gctronic.com

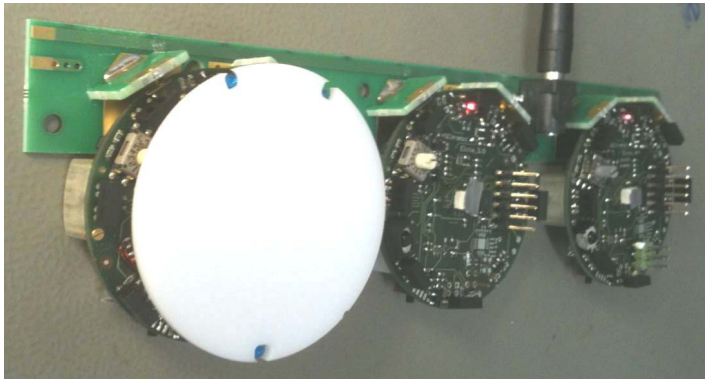
Elisa-3 - GCtronic wiki

46–59 Minuten

- [+]1 Overview
- [+]2 Hardware
- [+]3 Communication
- [+]4 Software
- [+]5 Odometry
- [+]6 Tracking
- 7 Local communication
- [+]8 ROS

- [+]9 Videos

1 Overview



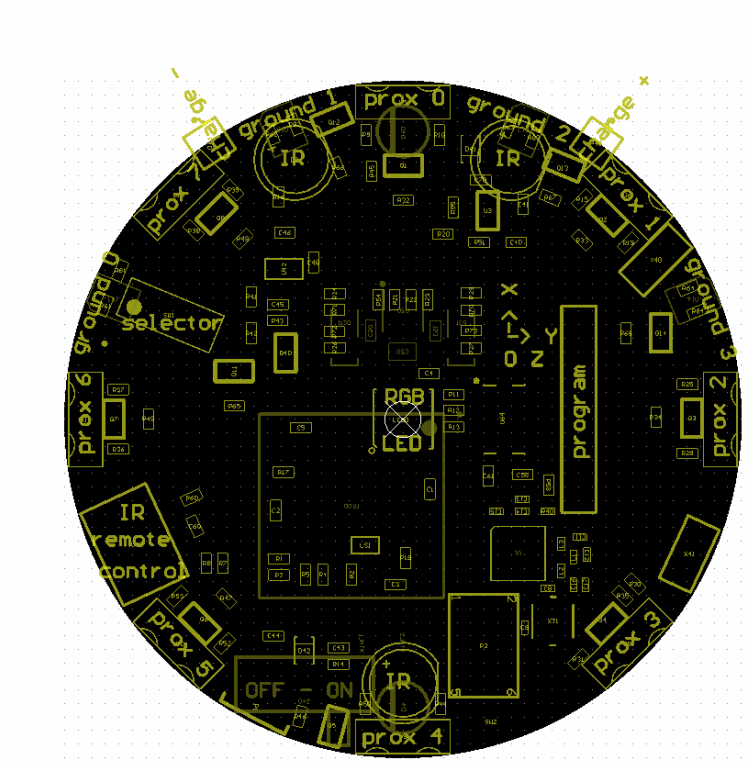
Elisa-3 is an evolution of the Elisa robot based on a different microcontroller and including a comprehensive set of sensors:

- Atmel 2560 microcontroller (Arduino compatible)
- central RGB led
- 8 green leds around the robot
- IRs emitters
- 8 IR proximity sensors (Vishay Semiconductors Reflective Optical

Sensor)

- 4 ground sensors (Fairchild Semiconductor Minature Reflective Object Sensor)
- 3-axis accelerometer (Freescale MMA7455L)
- RF radio for communication (Nordic Semiconductor nRF24L01+)
- micro USB connector for programming, debugging and charging
- IR receiver
- 2 DC motors
- top light diffuser
- selector

