1 About this document

1.1 Attention Symbols

The following symbols are used throughout this document to draw attention to important operating information, special instructions, and warnings.

Note - Pertinent information that clarifies a process, operation, or ease of use preparations regarding the product.

Warning - Instructions to avoid harming yourself or damaging the equipment.

1.2 Revision History

Revision	Date	Author	Changes
A	09/03/2011	Laurent DUCRET	Creation of the document

2 Product description

2.1 Presentation

The RobuROC4 is a differential platform with 4 independent wheels integrating the following components:

- 4 servo drives and resolvers to control the wheels motors
- 2 ultrasonic telemeters
- 2 bumpers
- 1 wireless emergency stop system
- 1 wireless game pad



The platform can be controlled in 2 modes:

- Game pad mode (a game pad allows to move the platform)
- UDP mode (UDP telegrams allow the user to make its own application)

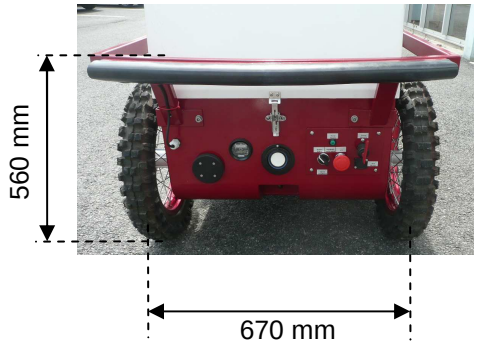
Each of these modes can control the system with different priorities.

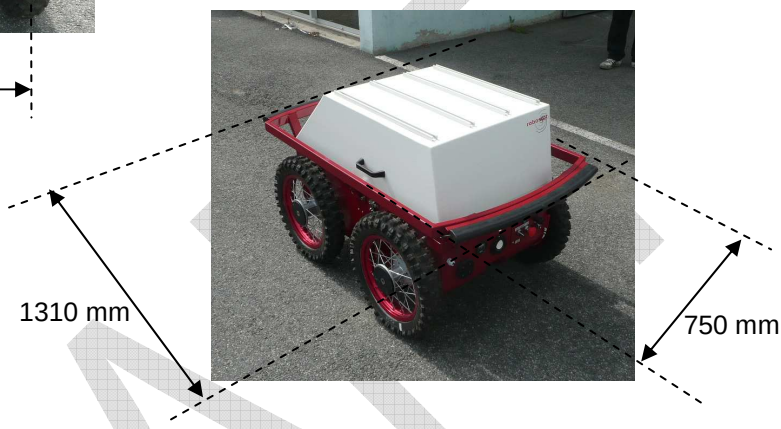
Highest priority : Game pad mode
Lowest priority : UDP mode

These priorities on the control modes allow for example to use UDP mode and take temporary the control of the platform using the game pad in case of problem

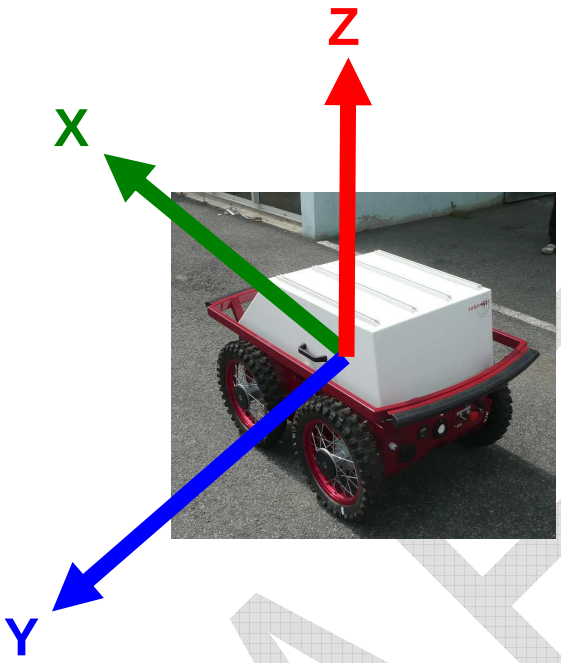
2.2 Dimensions

Main dimensions of the vehicle are 750 x 1310 x 770 mm





2.3 Coordinate system



3 Quick start

3.1 Requirements

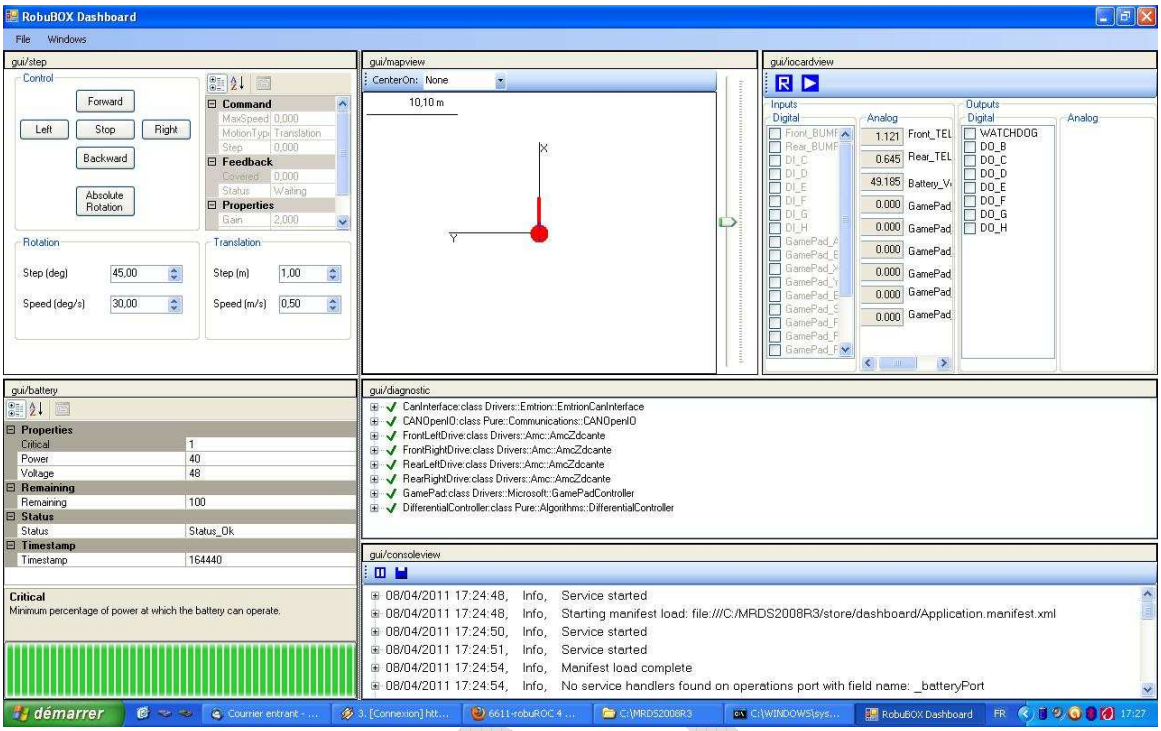
- Pc with Windows XP SP2 and application « ***Microsoft CCR and DSS Runtime 2008 R3 Redistributable.exe*** » installed
- RJ45 network cable
- Application « DashBoard » RobuBOX

