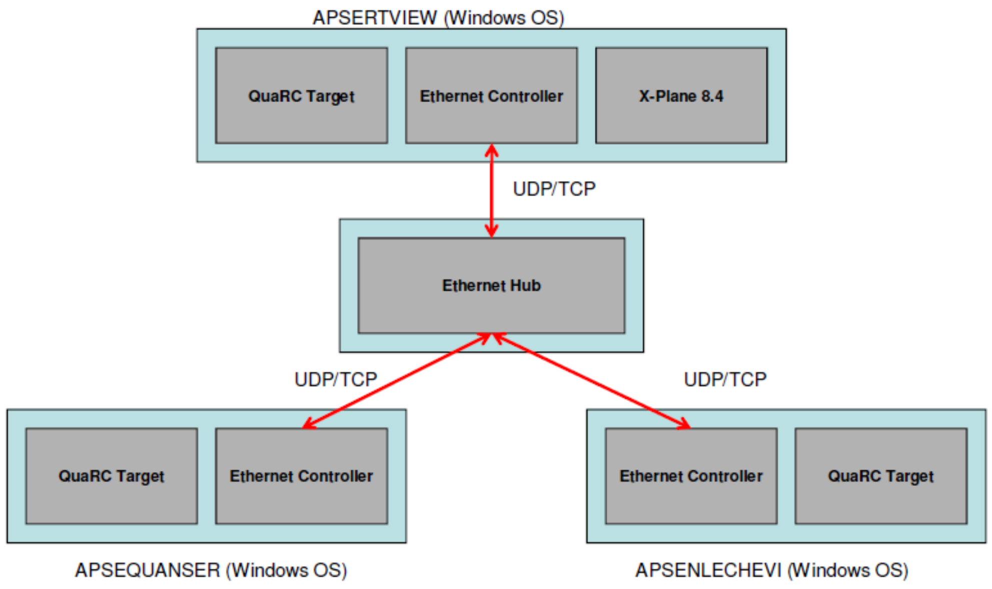


Figure 36: QuaRC Distributed Simulation Hardware Setup

Following tables presents QuaRC computers specifications.

Computer Name	APSERTVIEW
Computer Type	DELL XPS 600
Processor	Intel ® Pentium® D CPU 3.46 GHz
Cores	4
Memory	2 GB Ram
Video Card	nVIDIA GeForce 7900 GTX
Operating System	Microsoft Windows XP Professional SP2
Matlab Version	R2007b
QuaRC Version	1.2
X-Plane version	8.4

Table 8: APSERTVIEW computer specifications

Computer Name	APSENLECHEVI
Computer Type	DELL Precision 360
Processor	Pentium® 4 3.20GHz
Cores	1
Memory	1 GB Ram
Video Card	nVIDIA Quadro FX 500/600
Operating System	Microsoft Windows XP Professional SP2
Matlab Version	R2007b
QuaRC Version	1.2

Table 9: APSENLECHEVI computer specifications

Computer Name	APSEQUANSER
Computer Type	DELL OPTIPLEX GX280
Processor	Pentium® 4 3.00GHz
Cores	1
Memory	1.5 GB Ram
Video Card	Intel® 82915G/GV/910GL Express
Operating System	Microsoft Windows XP Professional SP3
Matlab Version	R2007b
QuaRC Version	1.2

Table 10: APSEQUANSER computer specifications

3. QuaRC HIL setup

The Figure 37 presents the block diagram of the QuaRC HIL setup. This setup is used to execute Hardware In the Loop simulations and Wheeled Mobile Robot experiments. This setup is composed of one QuaRC computer (APSEPORCOOP), one X-Plane computer [4] (APSERTCLUSTER), one OptiTrack computer (APMVSWAT01LAB) and Wheeled Mobile Robots (QBots). Note that the QBot possesses an embedded computer on board. Each computer can exchange data using an Ethernet link. Indeed, The OptiTrack computer sends visual feedback data to the QuaRC computer using Ethernet/UDP. The QuaRC computer sends simulation results to the X-Plane computer using Ethernet/UDP. Note also that the QuaRC computer and the QBot can exchange data using a wireless link (Wi-Fi).

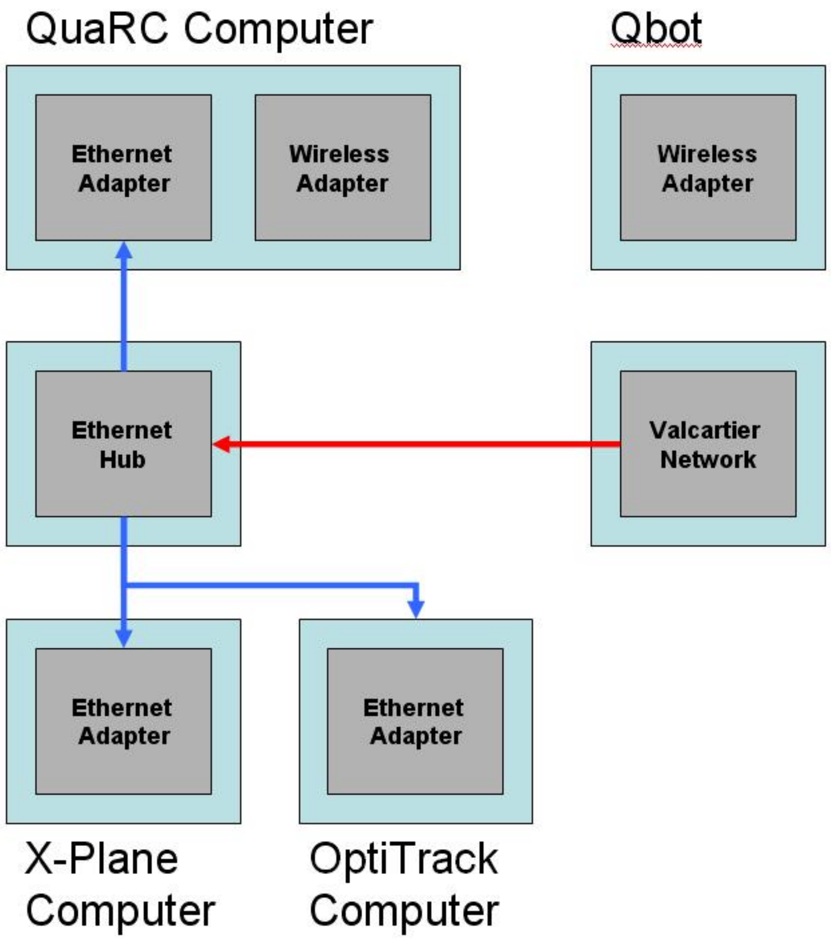


Figure 37: QuaRC HIL hardware specifications

3.1. QuaRC Computer

This computer is used to develop, compile and load QuaRC models. This computer can also be used as a QuaRC Real-Time Target or processing base station for Wheeled Mobile Robots. In fact, visual feedback is relayed by this computer from the Camera feedback computer to QBots.

Computer Name	APSEPORCOOP
Computer Type	DELL Inspiron 9300
Processor	Intel® Pentium M 2.00 GHz
Cores	2
Memory	2 GB Ram
Video Card	nVIDIA GeForce Go 6800
Operating System	Microsoft Windows XP Professional, Service Pack 3
Matlab Version	R2007b
QuaRC Version	1.2

Table 11: APSEPORCOOP computer specifications

3.2. Natural Point OptiTrack computer

This computer is used to process in real-time the infrared cameras data and extract the position and bearing of rigid bodies. This computer is connected to seven infrared cameras (OptiTrack V100). The Tracking Tools software 2.0 beta [2] is used to process visual data.

Computer Name	APMVSWAT01LAB
Computer Type	DELL XPS 720
Processor	Intel core™ 2 extreme Q6800 @ 2.93 GHz
Cores	4
Memory	3 GB Ram
Video Card	nVIDIA GeForce 7300 GS
Operating System	Microsoft Windows XP media center, Service Pack 2
Tracking tools version	2.00 Beta
Infrared Cameras	OptiTrack V100

Table 12: APMVSWAT01LAB computer specifications

3.3. X-Plane Computer

This computer is used to show the simulation results in a 3d environment called X-Plane [4]. X-Plane is used to display up to ten aerial vehicles in a 3D environment. Vehicles position and orientation can be updated in Real-Time during simulations or Wheeled Mobile Robot experiments.

Computer Name	APSERTCLUSTER
Computer Type	Dell Precision T5400
Processor	Intel® Xeon® E5405 @ 2.00 GHz
Cores	1
Memory	3.25 GB Ram
Video Card	nVIDIA Quadro NVS 290
Operating System	Microsoft Windows XP Professional, Service Pack 2
X-Plane version	8.4

Table 13: APSERTCLUSTER computer specifications

3.4. QBot computer

The QBot is a Wheeled Mobile Robot that can be programmed using Real-Time Workshop/QuaRC. This robot has an embedded computer that allows executing algorithms on board. This robot also possesses a Wi-Fi interface that allows communicating with the QuaRC base station or other QBots.