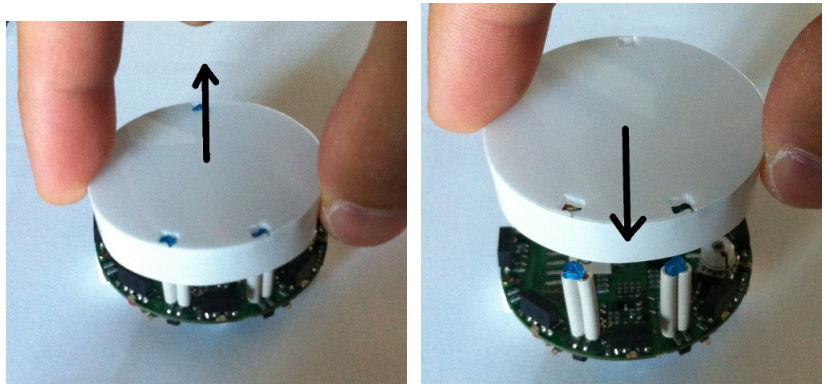
  The robot is able to self charge using the charger station, as shown in the previous figure. The following figure illustrates the position of the various sensors:



1.1 Useful information

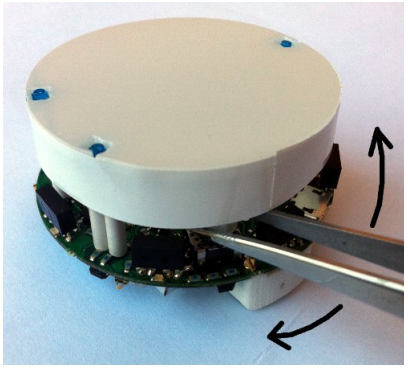
- the top light diffuser and robot are designed to lock together, but the diffuser isn't fixed and can thus be removed as desired; the top light diffuser, as the name suggests, helps the light coming from the RGB led to be smoothly spread out, moreover the strip attached

around the diffuser let the robot be better detected from others robots. Once the top light diffuser is removed, pay attention not to look at the RGB led directly. In order to remove the top light diffuser simply pull up it, then to place it back on top of the robot remember to align the 3 holes in the diffuser with the 3 IRs emitters and push down carefully untill the diffuser is stable; pay attention to not apply too much force on the IRs emitters otherwise they can bend and stop working.

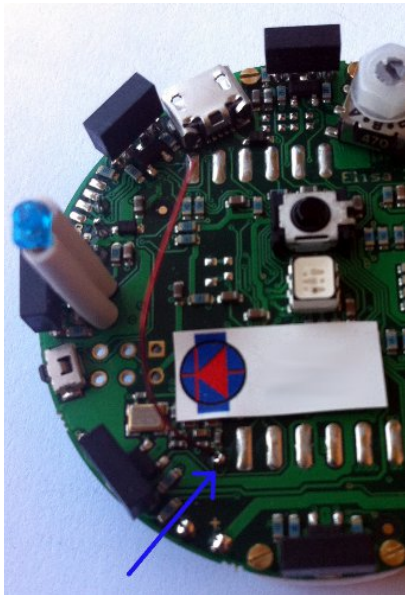


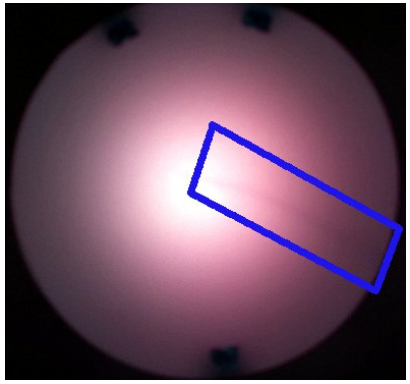
- when the top light diffuser is fit on top of the robot, then in order to change the selector position you can use the tweezers; the selector is located near the front-left IR emitter, as shown in the following

figure:



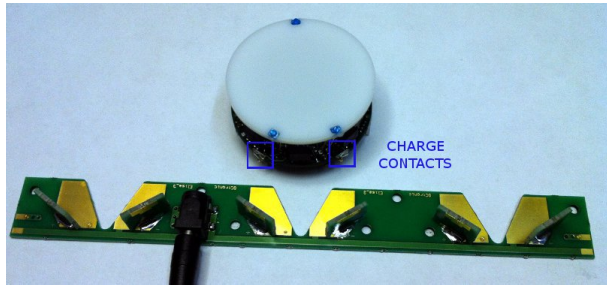
- if you encounter problems with the radio communication (e.g. lot of packet loss) then you can try moving the antenna that is a wire near the robot label. Place the antenna as high as possible, near the plastic top light diffuser; try placing it in the borders in order to avoid seeing a black line on the top light diffuser when the RGB led is turned on.