3.2 Start procedure

- Check nothing is in the environment of the platform
- Turn ON the main power switch
- Check the brakes mode switch is in AUTOMATIC position
- Release the local emergency stop
- Release the remote emergency stop
- Enable the remote control pressing its green button
- Press the green button of the platform's control panel

- Press the  button of the gamepad to turn it ON
- Refer to the "4.1Game pad mode" section to move the platform using the gamepad

- To be able to use the DashBoard application, connect a laptop with an Ethernet wire
- Start the DashBoard application to receive the platform's status

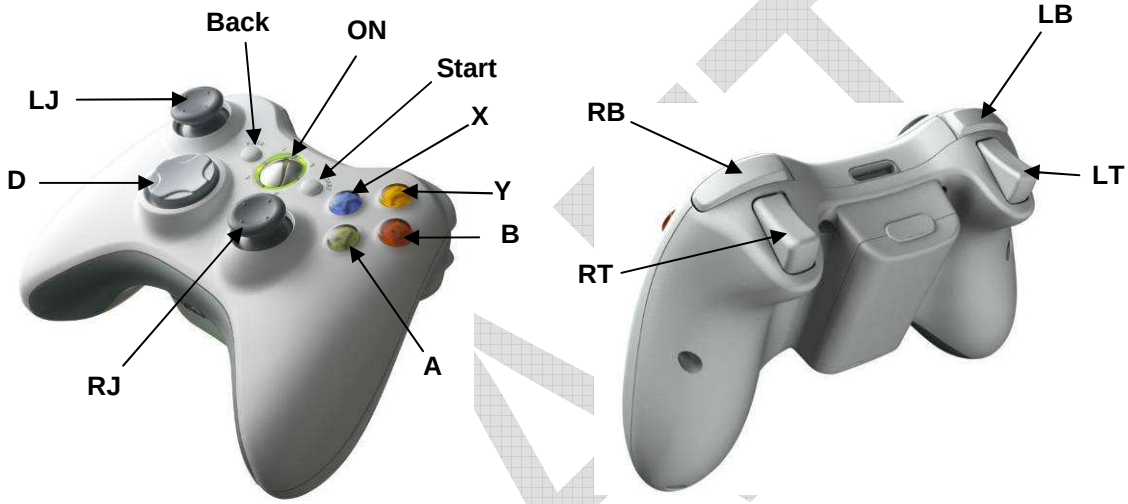

RobuBOX Dashboard
File
Windows
gui/step
Control
Forward
Left
Stop
Right
Backward
Absolute Rotation
Command
Feedback
Properties
Rotation
Step (deg)
45,00
Speed (deg/s)
30,00
Translation
Step (m)
1,00
Speed (m/s)
0,50
gui/mapview
CenterOn: None
10,10 m
gui/iocardview
Inputs
Digital
Analog
Outputs
Digital
Analog
WATCHDOG
DO_B
DO_C
DO_D
DO_E
DO_F
DO_G
DO_H
gui/battery
Properties
Critical 1
Power 40
Voltage 48
Remaining
Remaining 100
Status
Status Status_Ok
Timestamp
Timestamp 164440
Critical
Minimum percentage of power at which the battery can operate.
gui/diagnostic
CanInterface:class Drivers::Emtrion::EmtrionCanInterface
CANOpenIO:class Pure::Communications::CANOpenIO
FrontLeftDrive:class Drivers::Amc::AmcZdcante
FrontRightDrive:class Drivers::Amc::AmcZdcante
RearLeftDrive:class Drivers::Amc::AmcZdcante
RearRightDrive:class Drivers::Amc::AmcZdcante
GamePad:class Drivers::Microsoft::GamePadController
DifferentialController:class Pure::Algorithms::DifferentialController
gui/consoleview
08/04/2011 17:24:48, Info, Service started
08/04/2011 17:24:48, Info, Starting manifest load: file:///C:/MRDS2008R3/store/dashboard/Application.manifest.xml
08/04/2011 17:24:50, Info, Service started
08/04/2011 17:24:51, Info, Service started
08/04/2011 17:24:54, Info, Manifest load complete
08/04/2011 17:24:54, Info, No service handlers found on operations port with field name: _batteryPort
démarrer
RobuBOX Dashboard
17:27

4 Control modes

4.1 Game pad mode

It is the default mode of the platform.
Maintain the [A] button to activate the game pad mode. UDP commands are then ignored but the return of UDP status is still active.

Attention : When the [A] button is released, the UDP mode resumes control



Action	Game pad button	
Turn ON the game pad		Hold this button 3 seconds to power the game pad. The green LEDs must stay fixed.
Activate game pad mode	[A]	Hold this button to activate this mode. If this button is released, the UDP mode is active.
Forward speed	[RT]	Gradually press this button to reach the desired linear speed
Backward speed	[LT]	Gradually press this button to reach the desired linear speed
Rotation speed	[LJ]	Gradually move this button to the right or the left to reach the desired angular speed

4.2 UDP mode

4.2.1 Presentation

It is possible to create its own software to control the system. The computer used should be connected to the onboard computer system by a network interface.