The QBot has following sensors on board:

- 5 IR Sensors
- Battery Capacity Reading from the HIl Read Write Block
- X-Y-Z Magnetometers
- 3 Sonars
- 7 Analog Inputs
- 7 Digital I/O
- 8 PWM Outputs
- 22 5V outputs
- 23 ground
- 1 Webcam
- 1 Wi-Fi interface (802.11)
- 3 Bumper Sensors
- 3 Wheel drop sensors
- 1 Wall sensor
- 4 Cliff sensors
- 1 Omnidirectional; Ir Receiver
- Power Button
- Play Button
- Advance Button
- Battery Voltage, Battery Current, Battery Temperature, Battery Charge and Battery Capacity sensor
- Velocity and Radius sensor
- Roomba drive (2 motors for wheels)
- 1 Speaker

Table 14 presents the hardware specification of the QBot on board computer:

Computer Type	GumStix verdex pro XL6P [12]

Processor	Marvell® PXA270 with XScale™ @ 600 Mhz
Cores	1
Memory	128 MB ram, 32 MB Flash
Operatin g System	Linux 2.6.21
Web site	http://www.gumstix.net/Hardware/view/Hardware-Specifications/Verdex-Pro-Specifications/112.html

Table 14: QBot computer specifications

Annex 2: RT-LAB Quick Start Guide

1. RT-Lab Quick Start Guide

RT-Lab is a software that allows to developing simulation models that can be executed in real-time. Models are developed using Matlab/Simulink. They can then be converted to C++ and compiled using the RT-Lab Real-Time-Workshop compiler. RT-Lab also offers to split models into smaller models in order to distribute the computation task to more than one computer or more than one computer core. Each computation node (computer or core) will be called a Target. For example, if a computer has 4 cores, it is possible to execute 4 different models (one on each core). This computer can then support 4 Targets.

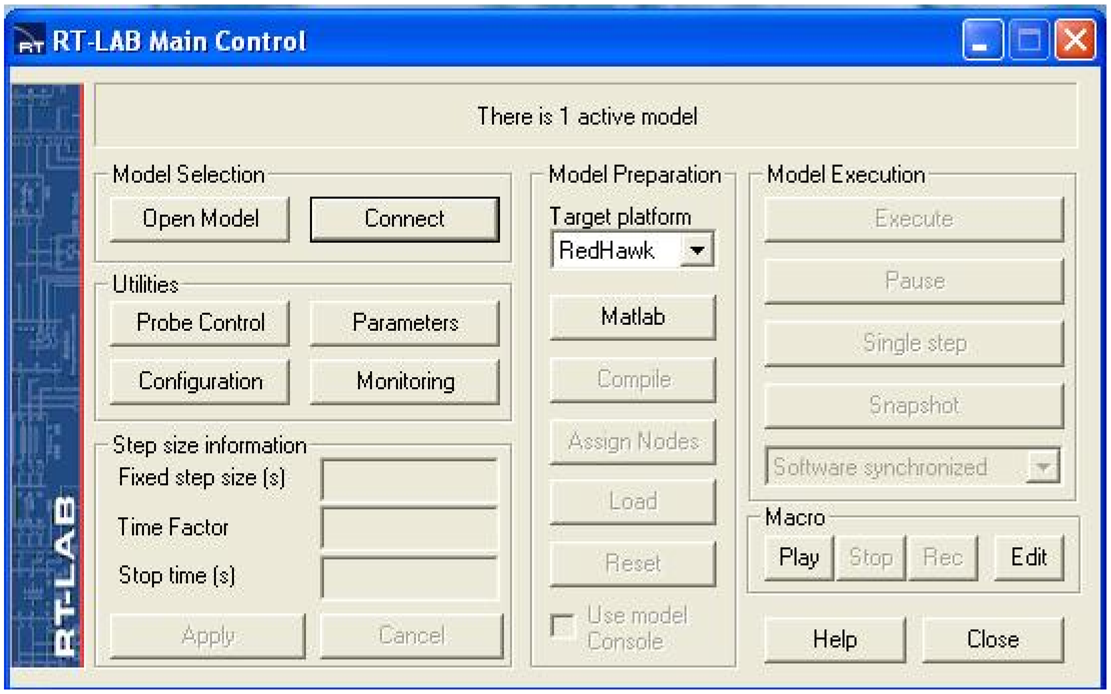
1.1. Available example

Following models are available the DRDC-Valcartier Precision Weapon Section. These models can be used as example to begin using RT-Lab.

- RT-Lab:AFG_summer2007joystick
- RT-Lab:AFG_summer2007joystick2CPUs
- RT-Lab:AFG_summer2007joystickHWsynchro
- RT-Lab:ALTAVconvoyTeamFDRSummer2007(DAFD)HWSynchro
- RT-Lab:ALTAVconvoyTeamFDRSummer2007joystick(DAFD)
- RT-Lab QuadRotor model

1.2. RT-Lab Main Control

The Figure 38 shows the RT-Lab main control panel. This panel allows configuring simulation and models parameters. In order to compile and execute a model, RT-Lab must connect to the Simulink model; click on the Connect button and select the desired model. Once the model is connected, click on compile. A new window will then open and display compilation results. Once models are compiled, click on load to send executables to Targets. Once models are loaded, click on execute to start models.


RT-LAB Main Control

There is 1 active model

Target platform: RedHawk

Software synchronized

Figure 38: RT-Lab Main Control Panel

The Figure 39 presents RT-Lab an example of model distribution scheme. Four Targets are presented: SC_Console, SM_Master, SS_Slave1 and SS_Slave2. The model called SC_Console is a Non Real-Time Target executed on windows. This Target is used to give the user an interface with Real-Time Targets. Indeed, users can change simulation parameters in the console. These changes will then be sent to Real-Time Targets (Each Target besides the SC_Console). SM_Master is the target that handles software synchronization or hardware synchronization. This Target can also be used as a calculation node. This Target is mandatory in a Real-Time RT-Lab simulation. SS_Slave1 and SS_Slave2 are used as calculation nodes and are optional in RT-Lab simulations. However, these Targets can be useful to distribute calculation tasks.

Note that RT-Lab models are split as such:

Each Targets are defined in the same Simulink Model. Targets are subsystems called SC_"Something" for the console, SM_"Something" for the Master and SS_"Something" for slaves. Where "Something" is a name the user can choose.

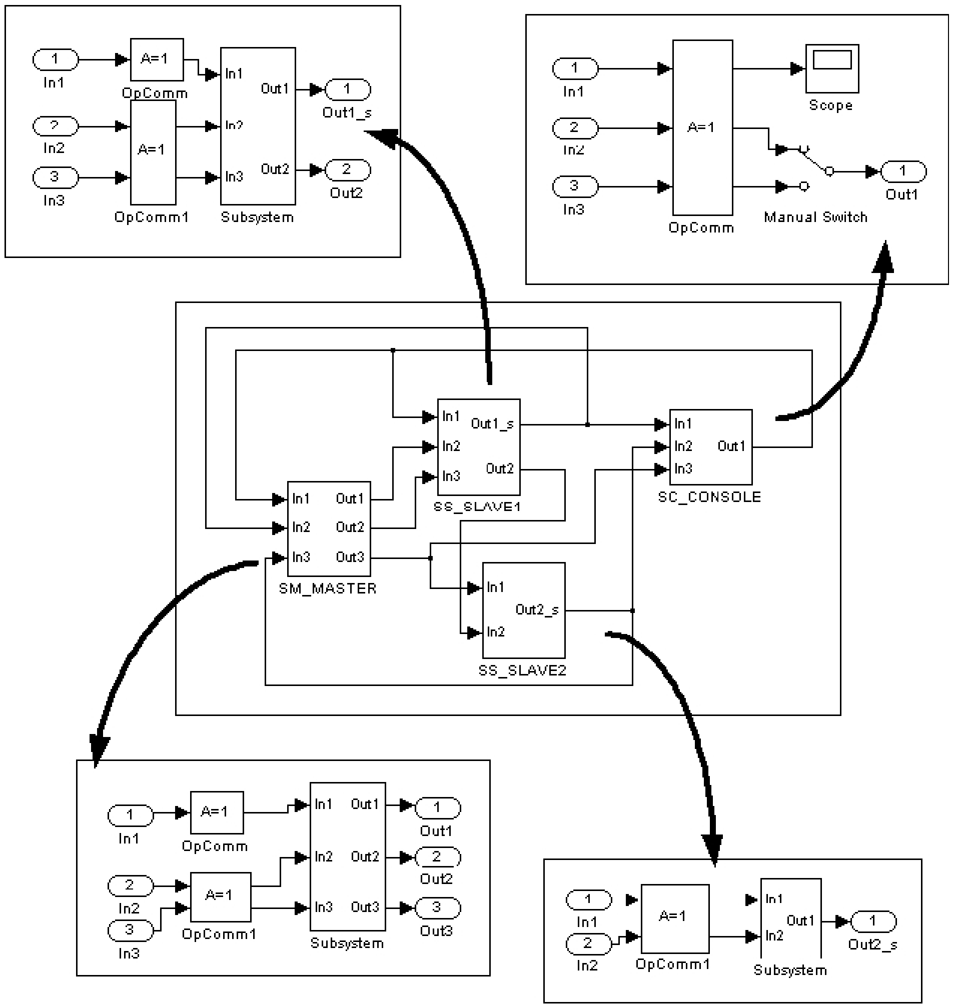


Figure 39:RT-Lab Model Separation

For more information about RT-Lab or RT-lab blocks, refer to the RT-Lab help located in the Matlab Help or to the RT-Lab Web site: **http://www.opal-rt.com/**

Annex 3: QuaRC documentation

This Annex presents QuaRC documentation and QuaRC demos that a new QuaRC user should explore to learn QuaRC basics.