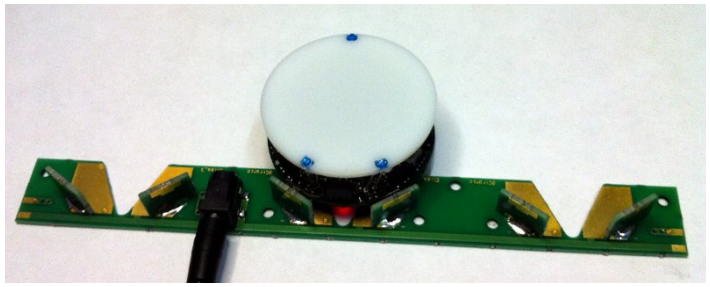




1.2 Robot charging

The Elisa-3 can be piloted in the charger station in order to be automatically self charged; there is no need to unplug the battery for charing. The following figures shows the robot approaching the charger station; a led indicates that the robot is in charge:





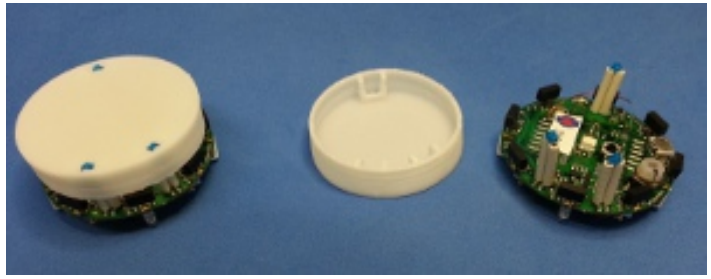
The microcontroller is informed when the robot is in charge and this information is also transferred to the PC in the *flags* byte; this let the user be able to pilote the robot to the charger station and be informed when it is actually in charge. More information about the radio protocol can be found in the section Communication.

Moreover the robot is also charged when the micro USB cable is connected to a computer; pay attention that if the USB cable is connected to a hub, this one need to be power supplied.

The following video shows the Elisa-3 piloted through the radio to the charging station using the monitor application:

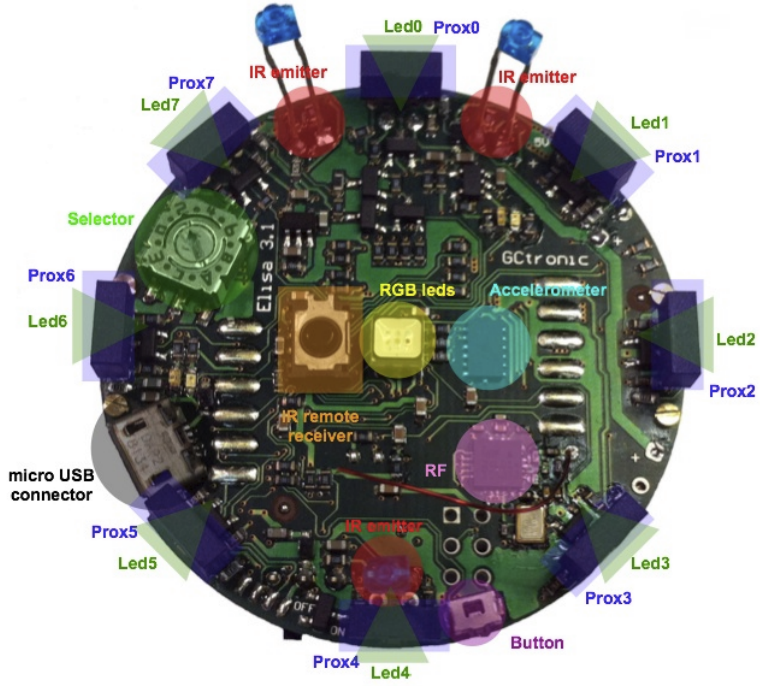
1.3 Top light diffuser

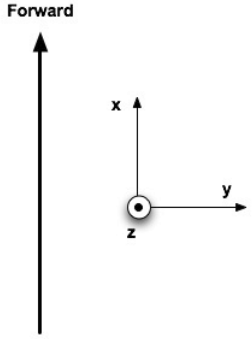
From February 2013 onwards the Elisa-3 is equipped with a new top light diffuser designed to fit perfectly in the 3 IRs emitters of the robot. The diffuser is made of plastic (3d printed), it is more robust and it simplifies the removal and insertion. Here is an image:



2 Hardware

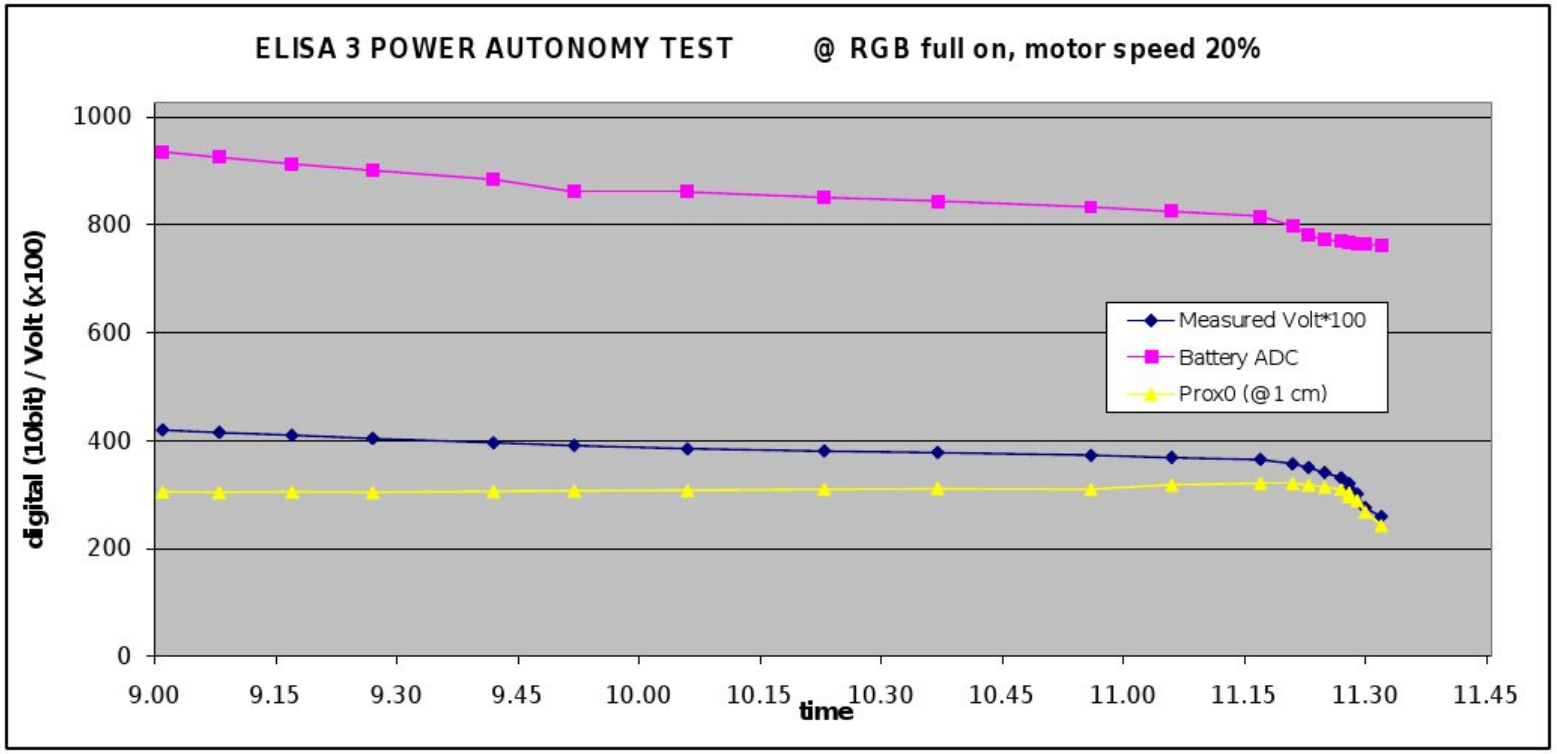
The following figures show the main components offered by the Elisa-3 robot and where they are physically placed:


Led0
Prox0
IR emitter
IR emitter
Prox7
Led7
Led1
Prox1
Selector
Elisa 3.1
GCtronic
Prox6
Led6
RGB leds
Accelerometer
Led2
Prox2
IR remote receiver
RF
micro USB connector
IR emitter
Prox5
Led5
Led3
Prox3
Button
Prox4
Led4


Forward
x
y
z

2.1 Power autonomy

The robot is equipped with two batteries for a duration of about 3 hours at normal usage (motors run continuously, IRs and RGB leds turned on).



2.2 Detailed specifications

Feature	Technical information
Size, weight	50 mm diameter, 30 mm height, 39 g
Battery, autonomy	LiIPo rechargeable battery (2 x 130 mAh, 3.7 V). About 3 hours autonomy. Recharging time

	about 1h e 30.
Processor	Atmel ATmega2560 @ 8MHz (~ 8 MIPS); 8 bit microcontroller
Memory	RAM: 8 KB; Flash: 256 KB; EEPROM: 4 KB
Motors	2 DC motors with a 25:1 reduction gear; speed controlled with backEMF
Magnetic wheels	Adesion force of about 1 N (100 g) depending on surface material and painting Wheels diamater = 9 mm Distance between wheels = 40.8 mm
Speed	Max: 60 cm/s
Mechanical structure	PCB, motors holder, top white plastic to diffuse light
IR sensors	8 infra-red sensors measuring ambient light

	and proximity of objects up to 6 cm; each sensor is 45° away from each other 4 ground sensors detecting the end of the viable surface (placed on the front-side of the robot)
IR emitters	3 IR emitters (2 on front-side, 1 on back-side of the robot)
Accelerometer	3D accelerometer along the X, Y and Z axis
LEDs	1 RGB LED in the center of the robot; 8 green LEDs around the robot
Switch / selector	16 position rotating switch
Communication	Standard Serial Port (up to 38kbps) Wireless: RF 2.4 GHz; the throughput depends on number of robot: eg. 250Hz for 4 robots, 10Hz for 100 robots; up to 10 m

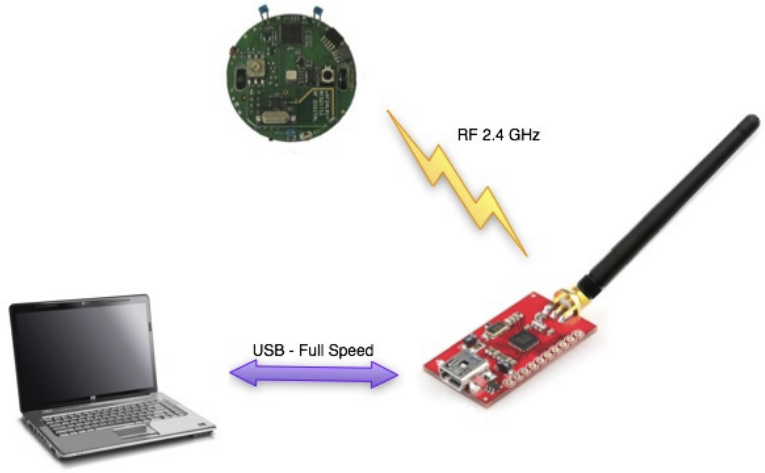
Remote Control	Infra-red receiver for standard remote control commands
Expansion bus	Optional connectors: 2 x UART, I2C, 2 x PWM, battery, ground, analog and digital voltage
Programming	C/C++ programming with the AVR-GCC compiler (WinAVR for Windows). Free compiler and IDE (AVR Studio / Arduino)