Comments:

- The software created must be a UDP client
- The UDP server has the following properties :
  - **IP** : 192.168.0.2
  - **port** : 60000

- During data exchanges, when data has several bytes, the lower byte is sent first, the higher byte last.
- Take care of the headers values (it change from a telegram to the other)

The UDP communication is based on software services. A service is a software component allowing controlling a specific device of the platform. The different services can be accessed (send commands, receive status) using their IDs:

Service ID	Service Name	Description
0 (0x0000)	Directory	This service lists other services. This service is an entry point for all applications using this protocol. The first element of the list is the directory service itself
1 (0x0001)	Notification	This service is in charge of sending periodically the status of the other services. To receive the services' status, a subscription (specifying the notification period) is needed
2 (0x0002)	IOCard	This service allows to • set the digital and analog outputs of the platform • read the digital and analog inputs of the platform • read the game pad buttons' states
3 (0x0003)	Telemeter	This service groups the telemeters' values (ultrasonic sensors)
4 (0x0004)	Battery	This service indicates the battery level of the platform
5 (0x0005)	Localization	This service contains odometry data
6 (0x0006)	Differential	Use this service to move the platform

- Refer to the properties and description of the directory service to get the services' IDs
- Refer to the appendix section in order to know the software links between hardware and the services

For more information on the communication protocol, refer to the PURE documentation:
http://www.doc-center.robosoft.com/Robubox_home_page/RobuBOX_Pure_User_Manual

4.2.2 Services properties

It is possible to receive the properties of each service. To receive data, the request telegram must be sent

It is important to get properties so that status telegrams can be properly decoded and command telegrams can be correctly sent

4.2.2.1 General request properties telegram

When this datagram is sent, properties are received as an answer

Header (0x01)	Action (0x00)	Service ID

Field	Field size	description
Header	1 byte – type unsigned char	Header of the datagram. Value :1 (0x01)
Action	1 byte – type unsigned char	GET Value : 0 (0x00)
Service ID	2 bytes – type short	ID of the service that has to send properties (refer to 4.2.1 section to get the services' IDs)

4.2.2.2 Directory properties telegram

Send a request telegram to receive these properties

Header (0x01)	Action	Target	Error	Type_1	ID_1	• • •	Type_**n**	ID_**n**

Field	Field size	description
Header	1 byte – type unsigned char	Header of the datagram. Value :1 (0x01)
Action	1 byte – type unsigned char	GET Value : 0 (0x00)
Target	2 bytes – type short	Instance number of the service that sends properties Value : ID of the service that sends its properties (refer to 4.2.1 section to get the services' IDs)
Error	1 byte – type unsigned char	• 0 (0x00) : Ok, no error • Other value : error
Type_1	2 bytes – type UInt16	Type of the first service (see description for a string description)
ID_1	2 bytes – type UInt16	ID of the first service (this ID is used for UDP communication)
…	**…**	**…**
Type_**n**	2 bytes – type UInt16	Type of the **n**th service (see description for a string description) The number **n** of services can be computed using the number of bytes **size** of the received UDP message : **n** = (**size** - 5) / 4
ID_**n**	2 bytes – type UInt16	ID of the **n**th service (this ID is used for UDP communication)

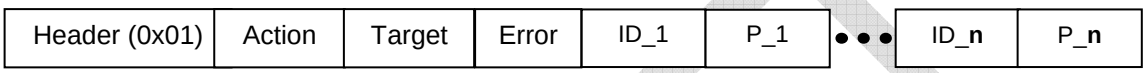
		The number **n** of services can be computed using the number of bytes **size** of the received UDP message : **n** = (**size** - 5) / 4

4.2.2.3 Notification properties telegram

It is possible to receive periodically the status of each service. To receive this status, it is mandatory to send a subscribe telegram for each service (only one time, at the startup of your application for example) to the notification service.

The properties of the notification service contain the list of the services that will send status periodically.

Send a request telegram to receive these properties



Field	Field size	description
Header	1 byte – type unsigned char	Header of the datagram. Value :1 (0x01)
Action	1 byte – type unsigned char	GET Value : 0 (0x00)
Target	2 bytes – type short	Instance number of the service that sends properties Value : ID of the service that sends its properties (refer to 4.2.1 section to get the services' IDs)
Error	1 byte – type unsigned char	• 0 (0x00) : Ok, no error • Other value : error
ID_1	2 bytes – type UInt16	ID of the first service recorded for status notifications
P_1	1 byte – type unsigned char	Period of the status sending of the first service. (Value is number of low level cycles, that means number of 10ms) Example : 10 (0x0A) ➔ 100ms
…	…	…
Type_**n**	2 bytes – type UInt16	ID of the **n**th service recorded for status notifications The number **n** of services can be computed using the number of bytes **size** of the received UDP message : **n** = (**size** - 5) / 3
ID_**n**	1 byte – type unsigned char	Period of the status sending of the **n**th service. (Value is number of low level cycles, that means number of 10ms) Example : 10 (0x0A) ➔ 100ms The number **n** of services can be computed using the number of bytes **size** of the received UDP message : **n** = (**size** - 5) / 3

4.2.2.4 IOCard properties telegram

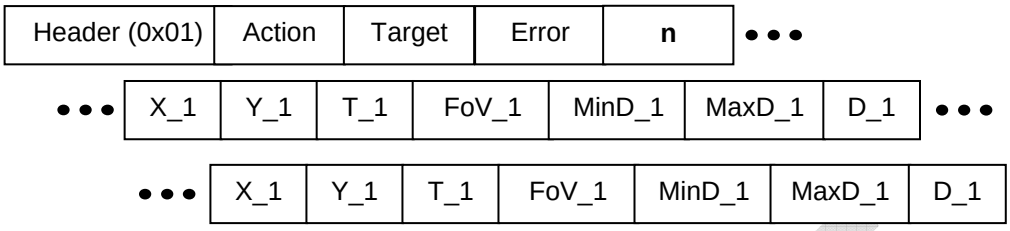
Send a request telegram to receive these properties