1. Demo QuaRC

To access the QuaRC demo list, type qc_show_demos in the Matlab console.

Note that to execute demos, simply choose a QuaRC demo and follow the instructions.

The following demos are recommended for new QuaRC users:

- QuaRC Sine and Scope Demo (Only execute the windows target demo)
- QuaRC Data Logging Demo
- QuaRC Computation Time Demo
- QuaRC System Time Base Demo
- QuaRC Basic Communications Demo
- QuaRC Intermediate Communications Demo

2. Demo QBot

Type qc_show_demos in Matlab console and execute the following demo:

- Running Models using the QuaRC GumStix (Linux ARM) Target

Also executes the QuaRC QBot demo models delivered with the purchase of the QBot

- keyboard_control
- qbot_drive
- qbot_camera
- qbot_sensors

Note: It is recommended to execute demos presented in the section 1 before these ones.

3. Matlab help

For any help concerning QuaRC blocks or QuaRC functions, refer to the Matlab/Simulink help under the QuaRC Target tab.

4. IRobot user manual

It is recommended to read the QuaRC iRobot documentation that has been provided with the purchase of the QBot.

5. Quarc Installation Guide

It is recommended to read the QuaRC installation guide that has been provided with the purchase of QuaRC.

Annex 4: Experimental Setup Quick Start Guide

This Annex presents a quick start guide for executing wheeled mobile robot experiments using the QuaRC Wheeled Mobile Robot/Hardware In the Loop setup. This annex will present software and hardware requirements and software configurations.

1. Requirements

To be able to operate the OptiTrack + QuaRC setup, one must have the following hardware and software:

- A QuaRC computer with a wireless adapter and an Ethernet adapter.
- Valid QuaRC models with the proper Quarc_init.m file
- 1 or more QBot
- A Computer with OptiTrack Tracking Tools
- A version of VRPN Streamer configured to send data to the IP address of the QuaRC computer
- Infrared reflectors
- Calibration Wand
- A computer with X-Plane (Required only to visualize simulation results in real-time)

One would follow these steps:

- Log on computers
- Start X-Plane
- Calibrate the camera setup (Or open an existing one)
- Set the ground plane in the Tracking Tools Software (Only after a calibration)
- Place Infrared Reflectors on QBots (or the objects to track)
- Create Trackables in the Tracking Tools Software (Or Open an existing one)
- Configure the Tracking Tools Software to send Trackables data to a VRPN port
- Configure the VRPN Streamer to send the data to the QuaRC Computer
- Open QBots and place them in the playground
- Verify that the QuaRC computer receives the camera data.
- Open Models and Quarc_init.m
- Enable the wireless adapter of the QuaRC Computer
- Verify that QBots are connected to the QuaRC Computer
- Open remote consoles on QBots (Facultative)
- Launch Quarc_init.m

2. Additional documentation

There is also a video based on this document that is available. The video shows how to start a simulation using the camera setup and the QuaRC setup. It also shows how to calibrate and create Trackables with the OptiTrack Tracking tools software.

3. Log on computers

Make sure that the Ethernet adapter of each computer is connected to the Valcartier network, then log on using your personal login name and password. One would log on the QuaRC computer, the viewer computer (X-Plane computer) and the OptiTrack camera computer.

4. Start X-Plane

Launch X-Plane on the X-Plane computer.

5. Calibrate the camera setup

In order to calibrate, refer to the Tracking Tools video tutorial section: http://www.naturalpoint.com/optitrack/products/tracking-tools/videos.html.
One can then save the calibration by clicking on File->Save Calibration. One could open an existing calibration file and start using the setup right away. Note that a calibration file also saves the ground plane.

6. Set the ground plane

In order to set the ground plane properly, refer to the Tracking Tools video tutorial section: http://www.naturalpoint.com/optitrack/products/tracking-tools/videos.html.

Note that each model has been developed to use only the X and Y axis and ignore the Z axis (elevation).

7. Place Infrared Reflectors on QBots

The Figure 40 presents a QBot on which infrared reflectors are installed (white glowing balls mounted on wooden sticks). It is recommended to respect following tips when placing reflectors:

Place at least 4 markers on QBots to ensure that the Tracking Tools Software will be able to estimate the position and the angle of the QBot properly. In order to do so, one must not place markers in a way that a plan is formed by the marker set. The reason this constraint is that 2 possible angles can now be estimated from the camera system. When placing markers on more than one QBot, make sure that the form created by these marker is not too similar to others because the Tracking Tools Software will have difficulty differencing them.

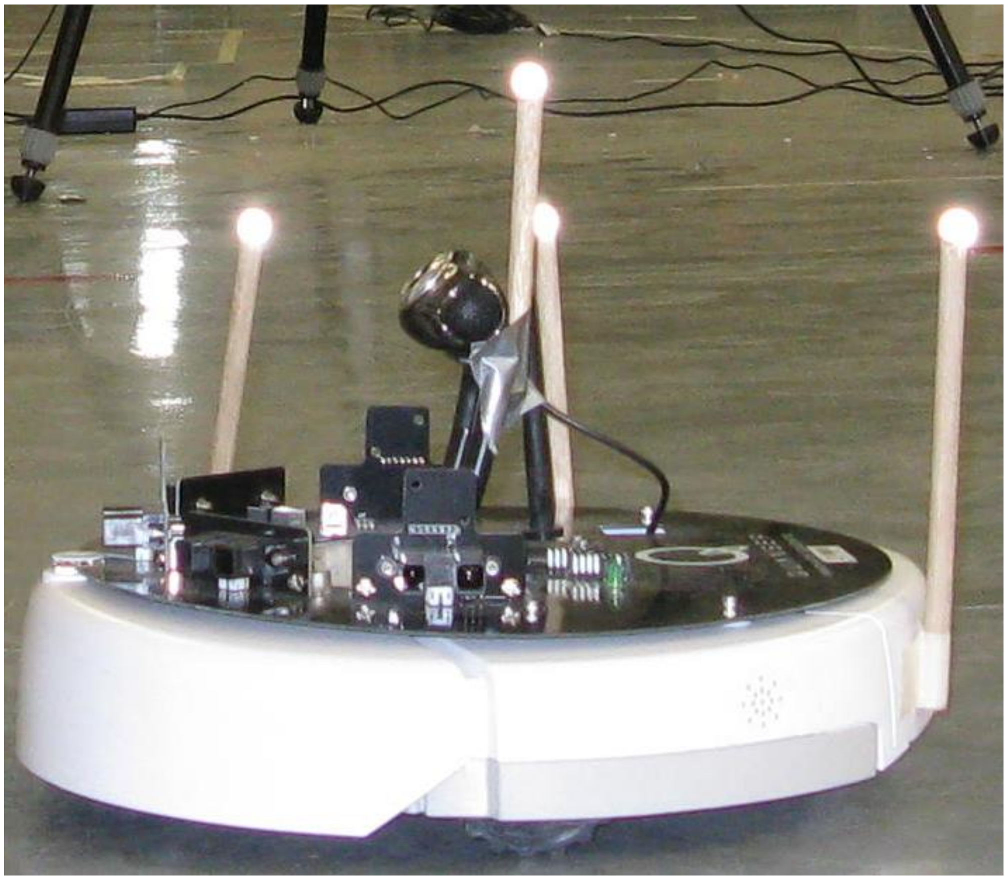
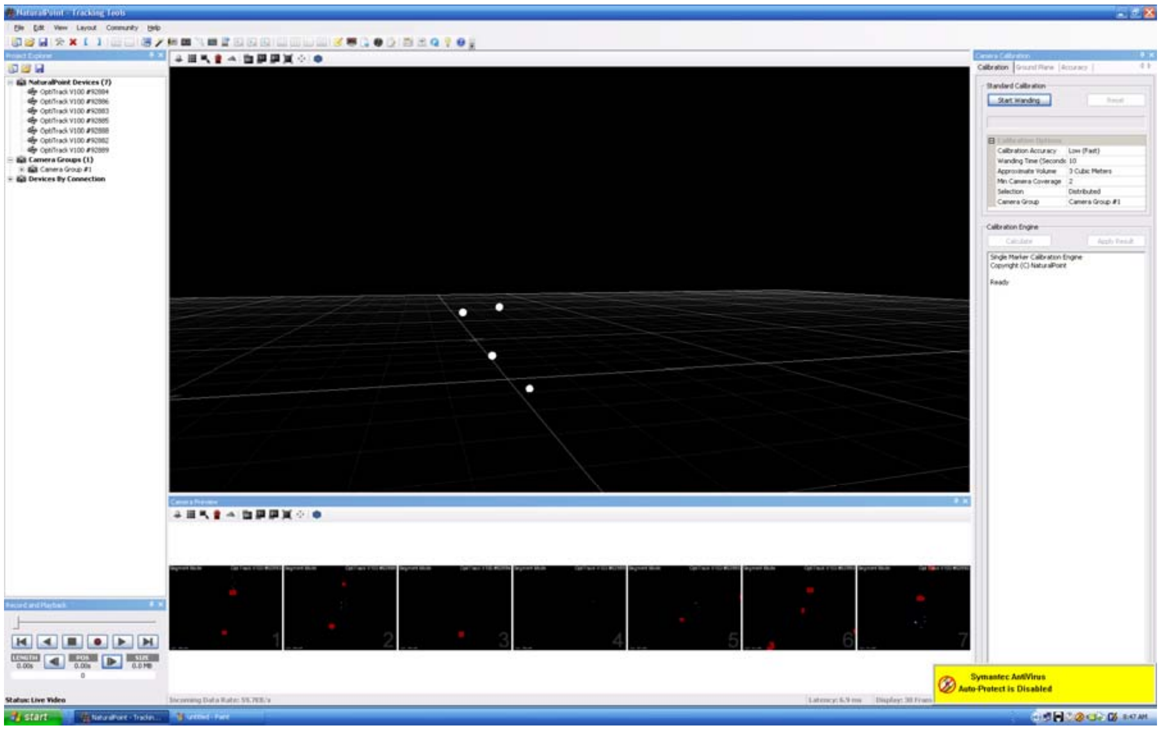


