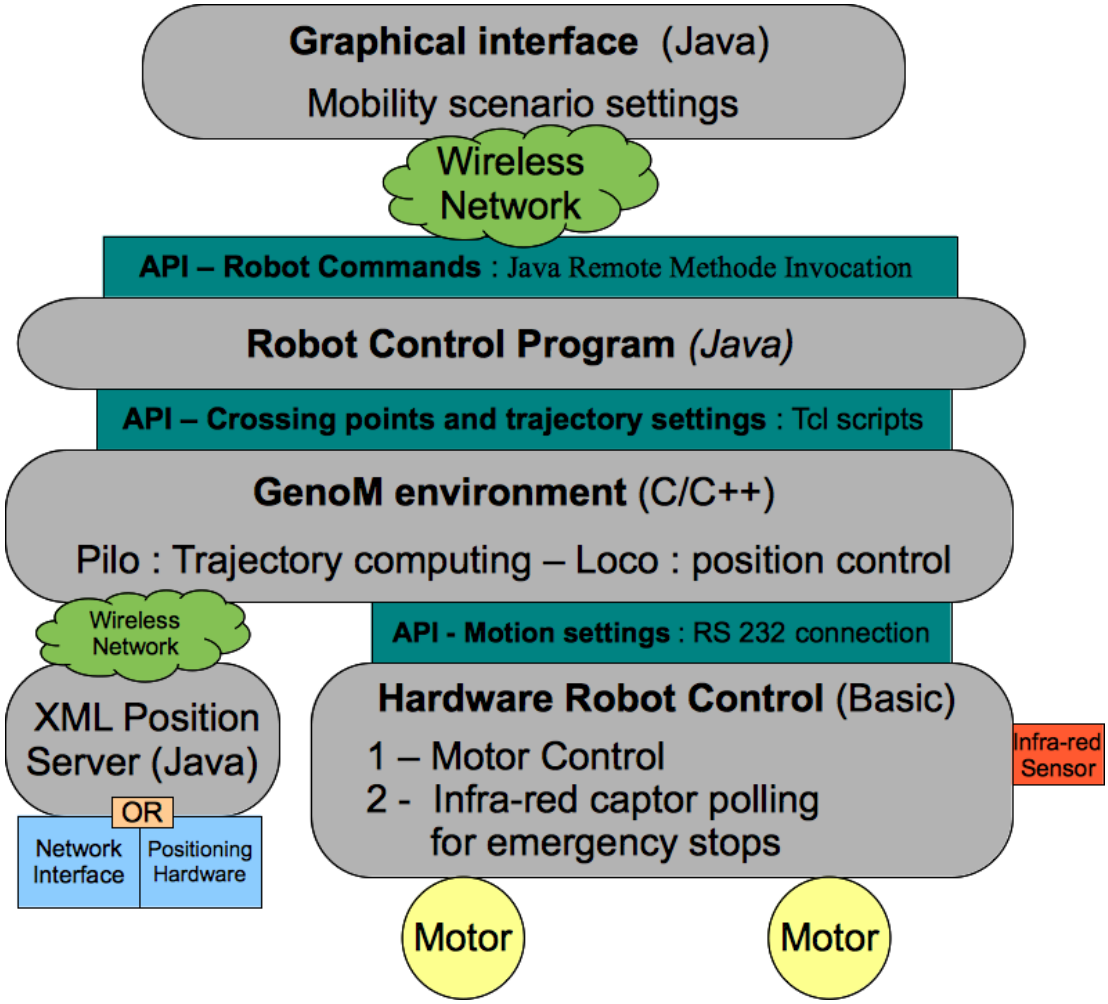
Mobility Platform – Technical Documentation