Header (0x01)	Action	Target	Error	Nb_AI	Nb_AO	Nb_DI	Nb_DO

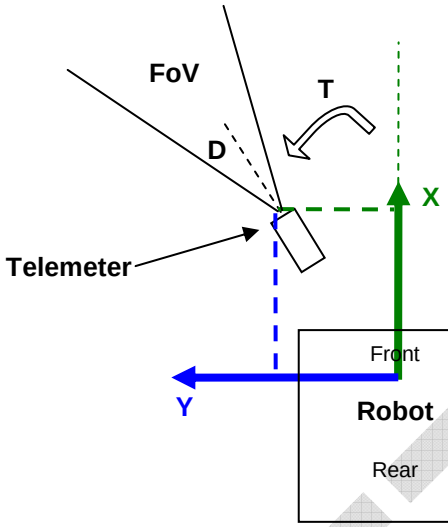
Field	Field size	description
Header	1 byte – type unsigned char	Header of the datagram. Value :1 (0x01)
Action	1 byte – type unsigned char	GET Value : 0 (0x00)
Target	2 bytes – type short	Instance number of the service that sends properties Value : ID of the service that sends its properties (refer to 4.2.1 section to get the services' IDs)
Error	1 byte – type unsigned char	• 0 (0x00) : Ok, no error • Other value : error
Nb_AI	4 bytes – type Int32	Number of analog inputs of the service
Nb_AO	4 bytes – type Int32	Number of analog outputs of the service
Nb_DI	4 bytes – type Int32	Number of digital inputs of the service
Nb_DO	4 bytes – type Int32	Number of digital outputs of the service

4.2.2.5 Telemeter properties telegram

Send a request telegram to receive these properties



Field	Field size	description
Header	1 byte – type unsigned char	Header of the datagram. Value :1 (0x01)
Action	1 byte – type unsigned char	GET Value : 0 (0x00)
Target	2 bytes – type short	Instance number of the service that sends properties Value : ID of the service that sends its properties (refer to 4.2.1 section to get the services' IDs)
Error	1 byte – type unsigned char	• 0 (0x00) : Ok, no error • Other value : error
n	4 bytes – type Int32	Number of telemeters managed by this service
X_1	4 bytes – type Float32	X coordinate of the first sensor, in meters in the robot frame
Y_1	4 bytes – type Float32	Y coordinate of the first sensor, in meters in the robot frame
T_1	4 bytes – type Float32	Orientation of the first sensor, in radians in the robot frame
FoV_1	4 bytes – type Float32	Field of view of the first sensor, in radians
MinD_1	4 bytes – type Float32	Minimum distance that can be measured by the first sensor, in meters
MaxD_1	4 bytes – type Float32	Maximum distance that can be measured by the first sensor, in meters
D_1	4 bytes – type Float32	Actual distance measured by the first sensor, in meters
…	**…**	**…**
X_**n**	4 bytes – type Float32	X coordinate of the **n**th sensor, in meters in the robot frame
Y_ **n**	4 bytes – type Float32	Y coordinate of the **n**th sensor, in meters in the robot frame
T_ **n**	4 bytes – type Float32	Orientation of the **n**th sensor, in radians in the robot frame
FoV_ **n**	4 bytes – type Float32	Field of view of the **n**th sensor, in radians
MinD_ **n**	4 bytes – type Float32	Minimum distance that can be measured by the **n**th sensor, in meters
MaxD_ **n**	4 bytes – type Float32	Maximum distance that can be measured by the **n**th sensor, in meters
D_ **n**	4 bytes – type Float32	Actual distance measured by the **n**th sensor, in meters



4.2.2.6 Battery properties telegram

Send a request telegram to receive these properties

Header (0x01)	Action	Target	Error	NV	P	C

Field	Field size	description
Header	1 byte – type unsigned char	Header of the datagram. Value :1 (0x01)
Action	1 byte – type unsigned char	GET Value : 0 (0x00)
Target	2 bytes – type short	Instance number of the service that sends properties Value : ID of the service that sends its properties (refer to 4.2.1 section to get the services' IDs)
Error	1 byte – type unsigned char	• 0 (0x00) : Ok, no error • Other value : error
NV	4 bytes – type Float32	Nominal voltage of the battery, in Volts
P	4 bytes – type Float32	Energy of the battery when fully charged, in Amperes. hour
C	1 byte – UInt8	Minimum percentage of power at which the battery can operate

4.2.2.7 Localization properties telegram

Due to its architecture (differential 4-wheel robot), the robuROC4 undergoes strong slides. Although the odometry data are provided, they are highly inaccurate (especially when the robot turns) because of the robots' slides.

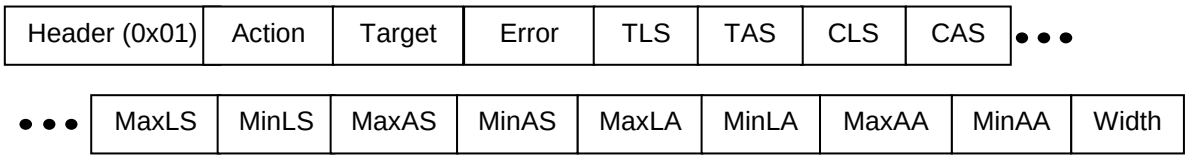
Send a request telegram to receive these properties

Header (0x01)	Action	Target	Error	X	Y	T	State

Field	Field size	description
Header	1 byte – type unsigned char	Header of the datagram. Value :1 (0x01)
Action	1 byte – type unsigned char	GET Value : 0 (0x00)
Target	2 bytes – type short	Instance number of the service that sends properties Value : ID of the service that sends its properties (refer to 4.2.1 section to get the services' IDs)
Error	1 byte – type unsigned char	• 0 (0x00) : Ok, no error • Other value : error
X	8 bytes – type Float64	Initial X coordinate of the Robot in meters in the main frame
Y	8 bytes – type Float64	Initial Y coordinate of the Robot in meters in the main frame
T	8 bytes – type Float64	Initial orientation of the Robot in radians in the main frame
State	4 bytes – type UInt32	Set of flags indicating the state of the localization system Value is composed with this bits : • Bit_0 : "Invalid" : Indicates the localization is not valid • Bit_1 : "Metric" : Indicates that the solution has a metric accuracy • Bit_2 : "Decimetric" : Indicates that the solution has a decimetric accuracy • Bit_3 : "Centimetric" : Indicates that the solution has a centimetric accuracy • Bit_4 : "ProprioceptiveInput" : ndicates that the solution was computed using proprioceptive sensors • Bit_5 : "ExteroceptiveInput" : Indicates that the solution was computed using exteroceptive sensors • Bit_6 : "Error" : Indicates that the solution provider has encountered a fatal error