Figure 40: QBot with IR Reflectors

8. Create Trackables in the Tracking Tools Software

In order to create a trackable, refer to the Tracking Tools video tutorial section: http://www.naturalpoint.com/optitrack/products/tracking-tools/videos.html. One can then save Trackables by clicking on File->Save Trackables. One could open an existing trackable file and jump to the next step.

As you can see on the Figure 41, markers corresponding to the QBot are displayed in the Tracking Tools Main window. On the Figure 42, you can see that the markers has been selected (hold the left mouse button and cover the markers area, then release it). With the right mouse button, click on one of the selected markers and click on create trackable. One the Figure 43, one can see that the trackable has been created. On the right pane, there are trackable options. One can change Trackables' name. Changing the name of the trackable is required to send data to the VRPN Streamer since the current VRPN Streamer is configured to accept trackable with the following names: Tracker and Tracker2.

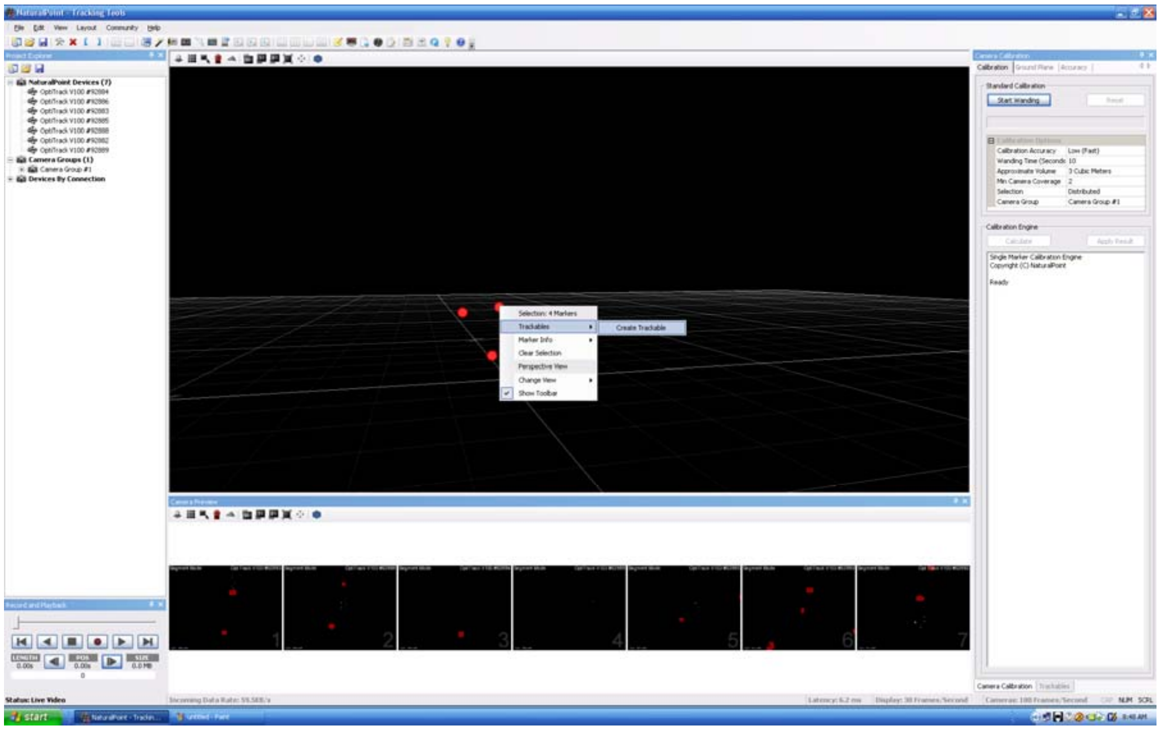


Calibration Accuracy	Low (Fast)
Wanding Time (Seconds	10
Approximate Volume	3 Cubic Meters
Min Camera Coverage	2
Selection	Distributed
Camera Group	Camera Group #1

Single Marker Calibration Engine
Copyright (C) NaturalPoint

Ready

Figure 41: QBot Markers

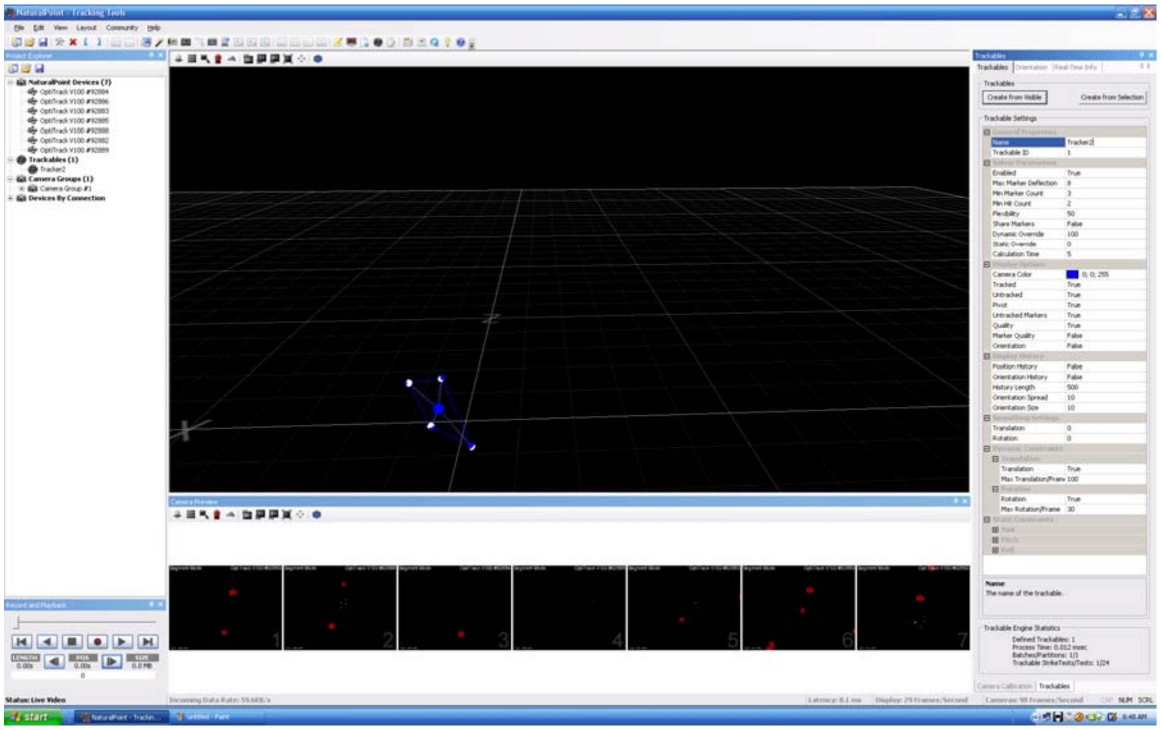


Calibration Accuracy	Low (Fast)
Wanding Time (Seconds	10
Approximate Volume	3 Cubic Meters
Min Camera Coverage	2
Selection	Distributed
Camera Group	Camera Group #1

```
Single Marker Calibration Engine
Copyright (C) NaturalPoint

Ready
```

Figure 42: Create Trackable from visible markers



Name	Tracker2
Trackable ID	1
Enabled	True
Max Marker Deflection	8
Min Marker Count	3
Min Hit Count	2
Flexibility	50
Share Markers	False
Dynamic Override	100
Static Override	0
Calculation Time	5
Camera Color	🟦 0; 0; 255
Tracked	True
Untracked	True
Pivot	True
Untracked Markers	True
Quality	True
Marker Quality	False
Orientation	False
Position History	False
Orientation History	False
History Length	500
Orientation Spread	10
Orientation Size	10
Translation	0
Rotation	0
Translation	True
Max Translation/Fram	100
Rotation	True
Max Rotation/Frame	30

Name
The name of the trackable.