General Architecture Overview



Index

Mobile Node Hardware

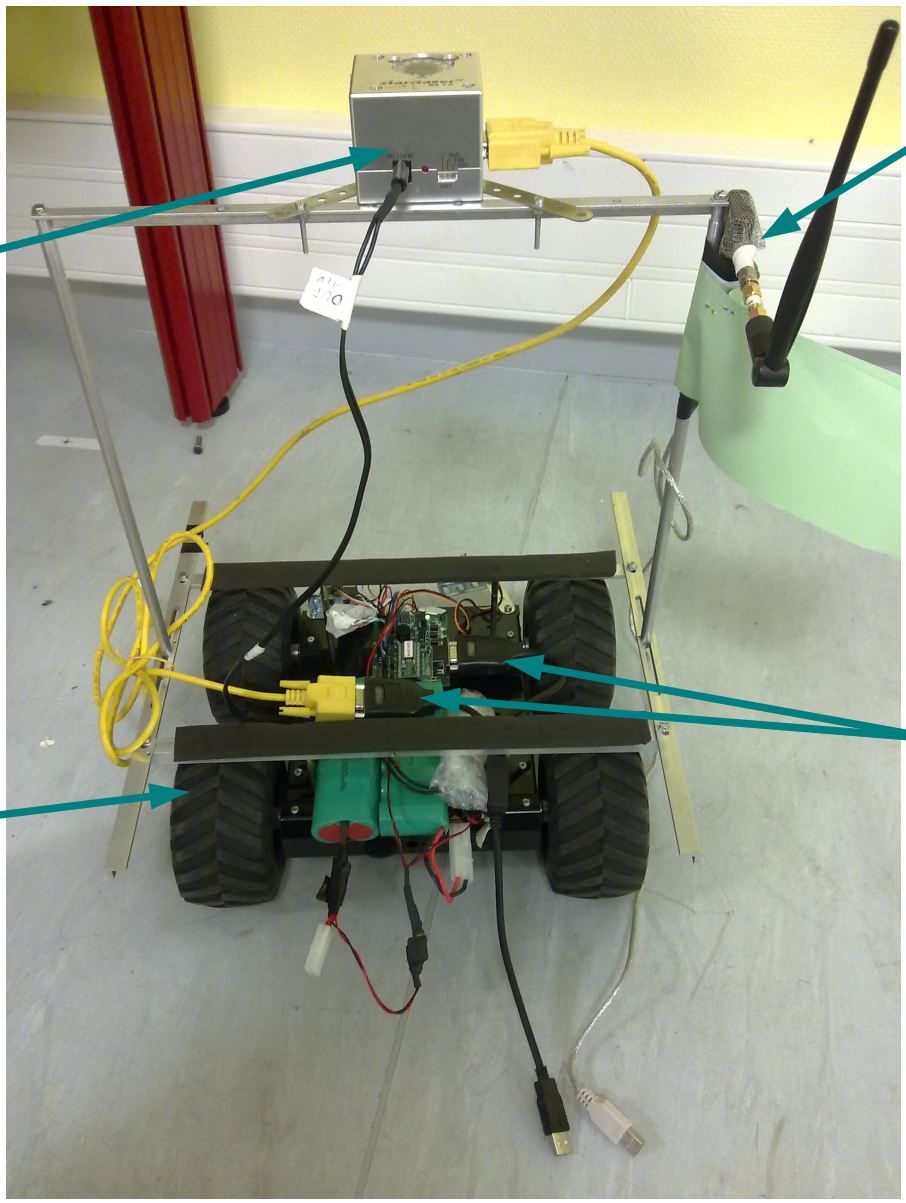


Illustration 1: A full equipped mobile robot

Power Supply

Two batteries of 7,5Volts are need to power supply the robot. One for the motors power card and one for the control card and the Infra-Red camera. A component (TracoPower tmr 3-0512) convert the 7,5V into 12Volts to supply the camera.

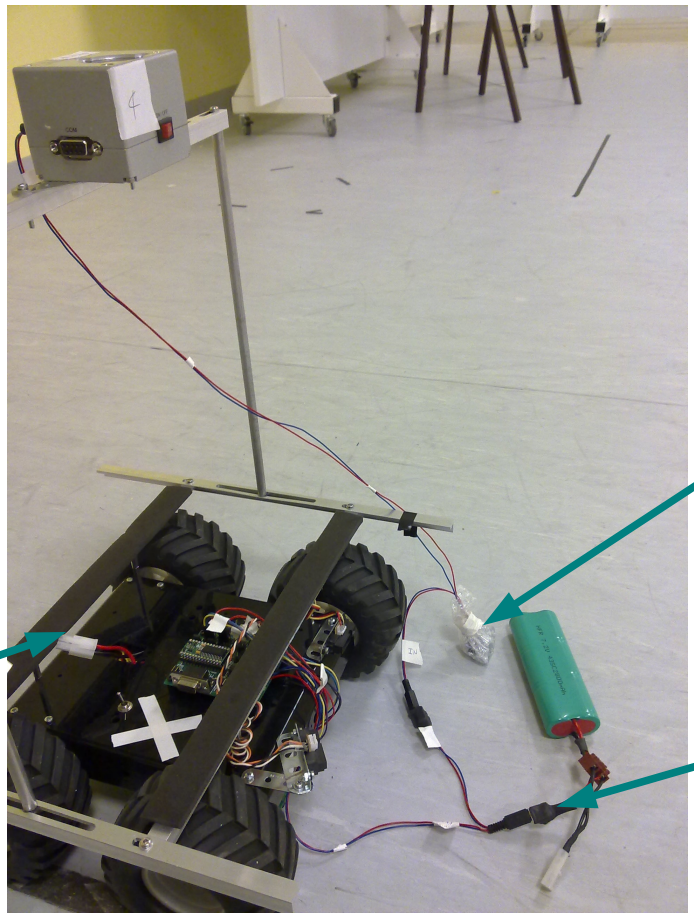
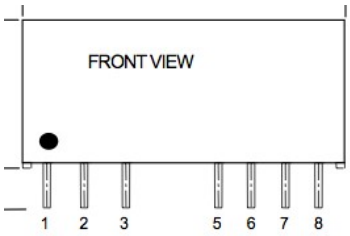