4.2.2.8 Differential properties telegram

Send a request telegram to receive these properties



Field	Field size	description
Header	1 byte – type unsigned char	Header of the datagram. Value :1 (0x01)
Action	1 byte – type unsigned char	GET Value : 0 (0x00)
Target	2 bytes – type short	Instance number of the service that sends properties Value : ID of the service that sends its properties (refer to 4.2.1 section to get the services' IDs)
Error	1 byte – type unsigned char	• 0 (0x00) : Ok, no error • Other value : error
TLS	4 bytes – type Float32	Target linear speed, in meters/second
TAS	4 bytes – type Float32	Target angular speed, in radians/second
CLS	4 bytes – type Float32	Current linear speed, in meters/second
CAS	4 bytes – type Float32	Current angular speed, in radians/second
MaxLS	4 bytes – type Float32	Maximum linear speed, in meters/second
MinLS	4 bytes – type Float32	Minimum linear speed, in meters/second
MaxAS	4 bytes – type Float32	Maximum angular speed, in radians/second
MinAS	4 bytes – type Float32	Minimum angular speed, in radians/second
MaxLA	4 bytes – type Float32	Maximum linear acceleration, in meters/second²
MinLA	4 bytes – type Float32	Minimum linear acceleration, in meters/second²
MaxAA	4 bytes – type Float32	Maximum angular acceleration, in radians/second²
MinAA	4 bytes – type Float32	Minimum angular acceleration, in radians/second²
Width	4 bytes – type Float32	Distance between the left and right wheels, in meters

4.2.3 Services description

In addition of the services properties, it is possible to receive extra information requesting the service description. Most of the time, when exists, this description contains strings describing the services inputs/outputs.

4.2.3.1 Directory description

This service lists other services. The number of services can be known requesting the properties of the directory service.

Send the following telegram to request the string description each service recorded in the directory's list

Header (0x01)	Action (0x01)	Service ID	List_ID

Field	Field size	description
Header	1 byte – type unsigned char	Header of the datagram. Value :1 (0x01)
Action	1 byte – type unsigned char	QUERY Value : 1 (0x01)
Service ID	2 bytes – type UInt16	ID of the service that has to send properties (refer to 4.2.1 section to get the services' IDs) Value : 0 (0x0000) for Directory service
List_ID	2 bytes – type UInt16	Index of the element in the directory's list you want to receive the string description Value : from 0 to (**n**-1) (**n** is the number of services, can be received reading the directory's properties)

The answer telegram is composed as follows:

Header (0x01)	Action	Target	Error	String

Field	Field size	description
Header	1 byte – type unsigned char	Header of the datagram. Value :1 (0x01)
Action	1 byte – type unsigned char	QUERY Value : 1 (0x01)
Target	2 bytes – type short	Instance number of the service that sends the description Value : ID of the service that sends its description (refer to 4.2.1 section to get the services' IDs)
Error	1 byte – type unsigned char	• 0 (0x00) : Ok, no error • Other value : error
String	N bytes – type char*	String description of the service The number of characters is the number of characters of the UDP message – 5 (header, + action + target + error)

4.2.3.2 Notification description

No available description for this service

4.2.3.3 IOCard description

IOCard allows using inputs/outputs and is composed of 4 types:

- Analog inputs (type value is 0x00)
- Analog outputs (type value is 0x01)
- Digital inputs (type value is 0x02)
- Digital outputs (type value is 0x03)

The number of elements of each type is in the service's properties.

Send the following telegram to request the string description each element of each type

Header (0x01)	Action (0x01)	Service ID	Type	Index

Field	Field size	description
Header	1 byte – type unsigned char	Header of the datagram. Value :1 (0x01)
Action	1 byte – type unsigned char	QUERY Value : 1 (0x01)
Service ID	2 bytes – type short	ID of the service that has to send properties (refer to 4.2.1 section to get the services' IDs)
Type	1 byte – type unsigned char	Value indicating the type of element you want to receive the string description Value: • 0x00: Analog inputs • 0x01: Analog outputs • 0x02: Digital inputs • 0x03: Digital outputs
Index	4 bytes – type Int32	Index of the element you want to receive the string description. Value : • 0x00000000 : first element • 0x00000001 : second element • … • Refer to the properties to know the number of available components of each type.

The answer telegram is composed as follows:

Header (0x01)	Action	Target	Error	String

Field	Field size	description
Header	1 byte – type unsigned char	Header of the datagram. Value :1 (0x01)
Action	1 byte – type unsigned char	QUERY Value : 1 (0x01)
Target	2 bytes – type short	Instance number of the service that sends the description Value : ID of the service that sends its description (refer to 4.2.1 section to get the services' IDs)
Error	1 byte – type unsigned char	• 0 (0x00) : Ok, no error • Other value : error
String	N bytes – type char*	String description of the element The number of characters is the number of characters of the UDP message – 5 (header, + action + target + error)

4.2.3.4 Telemeter description

No available description for this service

4.2.3.5 Battery description

No available description for this service

4.2.3.6 Localization description

No available description for this service

4.2.3.7 Differential description

No available description for this service

4.2.4 Services status

It is possible to receive periodically the status of each service. To receive this status, it is mandatory to send a subscribe telegram for each service (only one time, at the startup of your application for example).