Trackable Engine Statistics
Defined Trackables: 1
Process Time: 0.012 msec
Batches/Partitions: 1/1
Trackable StrikeTests/Tests: 1/24

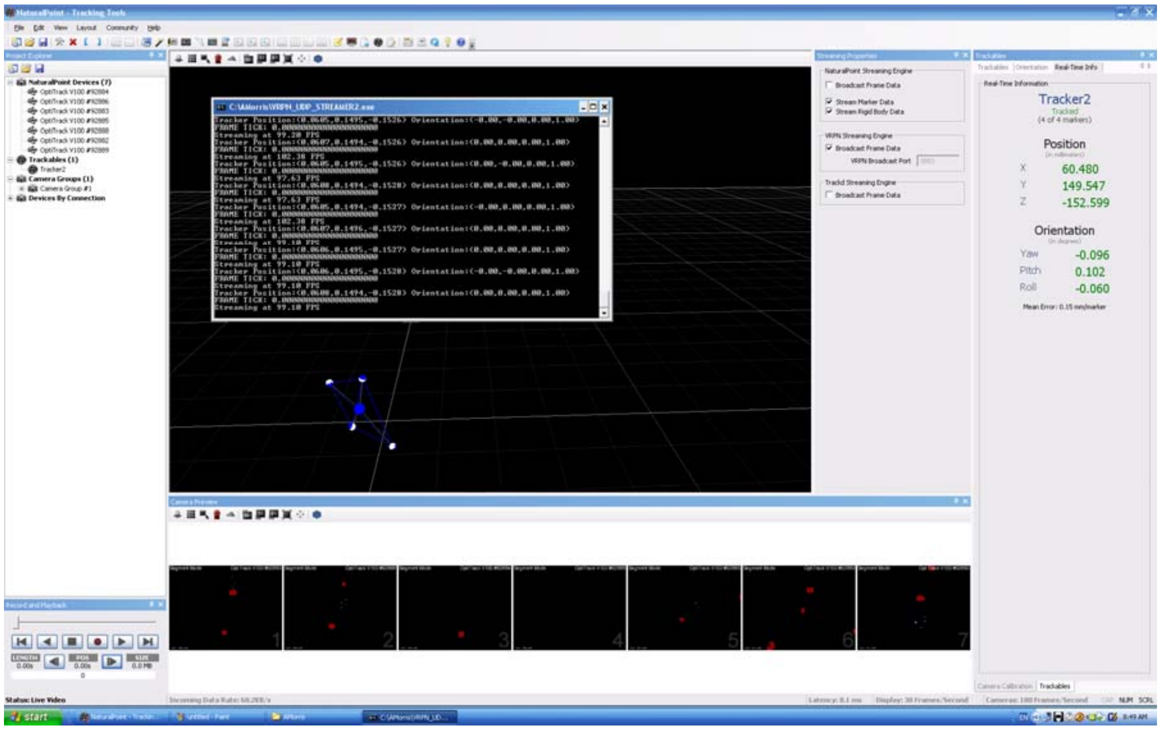
Figure 43: Trackable created

9. Configure Tracking Tools Software to send Trackable data

In order to send trackable data to the QuaRC computer, one must configure the Tracking Tools Software to send trackable data to the VRPN Streamer, and then the VRPN Streamer will send the Data to the QuaRC Computer. The VRPN Streamer is a software running on the OptiTrack Computer that retransmits trackable data from the Tracking Tools Software to the QuaRC Computer.

To send data to the VRPN Streamer, one would need to change the names of the tracker to valid names. The VRPN Streamer is configured to accept 2 Trackable: Tracker and Tracker2. One can see that a Trackables can be renamed on the trackable option pane as shown on the Figure 43.

To send data to the VRPN Streamer, the Tracking Tools Software must also be configured to broadcast on the VRPN port. In order to do so, click on view then click on streaming pane, then check "Broadcast VRPN Data" as shown on the Figure 44. Typically, the VRPN port used the VRPN Streamer is 3883. On the Figure 44, the VRPN streamer main window is shown. As you can see, there are data passing in this window. This means that the VRPN Streamer is sending vision data to another computer.



```
C:\Morris\VRPN_UDP_STREAMER2.exe
Tracker Position:(0.0605,0.1495,-0.1526) Orientation:(-0.00,-0.00,0.00,1.00)
FRAME TICK: 0.0000000000000000000000
Streaming at 99.20 FPS
Tracker Position:(0.0607,0.1494,-0.1526) Orientation:(0.00,0.00,0.00,1.00)
FRAME TICK: 0.0000000000000000000000
Streaming at 102.38 FPS
Tracker Position:(0.0605,0.1495,-0.1526) Orientation:(0.00,-0.00,0.00,1.00)
FRAME TICK: 0.0000000000000000000000
Streaming at 97.63 FPS
Tracker Position:(0.0608,0.1494,-0.1528) Orientation:(0.00,0.00,0.00,1.00)
FRAME TICK: 0.0000000000000000000000
Streaming at 97.63 FPS
Tracker Position:(0.0605,0.1494,-0.1527) Orientation:(-0.00,0.00,0.00,1.00)
FRAME TICK: 0.0000000000000000000000
Streaming at 102.38 FPS
Tracker Position:(0.0607,0.1496,-0.1527) Orientation:(0.00,0.00,0.00,1.00)
FRAME TICK: 0.0000000000000000000000
Streaming at 99.10 FPS
Tracker Position:(0.0606,0.1495,-0.1527) Orientation:(0.00,0.00,0.00,1.00)
FRAME TICK: 0.0000000000000000000000
Streaming at 99.10 FPS
Tracker Position:(0.0606,0.1495,-0.1528) Orientation:(-0.00,-0.00,0.00,1.00)
FRAME TICK: 0.0000000000000000000000
Streaming at 99.10 FPS
Tracker Position:(0.0608,0.1494,-0.1528) Orientation:(0.00,0.00,0.00,1.00)
FRAME TICK: 0.0000000000000000000000
Streaming at 99.10 FPS
```

NaturalPoint Streaming Engine
- [ ] Broadcast Frame Data
- [x] Stream Marker Data
- [x] Stream Rigid Body Data

VRPN Streaming Engine
- [x] Broadcast Frame Data
- VRPN Broadcast Port: 3883

Trackd Streaming Engine
- [ ] Broadcast Frame Data

Real-Time Information

Tracker2
Tracked
(4 of 4 markers)

Position (in millimeters)

X	60.480
Y	149.547
Z	-152.599

Orientation (in degrees)

Yaw	-0.096
Pitch	0.102
Roll	-0.060

Mean Error: 0.15 mm/marker

Figure 44: Streaming Pane

One can see the data on the VRPN Streamer main window as sown on the Figure 44.

10. Configure the VRPN Streamer to send the data to the QuaRC Computer

One must realize that the VRPN Streamer is the key software to send data from the OptiTrack computer to the QuaRC Computer. The VRPN Streamer must be configured to send trackable data to the IP address of the QuaRC Computer. Note that the VRPN Streamer is configured to send UDP packets to networked computers. It is possible to configure the port number and the IP address of the destination by recompiling the C++ VRPN Streamer project. QuaRC models developed to use the following UDP ports: 19000 and 19001.

11. Verify that the QuaRC computer receives the camera data.

Open the model that is used to receive the camera data (CamUDP.mdl is commonly used). Set the model to normal execution mode and click play. Verify that the FPS (Frame Per Second) box shows does not show 0. If so, there a connection issue.

Connection Issues can be:
- One or more Network cable are unplugged
- IP addresses are not configured properly either in the CamUDP model or in the VRPN Streamer.
- The camera system can not see QBots
- The OptiTrack software is not configured to stream Trackables on the VRPN port
- Trackables does not have the right name.

Note: One must initialize constants required to execute the model before launching it. Constants are commonly located in the Init.m file.

Note that in order to use CamUDP.mdl, it must be executed once in normal mode before being executed in external mode. It is sometimes required to launch the execution of this model in normal mode to reset the connection. For example, sometimes, CamUDP.mdl will fail to receive camera data after a compilation.

12. Open Models and Quarc_init.m

Open each models required to the simulation you wish to start and open the corresponding Quarc_init.m script file.

The reason to open required .mdl files and .m files at this moment is simple: Once a computer is disconnected from the Valcartier network, the windows file browser becomes very slow. Note that the effectiveness of Matlab, Simulink, Real-Time workshop and QuaRC remains unaffected.

13. Open QBots

Press the power button. The power led should light up. If the power is green, the QBot is ready. If the power led is not green or blinking, refer to the Quanser QBot user manual.