Illustration 2: Mobile robot power supply

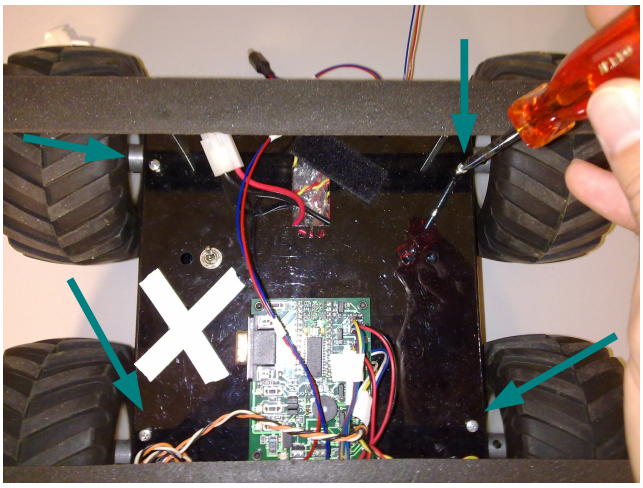
TracoPower tmr 3-0512 Connexions



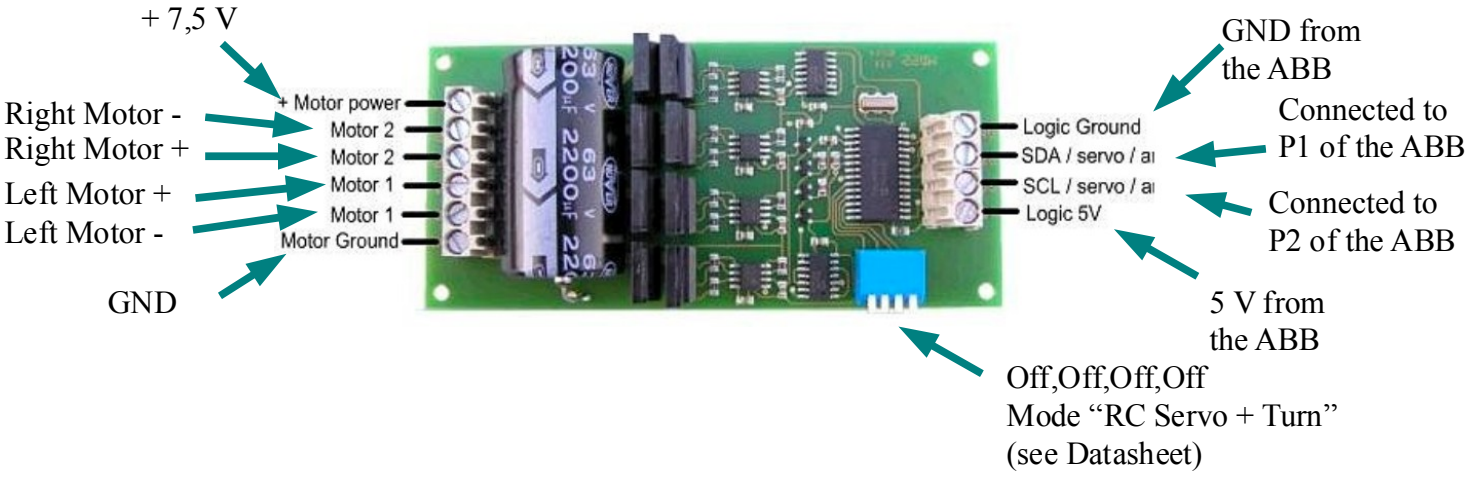
Pin 1 : Input / GND (7,5V)
Pin 2 : Input / + 7,5V
Pin 6 : Output / + 12V
Pin 7 : Output / GND (

Motors Power Card - MD22

To access to the motor control card (MD22 - Dual 24Volt 5Amp H Bridge Motor Drive) remove the top of the mobile robot (4 screws) :