It is important to get properties so that status telegrams can be properly decoded

4.2.4.1 Subscribe telegram

When this datagram is sent, status are sent periodically by the corresponding service

Header (0x01)	Action	ID Not.	ID Service	Period

Field	Field size	description
Header	1 byte – type unsigned char	Header of the datagram. Value :1 (0x01)
Action	1 byte – type unsigned char	INSERT Value : 4 (0x04)
ID Not.	2 bytes – type short	Instance number of the notification service Value : 1 (0x01)
ID Service	2 bytes – type short	ID of the service you want to subscribe to receive status notifications (refer to 4.2.1 section to get the services' IDs)
Period	1 byte – type unsigned char	Period of the status sending. (Value is number of low level cycles, that means number of 10ms) Example : 10 (0x0A) ➔ 100ms

4.2.4.2 Directory status

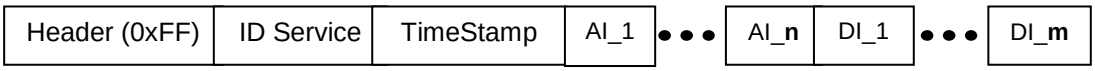
Status is not available for the directory service. A subscribe telegram sent for this service will cause an error telegram as acknowledge

4.2.4.3 Notification status

The notification service is special because it doesn't send its status itself.
Refer the properties of the notification service to receive the services currently recorded

4.2.4.4 IOCard status

The status telegram is the following:



Field	Field size	description
Header	1 byte – type unsigned char	Header of the datagram. Value :255 (0xFF)
ID Service	2 bytes – type short	Instance number of the service that sends the status Value : ID of the service that sends its status (refer to 4.2.1 section to get the services' IDs)
TimeStamp	8 bytes – type UInt64	Notification date. This date corresponds of the number of low level cycle since power on. This low level period is 10 ms
AI_1 to AI_**n** fields	**n** blocs composed of 4 bytes – type Float32	Analog inputs values (generally in volts) Refer to the properties to know the number "**n**" of analog inputs
DI_1 to DI_**m** fields	**m** blocs composed of 1 byte – type unsigned char	Digital inputs values Each bloc contains 8 digital inputs values. Refer to the properties to know the number "nDI" of digital inputs **m** = [(nDI – 1) / 8] + 1

4.2.4.5 Telemeter status

The status telegram is the following:

Field	Field size	description
Header	1 byte – type unsigned char	Header of the datagram. Value :255 (0xFF)
ID Service	2 bytes – type short	Instance number of the service that sends the status Value : ID of the service that sends its status (refer to 4.2.1 section to get the services' IDs)
TimeStamp	8 bytes – type UInt64	Notification date. This date corresponds of the number of low level cycle since power on. This low level period is 10 ms
D_1 to D_**n** fields	**n** blocs composed of 4 bytes – type Float32	Distances values in meters Refer to the properties to know the number "**n**" of telemeters

4.2.4.6 Battery status

The status telegram is the following:

Header (0xFF)	ID Service	TimeStamp	State	Remaining

Field	Field size	description
Header	1 byte – type unsigned char	Header of the datagram. Value :255 (0xFF)
ID Service	2 bytes – type short	Instance number of the service that sends the status Value : ID of the service that sends its status (refer to 4.2.1 section to get the services' IDs)
TimeStamp	8 bytes – type UInt64	Notification date. This date corresponds of the number of low level cycle since power on. This low level period is 10 ms
State	1 byte – type UInt8	Description of the battery state: Value : • 0 (0x00) : "Charging" : The battery is currently being charged • 1 (0x01) : "Charged" : The battery is fully charged, but still plugged in the charger • 2 (0x02) : "Ok" : The battery is operating • 3 (0x03): "Critical": The remaining energy is below the minimum. The battery should be recharged
Remaining	1 byte – type UInt8	Remaining percentage of battery energy

4.2.4.7 Localization status

Due to its architecture (differential 4-wheel robot), the robuROC4 undergoes strong slides. Although the odometry data are provided, they are highly inaccurate (especially when the robot turns) because of the robots' slides.

The status telegram is the following:

Header (0xFF)	ID Service	TimeStamp	X	Y	T	State

Field	Field size	description
Header	1 byte – type unsigned char	Header of the datagram. Value :255 (0xFF)
ID Service	2 bytes – type short	Instance number of the service that sends the status Value : ID of the service that sends its status (refer to 4.2.1 section to get the services' IDs)
TimeStamp	8 bytes – type UInt64	Notification date. This date corresponds of the number of low level cycle since power on. This low level period is 10 ms
X	8 bytes – type Float64	X coordinate of the Robot in meters in the main frame
Y	8 bytes – type Float64	Y coordinate of the Robot in meters in the main frame
T	8 bytes – type Float64	Orientation of the Robot in radians in the main frame
State	4 bytes – type UInt32	Set of flags indicating the state of the localization system Value is composed with this bits : • Bit_0 : "Invalid" : Indicates the localization is not valid • Bit_1 : "Metric" : Indicates that the solution has a metric accuracy • Bit_2 : "Decimetric" : Indicates that the solution has a decimetric accuracy • Bit_3 : "Centimetric" : Indicates that the solution has a centimetric accuracy • Bit_4 : "ProprioceptiveInput" : ndicates that the solution was computed using proprioceptive sensors • Bit_5 : "ExteroceptiveInput" : Indicates that the solution was computed using exteroceptive sensors • Bit_6 : "Error" : Indicates that the solution provider has encountered a fatal error

4.2.4.8 Differential status

The status telegram is the following:

Header (0xFF)	ID Service	TimeStamp	TLS	CLS	TAS	CAS

Field	Field size	description
Header	1 byte – type unsigned char	Header of the datagram. Value :255 (0xFF)
ID Service	2 bytes – type short	Instance number of the service that sends the status Value : ID of the service that sends its status (refer to 4.2.1 section to get the services' IDs)
TimeStamp	8 bytes – type UInt64	Notification date. This date corresponds of the number of low level cycle since power on. This low level period is 10 ms
TLS	4 bytes – type Float32	Target linear speed, in meters/second
CLS	4 bytes – type Float32	Current linear speed, in meters/second
TAS	4 bytes – type Float32	Target angular speed, in radians/second
CAS	4 bytes – type Float32	Current angular speed, in radians/second

4.2.5 Services commands

It is possible to send commands to services in order to control the robot.