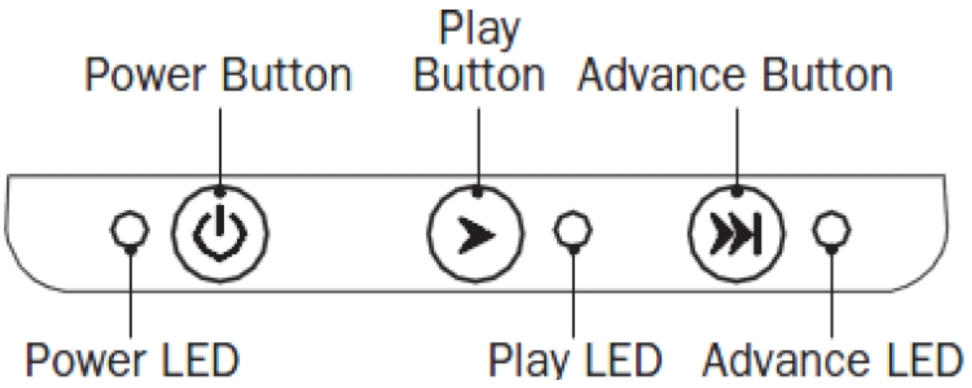


Figure 45:`QBot Button Layout

14. Enable the wireless adapter on the QuaRC computer

Since a computer with a wireless adapter should never be connected to the Valcartier network, one would have to unplug the Valcartier network while preserving a local network composed of the QuaRC computer, the X-Plane computer and the OptiTrack computer. The network configuration is shown on the Figure 46. It is obvious that the red arrow represents an Ethernet cable that connects the local network to the Valcartier network. By simply removing this Ethernet cable, it is now allowed to enable the wireless adapter on the QuaRC computer. When you do so, the IP address of all computers will remain unchanged. It is then possible to use the IP address of your computers even they are not connected to the Valcartier domain.

Important Note:

- It is important that the QuaRC computer, the X-Plane computer and OptiTrack computer stay connected since they need to exchange data during the simulation.
- The Valcartier network assigns computer IP addresses when a user logs in. It is then important that the user logs in before disconnecting the Valcartier network.

Once the wireless adapter is enabled, make sure to connect to the wireless ad-hoc network called GSAH. GSAH is an ad-hoc network broadcasted by QBots when they are running.

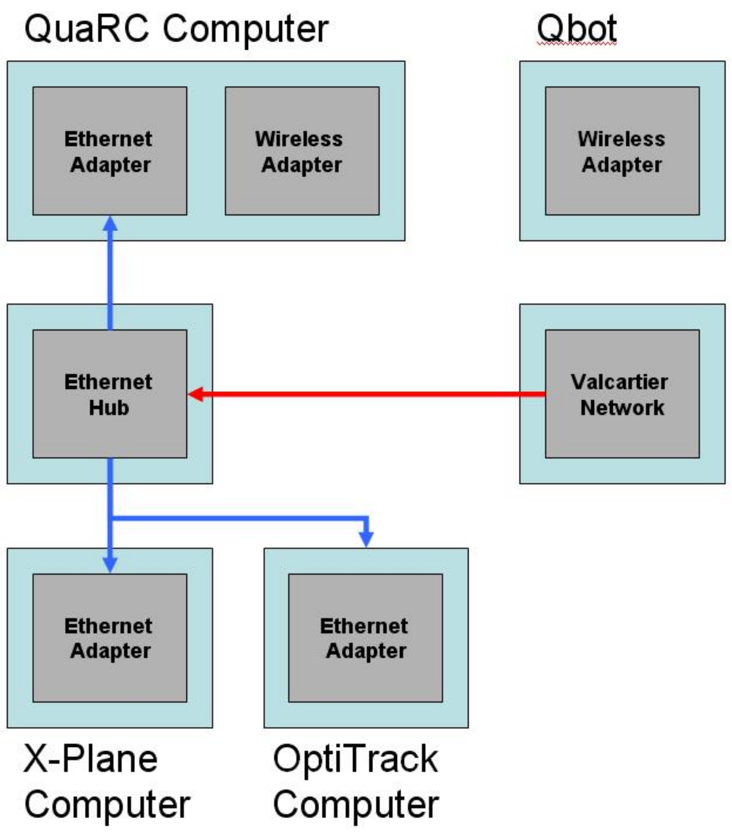


Figure 46: Network Management

15. Verify that QBots are connected to QuaRC computers and open remote consoles

One easy way to verify that QBots are connected to the QuaRC computer is to open a remote console on each QBot.

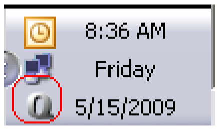
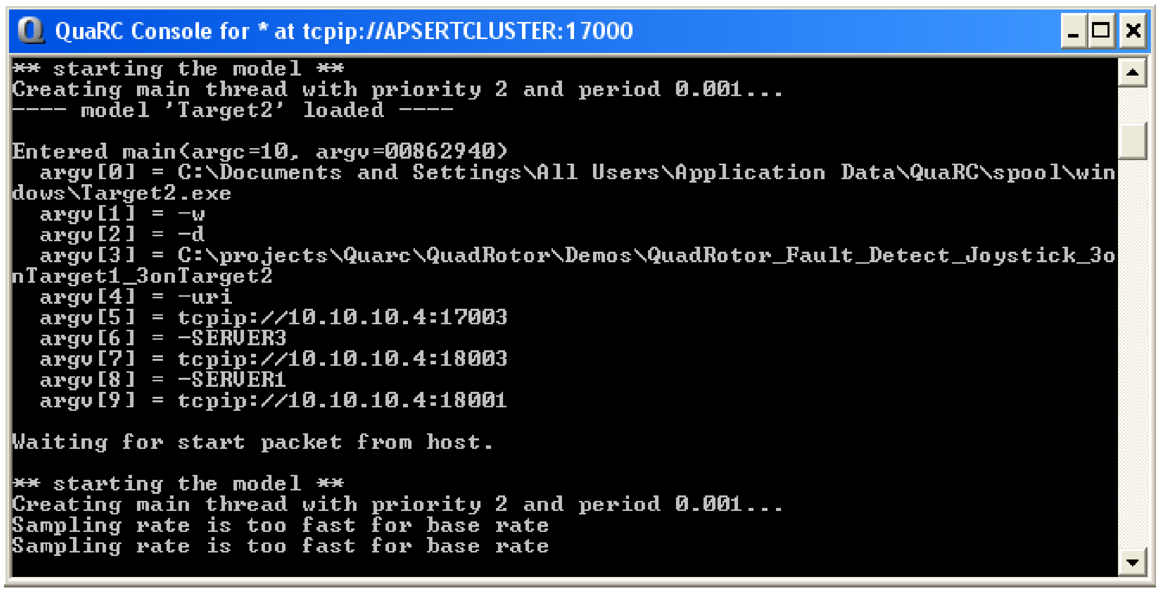


Figure 47: QuaRC Monitor

To access the console on a target that is running on another computer or even on the QBot, you can use the remote console.

- Click on the QuaRC monitor located in the task bar (see the Figure 47)
- Expand the Target tab
- Click on Remote…
- Enter the Target URI (QBots IP Addresses are written on QBots). The tcpip protocol must be used to reach a remote target. The port 17000 is always used to connect to a target. For example, use the following URI: tcpip://182.168.1.200:17000.
- Click OK
- Verify that the gray Q passes to green and has no exclamation mark. If there is and exclamation mark, the QuaRC monitor is attempting to connect to the target.
- Click again on the QuaRC monitor
- Click Console…
- Now you have access to the remote console.
- More than one QuaRC monitor can be opened at a time. This means that you can open a remote console on as many targets as you want. To open a new monitor, click on the Start menu -> All Programs -> Quanser -> QuaRC -> Monitor



```
** starting the model **
Creating main thread with priority 2 and period 0.001...
---- model 'Target2' loaded ----

Entered main(argc=10, argv=00862940)
  argv[0] = C:\Documents and Settings\All Users\Application Data\QuaRC\spool\win
dows\Target2.exe
  argv[1] = -w
  argv[2] = -d
  argv[3] = C:\projects\Quarc\QuadRotor\Demos\QuadRotor_Fault_Detect_Joystick_3o
nTarget1_3onTarget2
  argv[4] = -uri
  argv[5] = tcpip://10.10.10.4:17003
  argv[6] = -SERVER3
  argv[7] = tcpip://10.10.10.4:18003
  argv[8] = -SERVER1
  argv[9] = tcpip://10.10.10.4:18001

Waiting for start packet from host.

** starting the model **
Creating main thread with priority 2 and period 0.001...
Sampling rate is too fast for base rate
Sampling rate is too fast for base rate
```

Figure 48: QuaRC Console

16. Launch Quarc_init.m

Open each Simulink models required to the simulation you wish to start and open the corresponding Quarc_init.m script file.

Quarc_init.m is specifically made to build models, load and execute models, initialize constants and define model arguments. Note, there is one Quarc_init.m file for each simulations. Indeed, communication parameters are different for different simulation models.

For more information about the Quarc_init.m script file, refer to the Annex 8.

Annex 5: Configuring the VRPN Streamer Listening port and Broadcasting port

The VRPN streamer is a relay between the Tracking Tools software and the QuaRC computer. The VRPN streamer must then be connected to the Tracking Tools software and to the QuaRC computer.

In order to connect the VRPN Streamer to the Tracking Tools software, the VRPN Streamer must listen to the same VRPN port that Tracking Tools uses to broadcast visual data. Typically, the Tracking Tools software uses the VRPN port # 3883.