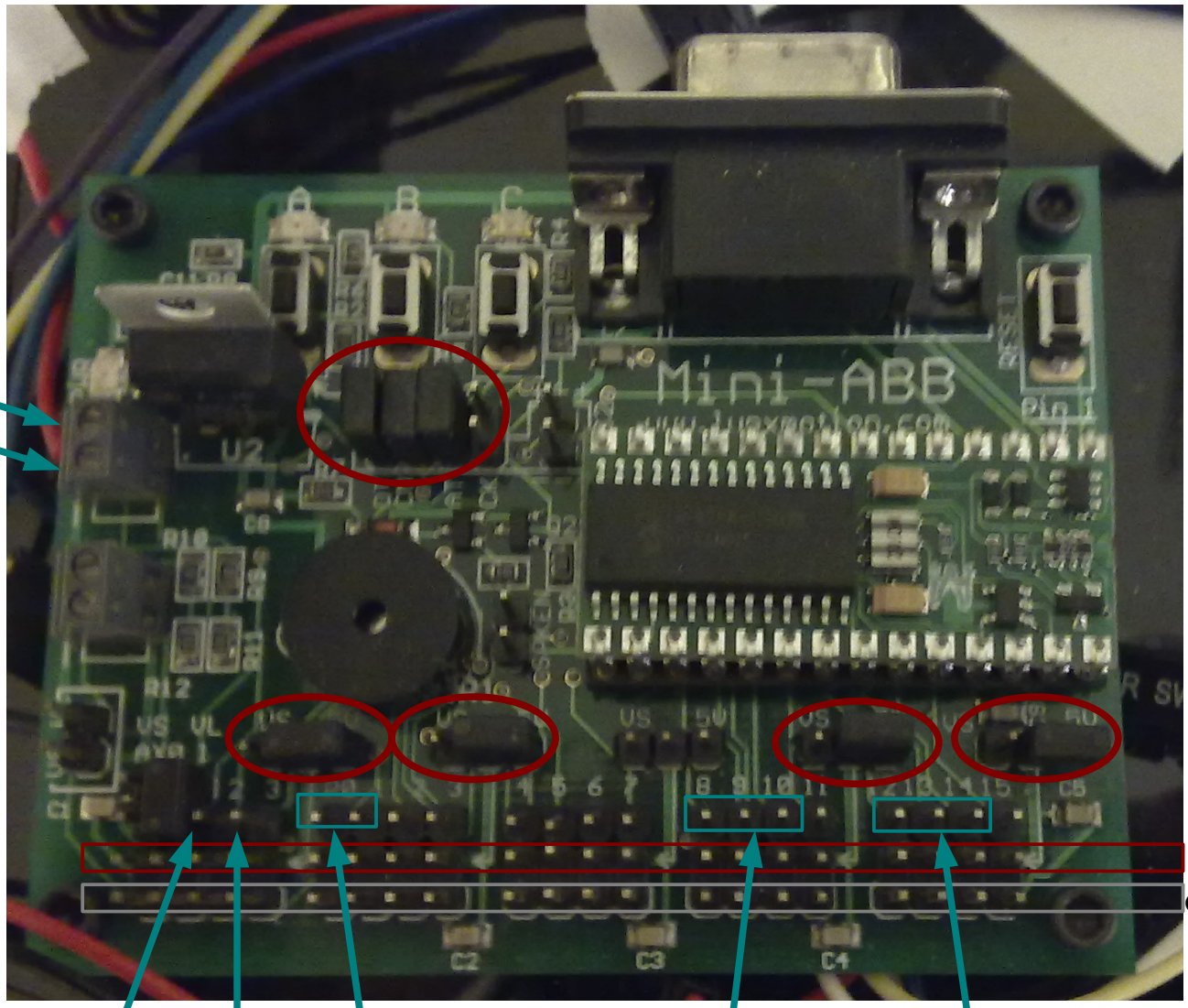


MD22 Connexions

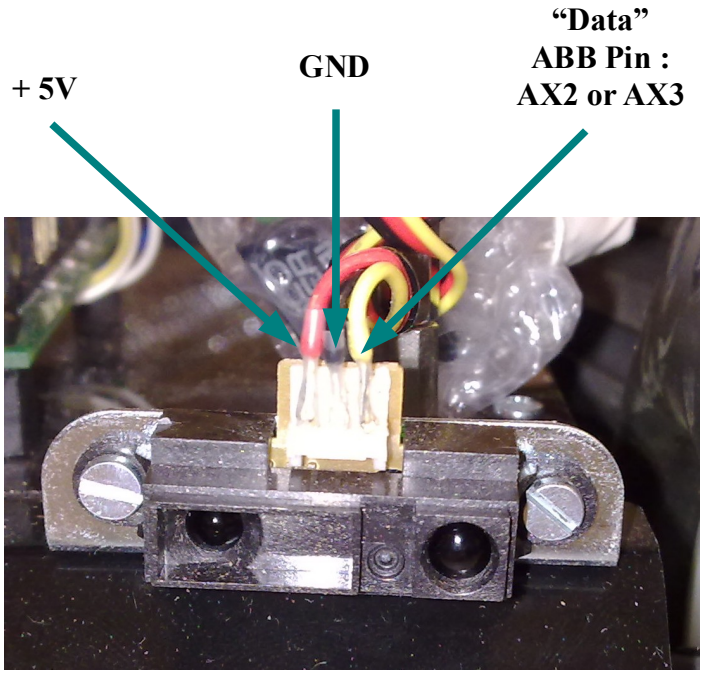


Control Card – Mini ABB

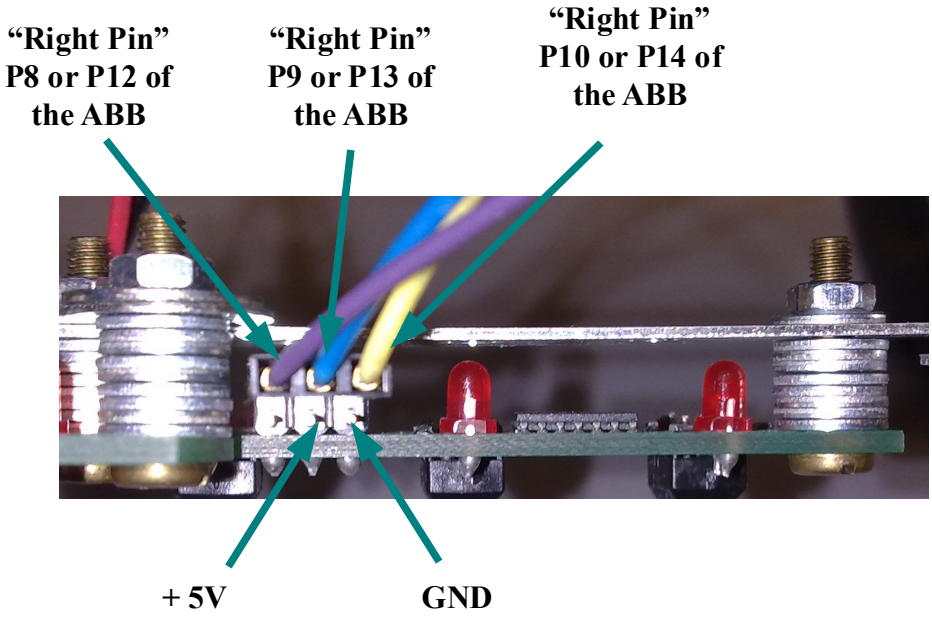
The Mini Atom Bot Board is on the top of the robot. Notice the position of the switches : , they enable the use of the puss buttons A,B,C, and select the + 5 V output in the middle line of the bottom Pins.



Infra-Red Proximity Sensor - Sharp GP2D12



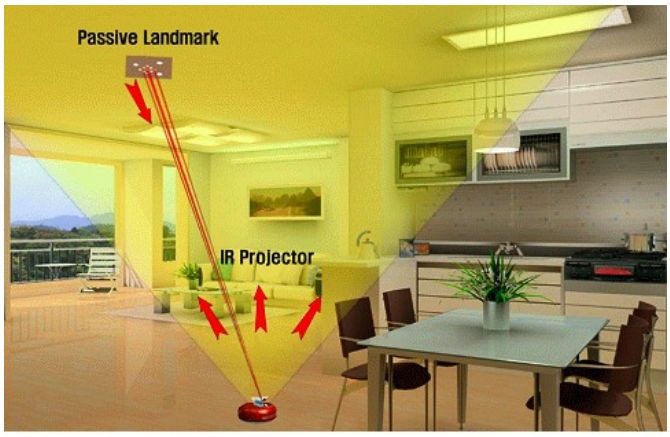
Infra-Red Line Tracker

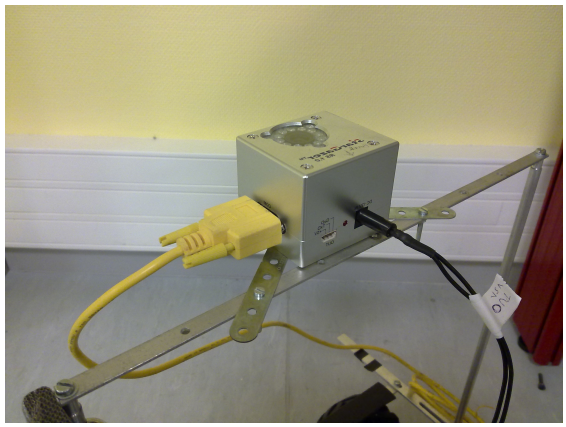


Positioning system

Infra-Red Camera - Hagisonic Stars Gazer

The camera uses passive landmark positioned on the celling. It is power supplied by a 12V tension and communicate through a serial communication (DB9 socket). Once calibrate and powered you just have to send the string "~#CalcStart`" the camera to start the positioning, then the camera will continuously send his position with messages like : “~^I512|+76.35|+81.46|+181.15|528.24”. Where the first number is the ID of the landmark seen, the second is the orientation of the camera followed by the X,Y and Z position.