It is important to get properties so that command telegrams can be properly sent

4.2.5.1 Directory command

No command telegram available for this service

4.2.5.2 IOCard command

The command telegram is the following:

Header (0xFF)	ID Service	AO_1	• • •	AO_**n**	DO_1	• • •	DO_**m**

Field	Field size	description
Header	1 byte – type unsigned char	Header of the datagram. Value :255 (0xFF)
ID Service	2 bytes – type short	Instance number of the service that sends the status Value : ID of the service that sends its status (refer to 4.2.1 section to get the services' IDs)
AO_1 to AO_**n** fields	**n** blocs composed of 4 bytes – type Float32	Analog outputs values (generally in volts) Refer to the properties to know the number "**n**" of analog outputs
DO_1 to DO_**m** fields	**m** blocs composed of 1 byte – type unsigned char	Digital outputs values Each bloc contains 8 digital outputs values. Refer to the properties to know the number "nDO" of digital outputs **m** = [(nDO – 1) / 8] + 1

4.2.5.3 Telemeter command

No available command telegram for this service

4.2.5.4 Battery command

No available command telegram for this service

4.2.5.5 Localization command

No available command telegram for this service

4.2.5.6 Differential command

The command telegram is the following:

Header (0xFF)	ID Service	TLS	TAS

Field	Field size	description
Header	1 byte – type unsigned char	Header of the datagram. Value :255 (0xFF)
ID Service	2 bytes – type short	Instance number of the service that sends the status Value : ID of the service that sends its status (refer to 4.2.1 section to get the services' IDs)
TLS	4 bytes – type Float32	Target linear speed, in meters/second
TAS	4 bytes – type Float32	Target angular speed, in radians/second

A Watchdog is used to secure the communication. If no command is received in a given delay (default value is 1000ms), the vehicle is stopped

It is possible to modify this watchdog value. To do this,

- connect the low level controller
- edit the "**udpserver.cfg**" file
- find and modify the "**Watchdog**" field of the "**DifferentialService**" section
- units are numbers of low level cycles (duration 10ms)

4.2.6 Services replace

It is possible force the state of a some services

It is important to get properties so that telegrams can be properly sent

4.2.6.1 Directory replace

No available replace telegram for this service

4.2.6.2 IOCard replace

No available replace telegram for this service

4.2.6.3 Telemeter replace

No available replace telegram for this service

4.2.6.4 Battery replace

No available replace telegram for this service

4.2.6.5 Localization replace

Due to its architecture (differential 4-wheel robot), the robuROC4 undergoes strong slides. Although the odometry data are provided, they are highly inaccurate (especially when the robot turns) because of the robots' slides.

It is possible to replace the state of this service, for example if you want to correct the localization using another device.

The replace telegram is the following:

Header (0x01)	Action	Target	TimeStamp	X	Y	T

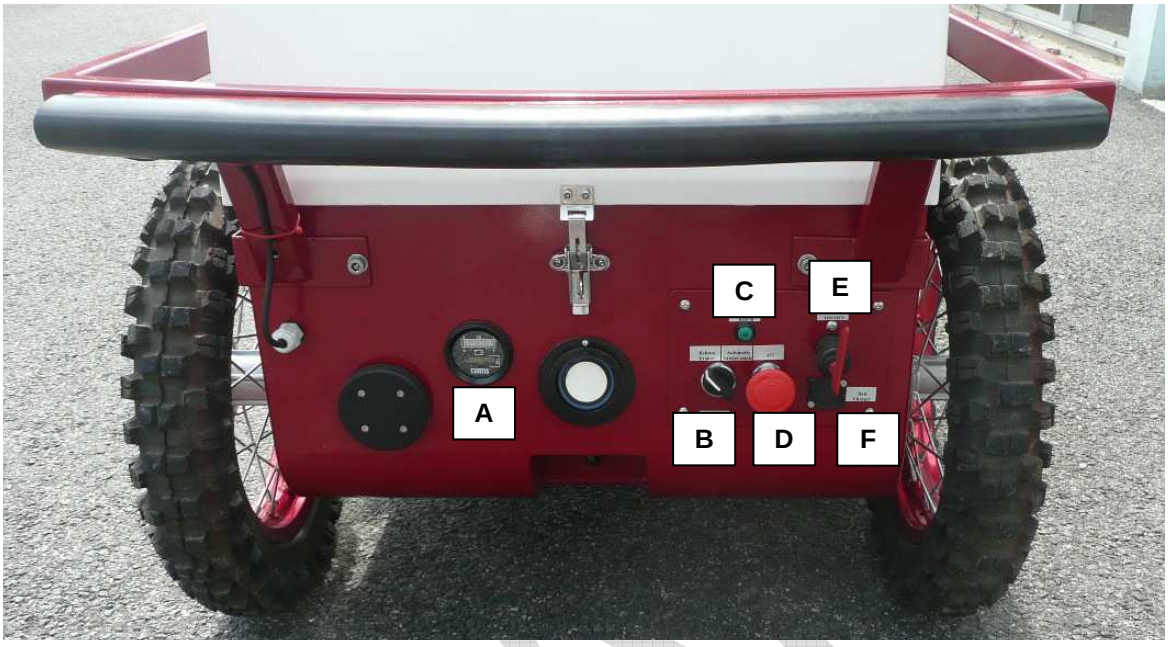
Field	Field size	description
Header	1 byte – type unsigned char	Header of the datagram. Value :1 (0x01)
Action	1 byte – type unsigned char	REPLACE Value : 2 (0x02)
Target	2 bytes – type short	Instance number of the service that will receive the replace telegram (refer to 4.2.1 section to get the services ID of Localization)
TimeStamp	8 bytes – type UInt64	Not used but mandatory
X	8 bytes – type Float64	New X coordinate of the Robot in meters in the main frame
Y	8 bytes – type Float64	New Y coordinate of the Robot in meters in the main frame
T	8 bytes – type Float64	New orientation of the Robot in radians in the main frame

4.2.6.6 Differential replace

No available replace telegram for this service

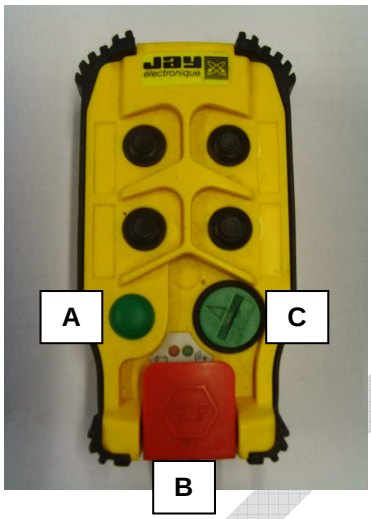
5 Appendix

5.1 Control panel



Identifier	Description
A	Battery level indicator
B	Brakes management selector • Released: used to be able to push the platform. In this position, the motors power cannot be enabled • Automatic: The brakes are managed by the low level software
C	Button used to enable the motor power. To be able to do this, both the local and remote emergency stops must be released
D	Local Emergency stop button. Press it to disable the motor power
E	Main power switch. Use it to turn ON/OFF the entire platform
F	Connector of the battery charger. Turn OFF both the battery charger and the platform before plugging the battery charger

5.2 Remote emergency stop



Identifier	Description
A	This button has 2 functions : • First use: Enable the link between the remote control and the receiver • Other uses: Enabling the motor power (disable first the local and the remote emergency stops). This function is also available on the platform's control panel
B	Brakes management selector • Released: used to be able to push the platform. In this position, the motors power cannot be enabled • Automatic: The brakes are managed by the low level software
C	Button used to enable the motor power. To be able to do this, both the local and remote emergency stops must be released

5.3 Changing the IP settings

The IP settings of the controller can be changed through a command line utility program present on the controller itself. The instructions given here assume that it is done from a Windows computer.