In order to connect the VRPN Streamer to the QuaRC computer, it must send data to the QuaRC computer's IP address and configured UDP ports.

Both changing the VRPN listening port and the UDP streaming address and port requires recompiling the VRPN C++ project.

To change the IP address or the port on which to send data, simply modify the following line of the VRPN-Listener.cpp file :

```
string tIP = "131.132.57.12"; // IP du PC executant xPC Target
unsigned short tPORT = 19000; // Port UDP utilisé sur le PC executant
xPC Target
unsigned short tPORT1 = 19001;
```

Where: 131.132.57.12 is the IP address of the computer that will receive the UDP data (QuaRC computer). UDP ports used here are 19000 and 19001. There are 2 ports used here because the VRPN Stream can handle 2 rigid bodies (Tracker and Tracker2 by default). Rigid bodies must be called Tracker and Tracker2 in the OptiTrack software.

To modify the VRPN port on which this program listens, modify these lines of the VRPN-Listener.cpp file:

```
    if(UseUDP)
        connection = new vrpn_Connection("localhost", 3883);
    else
 connection = new vrpn_Connection("tcp://localhost", 3883);
```

The VRPN port used here is 3883.

To modify the rigid bodies' name, modify the following lines:

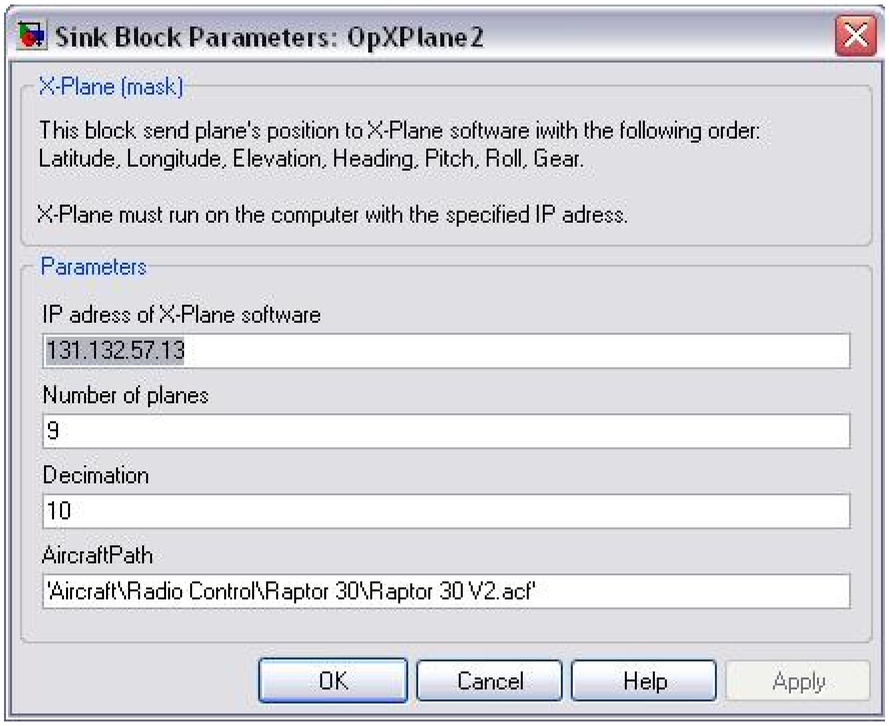
```
vrpn_Tracker_Remote  *tracker2  =  new  vrpn_Tracker_Remote("Tracker2",
connection);
```

```
 vrpn_Tracker_Remote  *tracker  =  new  vrpn_Tracker_Remote("Tracker",
connection);
```

Where: Tracker and Tracker2 are the rigid bodies names used in the Tracking Tools software.

Annex 6: Configuring X-Plane

The Figure 49 shows the X-Plane block parameters. As you can see, the IP Address of the X-Plane computer is required. The second parameter allows the user to set the number of planes to plot in X-Plane. Note that the maximum number of planes is 10 with the X-Plane version 8.40. The third parameter is the decimation which is used to reduce the amount of data sent to X-Plane. Indeed the decimation is simply a scale of down sample. Indeed, if the decimation is 10, vehicles position will be update 10 times less often than they are calculated by the simulation model. The fourth parameter is the 3D aircraft X-Plane models (.acf files) which will be used to display UAVs in the three dimensional environment. Each acf file is located in the X-Plane installation folder.


Sink Block Parameters: OpXPlane2

X-Plane (mask)

This block send plane's position to X-Plane software iwith the following order:
Latitude, Longitude, Elevation, Heading, Pitch, Roll, Gear.

X-Plane must run on the computer with the specified IP adress.

Parameters

IP adress of X-Plane software: 131.132.57.13

Number of planes: 9

Decimation: 10

AircraftPath: 'Aircraft\Radio Control\Raptor 30\Raptor 30 V2.acf'

Figure 49: X-Plane Block Parameters

Annex 7:Synchronizing QuaRC Models

When a simulation is distributed into more than one model, each model may require to be started at the same time. Since every models are started sequentially, models must be enabled when model are loaded. In order to start models at the same time, models are composed of enabled subsystems. These subsystems are enabled when every model are loaded (or connected). This will ensure that there is no delay between each enabled subsystems. The Figure 50 shows an enabled subsystem which the trigger signal indicates that each communication blocks are connected and exchanging data.

As an example, if there is a simulation composed of 2 Simulink models (model A and model B). The model A contains an UAV which leads a formation. The model B contains UAVs following the leader. If the synchronization is not done between the model A and the model B, it is easy to see that if the model B is started first, followers model will be executed before the leader's model. It is easy to see that a similar problem will occur if the model A is started first. A synchronization is then required.

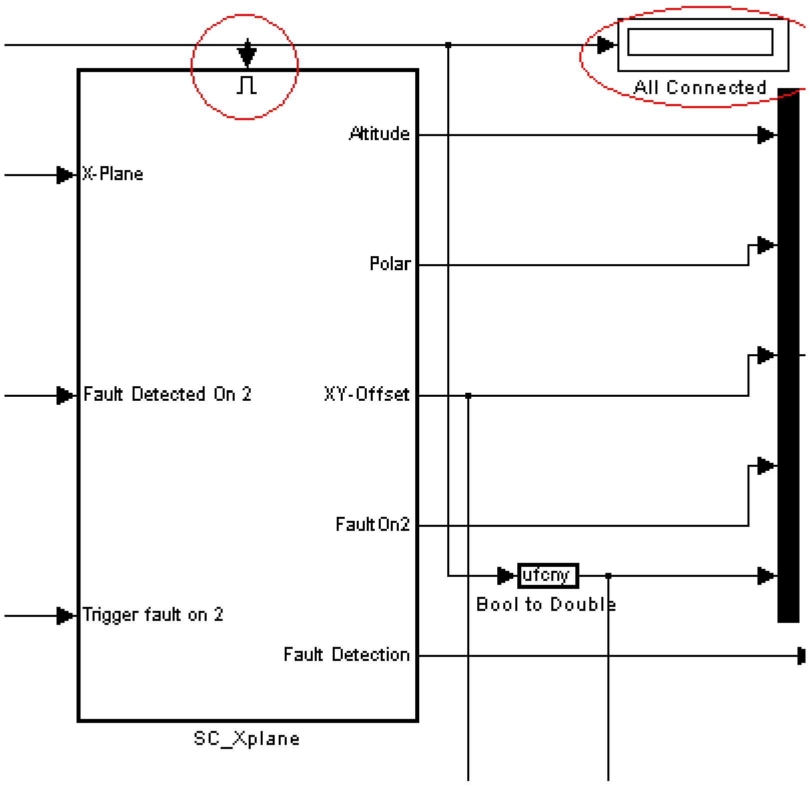


Figure 50: Synchronizing models

Annex 8:Quarc_init.m

There are several reasons for using Quarc_init.m. For instance, when executing more than one model at a given time (distributed simulations), it is required to input default model and target URI manually in the QuaRC preferences between each compilation and execution. By proceeding that way, one would need to wait until a model is loaded to a target, then change the default model and target URI for the next model in order to load it. Proceeding that way is time inefficient and induces many configuration mistakes. For such a reason, Quarc_init.m has been developed to automatize the process of building, assigning URIs and loading models.