Calibration – Building a map with a Hagisonic Star Gazer RS 1.0

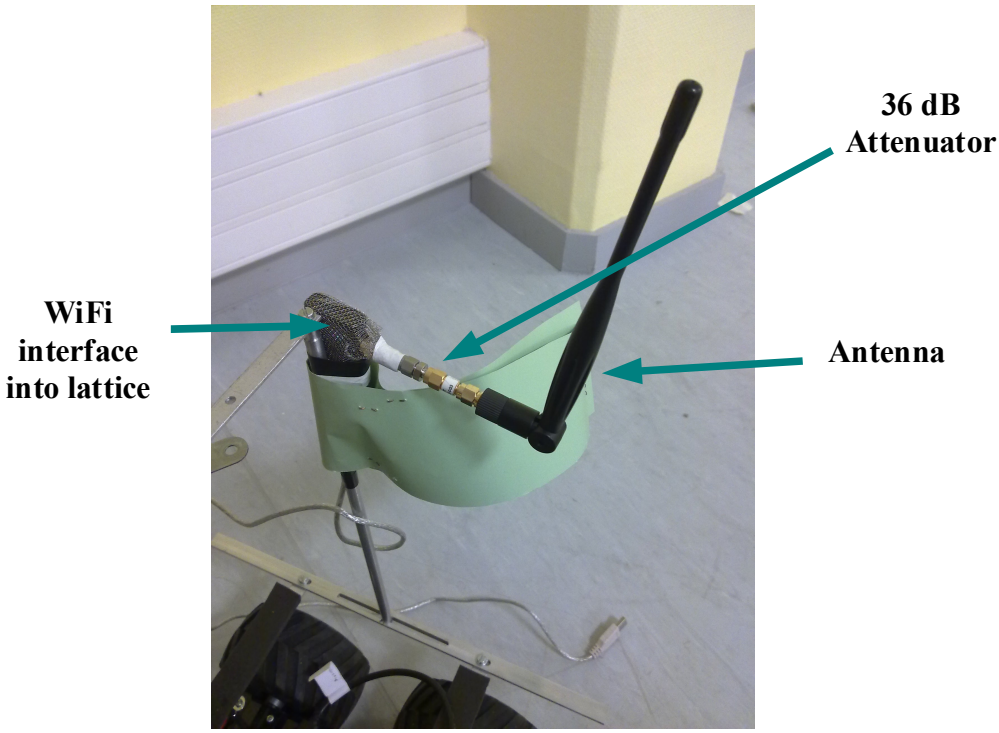
Fix the landmarks on the celling to cover all the room surface. Be careful to respect same the orientation for each tag (indicated by an small blue arrow) and do not place them to far from each other (2.5 to 3m maximum for ceiling height 2.4m).
Position the camera under the reference tag (x=0,y=0).
In Eclipse software, open the project “Mobility Platform” and select the class “IRCom2.java” in the package “hagisonicLocationService”.
Set the right serial port name (where the camera is connected).
Run the main.
Power on the cam.
Click on "Creation de la carte", set the number of tags (for us, now, 24) and the ID of the reference tag (here “96”).
Click on "suite" and place the camera some seconds under each tag until the tag list is completed.
Note : you should immediately see the position of the camera updating in the console, starting with the letter "F", if not restart the calibration procedure.
When all the tags have been detected you can stop software and start to use the camera for positioning.

Ubisense UWB

Building ...

Attenuated WiFi interface

We use “Edimax ew-7318usg” WiFi adaptors. To reduce the signal range an attenuator of 36dB is inserted between the WiFi interfaces and its antenna. And to avoid noise and signal leakage from the electronic circuits the WiFi interface is place into a Faraday lattice.



Mobile Node Software

SVN repository to get the sources

svn+ssh://gershwin.laas.fr/local/svn/hidenets/trunk/MobilityPlatform

Robot Control - Basic program

Two programs was developed in basic to control the robot : “overTakeSerial.bas” to make the robots follow black lines on the ground and “genomListener.bas” to control the robots in linear and angular speed. To compile them and program the robots ABB control card you must use the Basic Micro IDE software under Windows.

How to use “overTakeSerial.bas” :

With this program the robot follows a simple line with 2 Line trackers (3 IR captors each). Our standard scenario allow two robots to move on the circuit alternating overtaking (at a junction the one which go to the right overtake the one which go to the left).

Serial control Mode

To initialize a robot, you have to identify the first mobile sending a "M" and a "0" character and the second sending a "M" and a "1" character, on the ABB serial port.
Then each robot will “ask” periodically for directives, sending "Q" on the serial port. If it has, to stop it must receive "S" to start, “G” to continue or “C” to pass in crash mode (it then will go straight until it detects an obstacle).
Note : After the Init you can also set the delay between two serial port "asks" sending the characters "F" & "hundreds" & "tens" & "units" (default value 100, maximum value 255).

Autonomous Mode

This mode does not need serial communication to run the scenario, the robots will autonomously follow the track alternating overtaking endless.
To start this mode, push the "B" button while resetting the ABB, then identify the initial robot position : if it is the first mobile (it will be overtaken) push "A", if it is the second (it will overtake) push "C". Finally press "B" to validate you choice, the robot will start in 5 seconds.

How to use “genomListener.bas” :

In this mode the robot simply executes the control command sends to it on the ABB serial port. The commands control the linear speed and the angular speed and must be send character by character.
Send "I" to initialise the robot.
Send "G" or "B" to specify the linear speed direction (Go or Back) followed by the speed value (in two char) in centimetres/second unit.
Send "T" or "A" to specify the angular speed direction (Trigonometric or Ante-trigonometric) followed by the angular speed value (in two char) in deci-radians/second unit.

Typically, the GenoM motion control module “leloco” or the Java comComRemoteServer program sends those commands.

GenoM Environment – C program

How to install GenoM and modules needed:

Install robot Package (instructions here : http://homepages.laas.fr/mallet/robotpkg).
Then use it to install the modules (with "make update"command): architecture/genom
localization/pom
motion/contlaw
Dependency management takes care of the details (thanks to Anthony !).
Note : we keep the default install directory : ***" /opt/openrobots/".***

Copy the modified modules folders "pilo" et "leloco" in "/opt/openrobots/modules"
In each folder type :

"make regen" to regenerate the module code with the last version Genom (if needed)
"./configure –prefix=/opt/openrobots" to prepare the make file
"make"
"make install"

If needed set the environment variable PKG_CONFIG_PATH : ***"export PKG_CONFIG_PATH = /opt/openrobots/lib/pkgconfig/"***

Pilo Module :

"Pilo" is in charge to calculate the the trajectory from defined passage points. There is no modification to do in this module.