- First step: Open a telnet session.

For this, you need to open a command prompt. ("Start Menu -> Run", then enter "cmd" and click "Ok").

At the prompt, type the following command (It is supposed here the controller IP is 192.168.0.2):

```
telnet 192.168.0.2
```

You should get a welcoming screen from the controller.

- Second step : Use the netcfg.exe utility

In the telnet session, type the following command to change the address to 10.0.0.10 , IP mask to 255.0.0.0, with a gateway address 10.0.0.1:

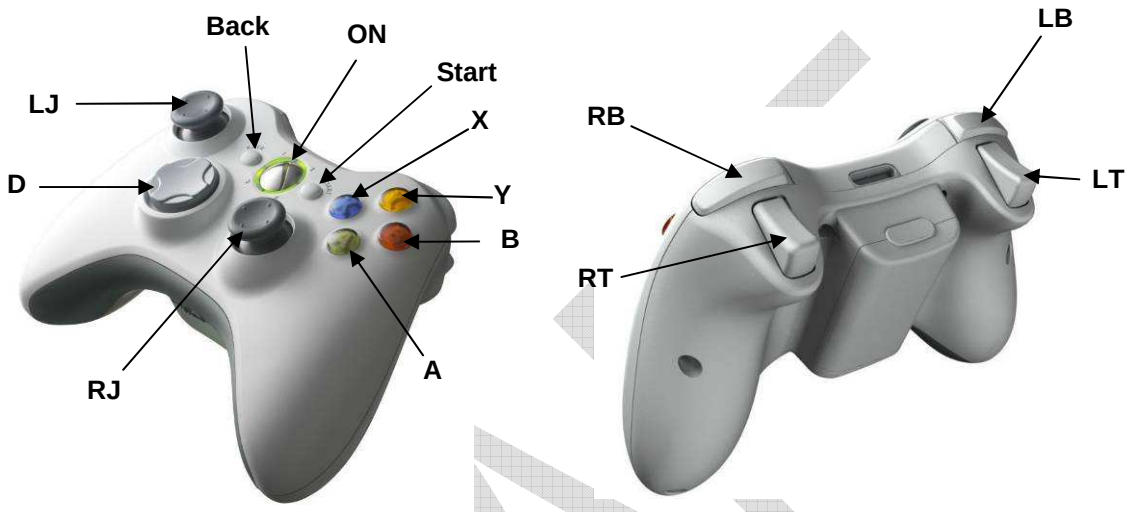
```
netcfg 10.0.0.10  255.0.0.0  10.0.0.1
```

After a few seconds, you will have a confirmation message.

- Final step : reboot the controller

You can now reboot the controller to have the changes take effect.
This can be done by powering off and on the controller

5.4 Game pad buttons



5.5 Links Hardware / Services

The following section describes the software link between the services and the "hardware" to help you using correctly the services

This links are software links. That means it is not representing the real wiring of the robot

5.5.1 IOCard Service

The following table presents all the "Float32" inputs of the service

Analog inputs		
ID	**Description**	**Units**
00	Front telemeter voltage	Volts
01	Rear telemeter voltage	Volts
03	Battery voltage	Volts
04	Game pad "LJ" Left/Right	[-1.0 ; +1.0]
05	Game pad "LJ" Down/Up	[-1.0 ; +1.0]
06	Game pad "RJ" Left/Right	[-1.0 ; +1.0]
07	Game pad "RJ" Down/Up	[-1.0 ; +1.0]
08	Game pad "LT"	[-1.0 ; +1.0]
09	Game pad "RT"	[-1.0 ; +1.0]

The following table presents all the "bool" inputs of the service. It is generally states like "pressed" / "not pressed"

Digital inputs	
ID	**Description**
00	Front bumper
01	Rear bumper
03	-- exists but not used --
04	-- exists but not used --
05	-- exists but not used --
06	-- exists but not used --
07	-- exists but not used --
08	-- exists but not used --
09	Game pad "A" button
10	Game pad "B" button
11	Game pad "X" button
12	Game pad "Y" button
13	Game pad "Back" button
14	Game pad "Start" button
15	Game pad "D" down button
16	Game pad "D" left button
17	Game pad "D" right button
18	Game pad "D" up button
19	Game pad "LJ" click
20	Game pad "RJ" click
21	Game pad "LB" button
22	Game pad "RB" button

The following table presents all the "Float32" outputs of the service

Analog outputs		
ID	**Description**	**Units**
	-- No analog outputs exists for this robot --	

The following table presents all the "bool" outputs of the service. It is generally states like "Enable" / "Disable"

Digital Outputs	
ID	**Description**
00	Watchdog (used by the low level software, cannot be used using the service)
01	-- exists but not used --
03	-- exists but not used --
04	-- exists but not used --
05	-- exists but not used --
06	-- exists but not used --
07	-- exists but not used --

5.5.2 Telemeter Service

The following table presents all the telemeter devices of the service. When receiving properties or status, the order of the devices is the following:

Telemeters	
ID	**Description**
00	Front ultrasonic sensor
01	Rear ultrasonic sensor

5.5.3 Battery Service

The Front left servo drive is used to monitor the battery voltage. This voltage is used by the Battery service

5.5.4 Localization Service

The front left and front right servo drive (encoder feedbacks) are used to compute localization

5.5.5 Differential Service

The 2 left drives are receiving the same command; the 2 right drives are receiving the same command