To build a model, you must follow a specific order of operations:

- Defining the model and the target default URI
- Build and load the model
- Connect to the model
- Start the model

Each of these steps is done by the Quarc_init.m script file.

```
%------------------------------------ %
%                                     %
% Target, Model and Server Configuration
%                                     %
%------------------------------------ %


%apseporcoop -  182.168.1.100
%QBOT -         182.168.1.202

%close all
%clear all

%% Constant definition these files are different for each models
LTAVNOVR_PLOTS
Init

%% Targets configuration

%Set the number of target to 0 (do not edit)
nTarget=0;

%Define servers that will be used by the comm blocks
Server1 = '-SERVER1 tcpip://182.168.1.202:18001';
Server2 = '-SERVER2 tcpip://182.168.1.202:18002';
Server6 = '-SERVER6 tcpip://182.168.1.202:18003';

%% Model 1

%Increase the number of target by 1
nTarget = nTarget+1;

%Name of the model
ConfigStruct.ModelInfo(nTarget).Name = 'CamUDP';

%Model URI - Since the model is executed on the machine that compile
and
%lauche the execution, the shared memory can be used
ConfigStruct.ModelInfo(nTarget).ModelUri = 'shmem://quarc-target:11';

%Target URI - Since the model is executed on the machine that compile
and
%lauche the execution, the shared memory can be used
```

```
ConfigStruct.ModelInfo(nTarget).TargetUri = 'shmem://quarc-target:1';

%Define the target type - This one is executed on a windows machine
ConfigStruct.ModelInfo(nTarget).TargetType = 'windows';

%The protocol used to communicate with the target is shared memory
ConfigStruct.ModelInfo(nTarget).Protocol = 'shmem';

%This model only uses the Server 6 to communicate with the QBot
ConfigStruct.ModelInfo(nTarget).Arguments = sprintf('%s', Server6);

%Always set this to Yes, it is used to stop the model with Quarc_stop.m
%(Yes/No)
ConfigStruct.ModelInfo(nTarget).Running = 'Yes';

%Decide wether you wish to stay connected with the model during the
%execution (Yes/No)
ConfigStruct.ModelInfo(nTarget).StayConnected = 'Yes';

%Decide if that model needs to be compiled (Yes/No)
ConfigStruct.ModelInfo(nTarget).Compile = 'No';

%% Model 2
%Increase the number of target by 1
nTarget = nTarget+1;

%Name of the model
ConfigStruct.ModelInfo(nTarget).Name = 'ALTAV';

%Model URI - Since the model is executed on the machine that compile
and
%lauche the execution, the shared memory can be used
ConfigStruct.ModelInfo(nTarget).ModelUri = 'shmem://quarc-target:12';

%Target URI - Since the model is executed on the machine that compile
and
%lauche the execution, the shared memory can be used
ConfigStruct.ModelInfo(nTarget).TargetUri = 'shmem://quarc-target:1';

%Define the target type - This one is executed on a windows machine
ConfigStruct.ModelInfo(nTarget).TargetType = 'windows';

%The protocol used to communicate with the target is shared memory
ConfigStruct.ModelInfo(nTarget).Protocol = 'shmem';

%This model uses the Server 1 and the Server 2 to communicate with the
%Qbot_Control model
ConfigStruct.ModelInfo(nTarget).Arguments = sprintf('%s', Server1,' ',
Server2);

%Always set this to Yes, it is used to stop the model with Quarc_stop.m
%(Yes/No)
ConfigStruct.ModelInfo(nTarget).Running = 'Yes';

%Decide wether you wish to stay connected with the model during the
%execution (Yes/No)
ConfigStruct.ModelInfo(nTarget).StayConnected = 'Yes';

%Decide if that model needs to be compiled (Yes/No)
ConfigStruct.ModelInfo(nTarget).Compile = 'No';

%% Model 3

%Increase the number of target by 1
```

```
nTarget = nTarget+1;

%Name of the model
ConfigStruct.ModelInfo(nTarget).Name = 'Qbot_Control';

%Model URI - Since the model is executed on the Qbot you have to use
the
%tcpip protocol to communicate with it
ConfigStruct.ModelInfo(nTarget).ModelUri                              =
'tcpip://182.168.1.202:17003';

%Target URI - Since the model is executed on the Qbot you have to use
the
%tcpip protocol to communicate with it
ConfigStruct.ModelInfo(nTarget).TargetUri                             =
'tcpip://182.168.1.202:17000';

%Define the target type - This one is executed on a linux_arm machine
%(The Qbot board is a linux_arm machine)
ConfigStruct.ModelInfo(nTarget).TargetType = 'linux_arm';

%The protocol used to communicate with the target is tcp
ConfigStruct.ModelInfo(nTarget).Protocol = 'tcpip';

%The Qbot Use the Server 1 and Server 2 to communicate with the Altav
model
%and the Server 6 to communicate with the CamUDP model
ConfigStruct.ModelInfo(nTarget).Arguments  =  sprintf('%s',  Server1,'
',Server2,' ', Server6);

%Always set this to Yes, it is used to stop the model with Quarc_stop.m
%(Yes/No)
ConfigStruct.ModelInfo(nTarget).Running = 'Yes';

%Decide wether you wish to stay connected with the model during the
%execution (Yes/No)
ConfigStruct.ModelInfo(nTarget).StayConnected = 'Yes';

%Decide if that model needs to be compiled (Yes/No)
ConfigStruct.ModelInfo(nTarget).Compile = 'No';

%% Building, loading models

%Display the number of model
nTarget

%For each model, compile if requested
for i=1:nTarget
    if(strcmp('Yes',ConfigStruct.ModelInfo(i).Compile)) %Verify if the
compilation is requested
        qc_set_default_target_uri(ConfigStruct.ModelInfo(i).Protocol,
ConfigStruct.ModelInfo(i).TargetUri) %Set the target URI
        qc_set_default_model_uri(ConfigStruct.ModelInfo(i).ModelUri)
%Set the model URI
        qc_build_model(ConfigStruct.ModelInfo(i).Name) %Build the model
    end
end


%-------------------------------------- %
%                                       %
% Execute models                        %
%                                       %
%-------------------------------------- %
```

```
%Load and execute each model
for i=1:nTarget
    qc_set_default_target_uri(ConfigStruct.ModelInfo(i).Protocol,
ConfigStruct.ModelInfo(i).TargetUri) %Set the target URI

qc_set_default_model_uri(ConfigStruct.ModelInfo(i).TargetType,ConfigStr
uct.ModelInfo(i).ModelUri) %Set the model URI
    qc_load_model(ConfigStruct.ModelInfo(i).Name,sprintf('%s','-w -d %d
-uri %u',' ',ConfigStruct.ModelInfo(i).Arguments)) %Load the model with
the proper arguments
    qc_connect_model(ConfigStruct.ModelInfo(i).Name)  %  Connect  to  the
model
    qc_start_model(ConfigStruct.ModelInfo(i).Name) % Start the model

    if ~strcmp(ConfigStruct.ModelInfo(i).StayConnected, 'Yes')
        qc_disconnect_model(ConfigStruct.ModelInfo(i).Name);
%Disconnect if requested
    end

end
```

DOCUMENT CONTROL DATA

1. ORIGINATOR (name and address) Numerica Technologies Inc	2. SECURITY CLASSIFICATION (Including special warning terms if applicable) UNCLASSIFIED	
3. TITLE (Its classification should be indicated by the appropriate abbreviation (S, C, R or U) Real-time Simulations using QuaRC and RT-LAB and Development of a Hardware-in-the-Loop Indoor Facility for Robot Formations		
4. AUTHORS (Last name, first name, middle initial. If military, show rank, e.g. Doe, Maj. John E.) Alexandre Morris, Pierre Gosselin		
5. DATE OF PUBLICATION (month and year) September 2009	6a. NO. OF PAGES 96	6b .NO. OF REFERENCES 14
7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. Give the inclusive dates when a specific reporting period is covered.) Contract report CR 2009-499		
8. SPONSORING ACTIVITY (name and address)		
9a. PROJECT OR GRANT NO. (Please specify whether project or grant)	9b. CONTRACT NO.	
10a. ORIGINATOR'S DOCUMENT NUMBER	10b. OTHER DOCUMENT NOS N/A	

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

- [✓] Unlimited distribution
- ☐ Restricted to contractors in approved countries (specify)
- ☐ Restricted to Canadian contractors (with need-to-know)
- ☐ Restricted to Government (with need-to-know)
- ☐ Restricted to Defense departments
- ☐ Others

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

Unlimited

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

This report will present two commercial software environments used to distribute and execute real-time simulations: QuaRC and RT-Lab. Both QuaRC and RT-Lab allow the user to develop simulation models using Matlab/Simulink and include hardware, such as data acquisition boards, to connect to real vehicles and systems. In addition, QuaRC can be used to program embedded systems such as wheeled mobile robots and aerial vehicles.

This report will present formation flight models that have been modified in order to be compliant to QuaRC or RT-Lab. The simulations are composed of six to ten unmanned aerial vehicles, or UAVs, following a commanded trajectory while maintaining a prescribed trajectory. Models presented also include abrupt fault detection and formation shape morphing on operator's request. Vehicle models and dynamics are based on almost lighter-than-air (ALTAV) vehicles, unicycles and quadrotor vehicles. Low-level controllers used to stabilize these UAVs are feedback linearization controllers. Formation controller is of leader-to-follower type. Simulation results are displayed in real-time on a three-dimensional viewer (X-Plane).

The feedback linearization controller has been implemented on an embedded computer on board a wheeled mobile robot (QBot). An infrared camera system (OptiTrack camera setup) is used to measure the QBot's position and orientation. This information is then sent from the base station to the wheeled mobile robot's embedded computer using a wireless link in order to close the low-level controller's loop.

This report will then present major differences between QuaRC and RT-Lab as well as advantages and inconvenient of using either software solution.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

RT-Lab

QuaRC

Real-time formation flight control: simulations and experiments.

UGV

Wheeled Mobile Robot Trajectory Tracking