Leloco Module :

"Leloco" controls the robot linear and angular speeds. It uses the robot position, get from the XML position server, and compares it to the trajectory calculated by "pilo" to generate the appropriate speed control command and send it to the robot, through the serial communication.
If you need to modify interaction with the robot or the position server, edit the C file "codel/lelocoMotionTaskCodels.c".
If you need to add external library edit the file "codel/Makefile.in".
If you need to modify the PID control values or other robot constants (speed max, acceleration max, motors axis width, ..) edit the file "lelocoConst.h".
Don't forget to compile again "pilo" and "loco" after modifications.

Run the modules :

Usually all this part is done by the Java "ComGenomServer.java" class. The different steps are described here.

To be able to run the modules you need to start the "pocolib" environment :
".../openrobots/bin/h2 init 2000000"
Then start the tcl server, to can communicate with the modules :
".../openrobots/bin/tclserv -c -p 9472"
Finally you can start the modules :
".../openrobots/bin/leloco"
".../openrobots/bin/pilo"

Note :
To stop the pocolib environment at the end of an experiment use :
"../openrobots/bin/h2 end"
If modules freeze or didn't finish properly you may need to kill them :
"killall tclserv picoweb pilo"
Then you will have to remove those file manually :
"rm $HOME/.leloco.pid*"
"rm $HOME/.pilo.pid*"

Communicate with the modules :

Initialisation of the tcl server connexion

To communicate with the modules you must use the tcl server run previously. Create a socket with it on the specified port (here 9472).
Then follow this handshake protocol :

Send ***"HELLO***" to the server. You will receive a message with the id of the socket.

Send "***REPLYTO "*** followed by the socket ID to ask the server to answer on the same socket. You will receive an acknowledgement.

Send ***"LM leloco"*** to load the "leloco" module. You will receive an acknowledgement.

Send ***"LM pilo"*** to load the "pilo" module. You will receive an acknowledgement.

Send ***"cs::init"*** to creat the mail box used to communicate. You will receive an acknowledgement.

Send ***"modules::connect"*** to start the connexion with the modules loaded. You will receive an acknowledgement.

The connexion is now Ready and you can start to communicate with the modules

Initialisation of the modules

Init "leloco" sending :
"leloco::init posServerName posServerPort genomClientName robotIdToTrack robotPortComm"

Where : posServerName = The name of position server host.
rosServerPort = Port used by the position server.
genomClientName = Name chosen for the GenoM Client. It will be send as an ID to the position server.
robotIdToTrack = ID of the mobile to track to know our position. Which is ask to the position server.
robotPortComm = Name of the serial port choose to communicate with the robot.

You will receive an acknowledgement.

Init the "pilo" module sending :
"pilo::Init lelocoMEPos" this specify the name poster used by the two modules to communicate.
You will receive an acknowledgement.

Start the position control sending :
"leloco::Track piloPosRef"

Send movement orders

Now, to control the robot you just have to send the appropriate movement orders to the "pilo" module. The "leloco" module will automatically follow the consign generate by "pilo" and send the appropriate speed command to the robot, depending on the position get from the position server. You can send those commands to "pilo" :

"pilo::move dist 80 80" to make the robot move straight. "dist" is the distance in meter. The following numbers represent the percent of the maximum speed and maximum acceleration permit to execute the movement.

"pilo::turn ang 0 80 80" to make the robot rotate of "ang" radians.

"pilo::GotoConfig PILO_GOTO_CONFIG PILO_ABSOLUTE_TRAJ 80 80 x y theta" to make the robot go to the position "x, y" with the "theta" orientation. The two first numbers represent the percent of the maximum speed and maximum acceleration permit to execute the movement.

To execute complex trajectory you should use ***"pilo::ExecTraj"*** command use with the following arguments :

A number representing the ID of the trajectory.
The number of segment in this trajectory.
PILO_LINE_SEG to describe a broken line trajectory (succession of segments) or PILO_ARC_SEG to describe a succession of couple segment/arc.
PILO_ABSOLUTE_TRAJ if you want to start from a given absolute position or PILO_RELATIVE_TRAJ if you want to start from the current robot position.
X0, the initial position (if absolute trajectory chosen).
Y0, the initial position (if absolute trajectory chosen).
Theta0, the initial position (if absolute trajectory chosen).
The maximum initial robot angular deviation (real position compare to initial position define) allowed to start the trajectory (if absolute trajectory chosen).
A suite of 50 (maximum possible) segment description. This description is composed of 3 numbers : an angle in radiant (rotation need from the previous robot orientation to got to the desired segment orientation), the distance of the segment in meter and the maximum difference of position from the segment (in meter) allowed for smoothing the path of the robot.
Note : You need to type all the 50 segments descriptions even if there are not used.

A command example :
"pilo::ExecTraj 1 5 PILO_LINE_SEG PILO_ABSOLUTE_TRAJ 1 0 2 1.5 1 0 1 0.5 0.74 2 0.5 0.77 2.33 0.5 1 1.5 0.5 1.6 4.6 0.5 0"

Stop the demo

You can stop the robot sending the consign :
"pilo::stop"
You can stop the tracking sending :
"leloco::stop"
To stop the modules send :
"KILL leloco abort"
"KILL pilo abort"

To stop the tcl server send :
"DIES"
You will receive acknowledgement for all those command, then you can close the socket.

XML Position Server – Java program

The position server is in charge to receive the position of different the mobiles, and to distribute it to the clients which need them (supervision, GenoM modules, …). Whatever the kind of client or positioning system used, the XML messages send and receive by the position server have always the same architecture. To Keep a standard position interface.

The position server is include in the "positionServer" Java package and you can use the ant file to run it.

XML messages description

InitMessage : a positioning system client (position writer) → Position Server
(in this example the new client "Gnocchi" specify that it will give a position calculated with the positioning system UWB)

```
<?xml version="1.0" encoding="UTF-8"?>
<message id="1" type="init" time="10:15:06.422+02:00" date="2010-04-14+02:00">

    <client name="Gnocchi" type="writer" positionService="UWB">
    <mobilesTracked/>

</message>
```

InitMessage : supervision/Genom client (position reader) → Position Server
(the new "Supervisor" client specify that it is waiting for positions of "Robot1" and "Robot2", or "All for the position of all the modules)

```
<?xml version="1.0" encoding="UTF-8"?>
<message id="1" type="init" time="10:15:06.422+02:00" date="2010-04-14+02:00">

    <client name="Supervisor" type="reader">
    <mobilesTracked>
        <mobile id="Robot1"/>
        <mobile id="Robot2"/>
        <mobile id="All"/>
    <mobilesTracked/>

</message>
```

Position Sending Message : Positioning system (position writer) → Position Server
(The client send positions of a mobile to the server)

```
<?xml version="1.0" encoding="UTF-8"?>
<message id="1" type="setPosition" time="10:15:06.422+02:00" date="2010-04-14+02:00">

    <mobile id="Robot1" posX="123.45" posY="45.12" posZ="0" cap="195" time="10:15:06.422+02:00"
                                                                    date="2010-04-14+02:00"/>
    <mobile id="Robot2" posX="13.45" posY="5.12" posZ="0" cap="5" time="10:15:06.422+02:00"
                                                                    date="2010-04-14+02:00"/>

</message>
```

Position Sending Message : Position Server → Supervision/GenoM client (position reader)
(the server send position information to the client)

```
<?xml version="1.0" encoding="UTF-8"?>
<message id="1" type="givePosition" time="10:15:06.422+02:00" date="2010-04-14+02:00">

    <mobile id="Robot1" posX="123.45" posY="45.12" posZ="0" cap="195" time="10:15:06.422+02:00"
                                                                    date="2010-04-14+02:00/>
    <mobile id="Robot1" posX="13.45" posY="5.12" posZ="0" cap="5" time="10:15:06.422+02:00"
                                                                    date="2010-04-14+02:00/>
</message>
```

RobotControl for GenoM – Java

To control the robot with GenoM modules you should run the “comGenomServer.java” class on the embedded laptop. The program receives the “Route” object and movement instructions via JAVA-RMI from the GUI and convert them in movement command to send to the GenoM modules via the tcl server. The GenoM modulse will then communicate the speeds orders to the robot control card, (which is running “genomListener.bas”) via the serial port.

Before run the class you have to define the constants :
host = host where the tcl server is running.
port = port of the tcl server.
Then informations to send to the Genom modules during the initialisation.
posServerName = host of the position server.
posServerPort = port of the position server.
genomClientName = name of the GenoM service as it will be saved by the position server.
robotIdToTrack = name of the robot to track.
robotPortComm = name of the serial port to communicate with the robots.

RobotControl for GUI Remote Control - Java

To control the robot with the GUI Remote Interface you need to run the “ComComRemoteServer.java” class on the embedded laptop. The program will receive the speeds orders from the GUI, via JAVA-RMI, and communicate them to the robot's ABB card, running “genomListener.bas” program, via the serial port.

Before run the program you just have to set the String constant “port” with the name of the serial port where the robot is connected.

Run a Demo

Load the batteries

Use the loaders in 1000mA mode, select “6 cells” to load the standard batteries and “5 cells” to load the smaller one (for the yellow robots). Load a battery take between 2 and 4 hours.

Run a demo

All the connexion variable (IP, name, …) are define in the property.properties files.
The robots should be programmed with the GenomListener basic code.

- Start the Air Port access point, connect the computer “gnocchi” to it (Gnocchi IP should be 10.0.1.10)
- Start the “PositionServer on gnocchi (MobilityPlatform/positionServer/ServerPos.java or use tha ant file). And the map designer (MobilityPlatform/MapDesigner/MapDesignerLauncher.java or use the ant file)

- Start the mac-books, log in the session “Arum”. Activate Air port and control that the wireless network selected is “hidenets”.
- Connect two batteries to each robots. One to the motors (white socket), and one the equipment (black “Jack” socket).
- Put the motor power switch in position ON (forward of the black robot and backward of the yellow robot).

Connect the USB/COM adepter to the robots (the longer COM cable is connected to the control card, the shorter to the IR camera) and to the macbook (usually the farthest from the screen USB socket)

- On the mac-books start the the position client (MobilityPlatform/locationService/hagisonic/HagisonicPositioning.java or use the ant file), then you can start the hagisonic camera (NOT BEFORE or you will full the COM buffer).
- On the mac-books start the the Genom RMI server (MobilityPlatform/comGenom/ComGenomServer.java or use the ant file). TO USE THE REMOTE CONTROL MODE (mapDesigner remote control panel joystick) : you should run the comCom RMI server (MobilityPlatform/comCom/ComComRemoteServer.java or use the ant file)

The Platform is now ready, you ca use the mapDesigner to control it. See next chapter for more details.

Troubleshooting

The robot don't move / stop moving

Change the batteries - Check the COM connexions – Check the motor power switch – From the mapDesigner Stop and reSend the route – Check there is no obstacles in front of the robot.

Note : If the robot control card receive the correct init from the macbook the three leds shoul be on (green, orange, red).

The robot go always bacward and seem drunk (yes it happens !)

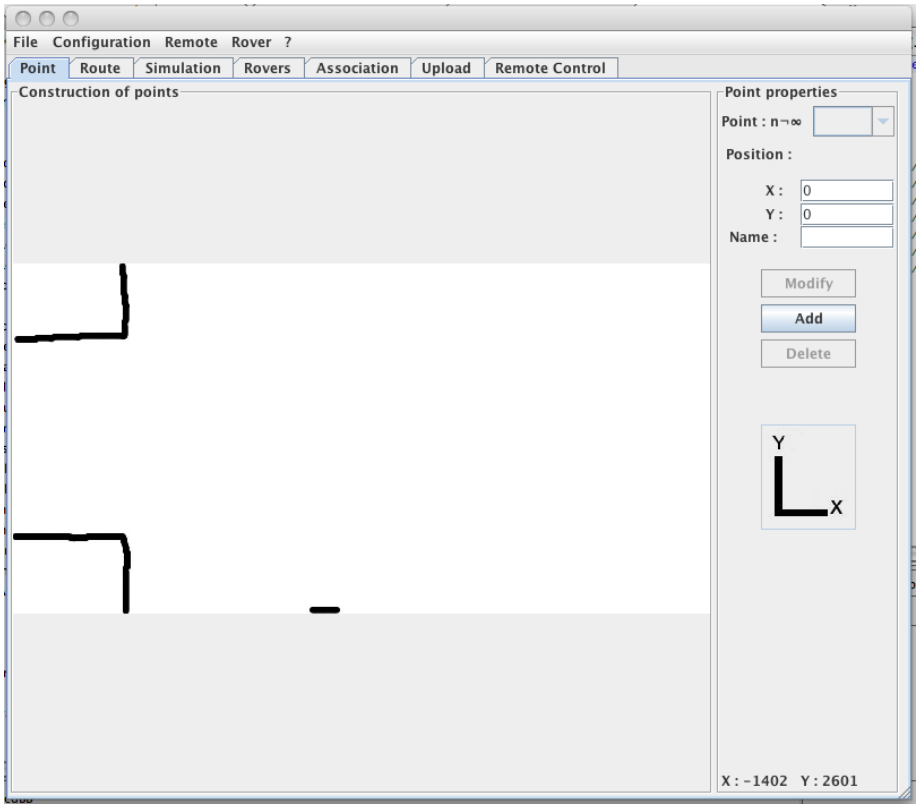
Restart the mac-book

Multiple RMI exceptions

Check the IP address – Restart the MapDesigner and the ComGenomServer

MapDesigner : Graphical User Interface - Java programs

The GUI to draw the mobility scenario and control the robots is in the “mapDesigner” package. You can run it via “MapDesignerLauncher.java” class.



It is composed of 7 different tabs : **“Point”** to define the passage points on the map, **“Route”** to define robot routes using the passage points, **“Simulation”** to see an overview of the robots movements, **“Rovers”** to manage connexions to the robots, **“Association”** to select the route each robot will execute, **“Upload”** to send the routes to the robots and run or stop their executions, **“Remote Control”** to manually control a robot and dynamically define a route.
You can save and load the complete map builder GUI environment or just the rovers settings with the top menu.

For more details you can read the ReadMe files in the folder “/data/mapDesigner” of the MobilityPlatform project.

“Point” Tab

In this tab you define passage points that will be used to build mobility scenario. You place them double clicking on the map or typing the coordinates. You can move a point by drag&drop and change his name or coordinate. If no points have been placed, you can also select a new picture for the map and change the scale of the map.
Note : Be careful to respect the scale factor of the room in the dimension of the picture.

The orientation of axis is show on the right of the screen.

"Route" Tab

Here you can create new routes.
Start typing the name of the route and click on the button "new" (or press Enter").
You then need to select a passage point on the map, by clicking on it, it will be the starting point of the route. One it is selected (you can see its ID on the right menu) click on "Add" button to add it to the route. You can add several passage points to you route, selecting a point then clicking on "Add" or pusing "Space" key. You see it being built dynamically. The green point is the starting point, the red one is the final point, if it is the same for both it appear orange.
If you route is a loop, you can set a number of repetition for this route in the field "Loop", validate clicking on "Set Loop".

You can define several routes and switch from one to another with the scroll list of route.

One a route is created you can edit the properties of each passing point or remove a point. You must select a point in the scroll list of points, not on the map. Then you can delete it or change one of the properties : "Depart" is the time (in second, relative to the start of the movement) when the robot will leave this point, "Break" indicate the time (in second) of a pause that the robot will make at this point, "Arrival" is the time (in second, relative to the start of the movement) when the robot will arrive at this point, the program change the speed of the robot between the previous point and this point to satisfy the arrival time, "Speed" is the speed (in mm/s) of the robot between this point and the next one.
Click on "Modify" button to validate the change, you can have an overview of a point settings by moving the mouse over it in the map.
Note : The speed of the robot is not implemented yet, in the GenoM trajectory control.

"Simulation" Tab

In this tab you can see an overview of the robots movements. You can select the routes you want to see and change the speed display factor.

Note : This display is just an overview and it doesn't reflect the real robots movements which will be smoothed between the different segments.

"Rover" Tab

Here you have to define the Name of the rover and the IP address of the host laptop where the "comGenomServer.java" program will be run. You can also specify a type and choose a colour for the robot.

"Association" Tab

In this tab you have to associate a route to a rover. A route can be associate to several rovers, but a rover can have only one route associate to it.

"Upload" Tab

In this tab you can see the list of the rovers which have a route associated. For each rover you can select :

"Send" to send to the rover its route instructions.

“Init” to order the robot to go to its starting position.

“Start” to start the robot movement.

“Stop” to stop the robot immediately.

“Break” to stop the robot at the next break in its curse. After a “Break” you can continue the route clicking on “Start”.

Note : Don't forget to execute the “Init” position before “Start” a run. You can select the same order for all the robots using the buttons in the bottom of the screen.

“Remote Control” Tab

In this tab you can manually control the robot. Select the robot in the scroll list (the program “comComRemoteServer” must be running on the embedded laptop), and click on connect.

You can then control the robot movements doing drag&drop on the yellow point in the middle of the joystick picture. To stop the robot, release the point.

If the position server is available you can see the position of the robot on the map. For that, type the name of the mobile to track in the corresponding field and validate pressing the key “Enter”. The robot pictogram should appear on the map. It is now possible to dynamically define a new route. Type a route name, click on “Record” then move the robot at the desired positions and click on “Take point” or press the key “Space” when you want to add a passing point, the route is automatically updated. When the route building is finish click on “Stop”. You can go back to the tab “Route” to modify your new route.