3 Communication

3.1 Wireless

The radio base-station is connected to the PC through USB and transfers data to and from the robot wirelessly. In the same way the radio chip (nRF24L01+) mounted on the robot communicates through SPI with the microcontroller and transfers data to and from

the PC wirelessly.
The robot is identified by an address that is stored in the last two bytes of the microcontroller internal EEPROM; the robot firmware setup the radio module reading the address from the EEPROM. This address corresponds to the robot id written on the label placed under the robot and should not be changed.



3.1.1 Packet format - PC to radio to robot

The 13 bytes payload packet format is shown below (the number in the parenthesis expresses the bytes):

Command (1)	Red led (1)	Blue led (1)	Green led (1)	IR + Flags (1)	Right motor (1)	Left motor (1)	Small green leds (1)	Flags2 (1)	F (

- Command: 0x27 = change robot state; 0x28 = goto base-station bootloader (this byte is not sent to the robot)
- Red, Blue, Green leds: values from 0 (OFF) to 100 (ON max power)
- IR + flags:
- first two bits are dedicated to the IRs:
- 0x00 => all IRs off
- 0x01 => back IR on
- 0x02 => front IRs on
- 0x03 => all IRs on

- third bit is reserved for enabling/disabling IR remote control (0=>disabled, 1=>enabled)
- fourth bit is used for sleep (1 => go to sleep for 1 minute)
- fifth bit is used to calibrate all sensors (proximity, ground, accelerometer) and reset odometry
- sixth bit is reserved (used by radio station)
- seventh bit is used for enabling/disabling onboard obstacle avoidance
- eight bit is used for enabling/disabling onboard cliff avoidance
- Right, Left motors: speed expressed in 1/5 of mm/s (i.e. a value of 10 means 50 mm/s); MSBit indicate direction: 1=forward, 0=backward; values from 0 to 127
- Small green leds: each bit define whether the corresponding led is turned on (1) or off (0); e.g. if bit0=1 then led0=on

- Flags2:
- bit0 is used for odometry calibration
- remaining bits unused
- Remaining bytes free to be used

3.1.1.1 Optimized protocol

The communication between the pc and the base-station is controlled by the master (computer) that continuously polls the slave (base-station); the polling is done once every millisecond and this is a restriction on the maximum communication throughput. To overcome this limitation we implemented an optimized protocol in which the packet sent to the base-station contains commands for four robots simultaneously; the base-station then separate the data and send them to the correct robot address. The same is applied in reception, that is the base-station is responsible of receiving the

ack payloads of 4 robots (64 bytes in total) and send them to the computer. This procedure let us have a throughput 4 times faster.

3.1.2 Packet format - robot to radio to PC

The robot send back to the base-station information about all its sensors every time it receive a command; this is accomplished by using the "ack payload" feature of the radio module. Each "ack payload" is 16 bytes length and is marked with an ID that is used to know which information the robot is currently transferring. The sequence is the following (the number in the parenthesis expresses the bytes):

ID=3 (1)	Prox0 (2)	Prox1 (2)	Prox2 (2)	Prox3 (2
ID=4 (1)	Prox4 (2)	Ground0 (2)	Ground1 (2)	Ground2

ID=5 (1)	ProxAmbient0 (2)	ProxAmbient1 (2)	ProxAmbient2 (2)	ProxAm (2)
ID=6 (1)	ProxAmbient4 (2)	GroundAmbient0 (2)	GroundAmbient1 (2)	GroundA (2)
ID=7 (1)	LeftSteps (4)	RightSteps (4)	theta (2)	xpos (2)

Pay attention that the base-station could return "error" codes in the first byte if the communication has problems:

- 0 => transmission succeed (no ack received though)
- 1 => ack received (should not be returned because if the ack is received, then the payload is read)
- 2 => transfer failed

Packet ID 3:

- Prox* contain values from 0 to 1023, the greater the values the nearer the objects to the sensor
- The *Flags* byte contains these information:
- bit0: 0 = robot not in charge; 1 = robot in charge
- bit1: 0 = button pressed; 1 = button not pressed
- bit2: 0 = robot not charged completely; 1 = robot charged completely
- the remainig bits are not used at the moment

Packet ID 4:

- Prox4 contains values from 0 to 1023, the greater the values the nearer the objects to the sensor
- Ground* contain values from 512 to 1023, the smaller the value the darker the surface

- AccX and AccY contain raw values of the accelerometer; the range is between -64 to 64
- TV remote contains the last interpreted command received through IR

  Packet ID 5:

- ProxAmbient* contain values from 0 to 1023, the smaller the values the brighter the ambient light
- Selector contains the value of the current selector position

  Packet ID 6:

- ProxAmbient4 contains values from 0 to 1023, the smaller the values the brighter the ambient light
- GroundAmbient* contain values from 0 to 1023, the smaller the values the brighter the ambient light
- AccZ contains raw values of the accelerometer; the range is

between 0 and -128 (upside down)

- Battery contains the sampled value of the battery, the values range is between 780 (battery discharged) and 930 (battery charged)

  Packet ID 7:

- LeftSteps and RightSteps contain the sum of the sampled speed for left and right motors respectively (only available when the speed controller isn't used; refer to xpos, ypos and theta when the speed controller is used)
- theta contains the orientation of the robot expressed in 1/10 of degree (3600 degrees for a full turn); available only when the speed controller is enabled
- xpos and ypos contain the position of the robot expressed in millimeters; available only when the speed controller is enabled

3.2 USB cable

You can directly connect the robot to the computer to make a basic functional test. You can find the source code in the following link Elisa3-global-test.zip (Windows).
To start the test follow these steps:

1. put the selector in position 6
2. connect the robot to the computer with the USB cable and turn it on
3. run the program, insert the correct COM port and choose option 1

With the same program you can also change the ID of the robot by choosing option 2 in the last step (not recommended).

Via USB cable you can also program the robot with Aseba.

4 Software

4.1 Robot

4.1.1 Requirements

In order to communicate with the robot through the micro USB the FTDI driver need to be installed. If a serial port is automatically created when connecting the robot to the computer you're done otherwise you need to download the drivers for your system and architecture:

- Windows Vista/XP, Windows 7/8/10 (run as administrator)
- Ubuntu: when the robot is connected the port will be created in `/dev/ttyUSB0` (no need to install a driver)
- Mac OS X 10.3 to 10.8 (32 bit), Mac OS X 10.3 to 10.8 (64 bit), Mac OS X 10.9 and above; after installing the driver the port will be created in `/dev/tty.usbserial-...`; you can find a guide on how to install the driver in the following link AN_134_FTDI_Drivers_Installation_Guide_for_MAC_OSX.pdf

  All the drivers can be found in the official page from the following link FTDI drivers.

Starting from robot ID 4000 the USB to serial chip can be one of the following: FTDI, Cypress CY7C65213 or Silicon Labs CP2102; this is due to chips availability.

4.1.2 AVR Studio 4 project

The projects are built with AVR Studio 4 released by Atmel.
The projects should be compatible also with newer versions of Atmel Studio, the last version is available from https://www.microchip.com/mplab/avr-support/avr-and-sam-downloads-archive.

4.1.2.1 Basic demo

This project is thought to be a starting point for Elisa-3 newbie users and basically contains a small and clean main with some basic demos selected through the hardware selector that show how

to interact with robot sensors and actuators. The project source can be downloaded from the repository https://github.com/gctronic/elisa3_firmware_basic; the hex file can be directly downloaded from Elisa-3 basic firmware hex. To program the robot refer to section Programming.

Selector position and related demo:

- 0: no speed controller activated => free running (all others positions have the speed controller activated)
- 1: obstacle avoidance enabled
- 2: cliff avoidance enabled (currently it will simply stop before falling and stay there waiting for commands)
- 3: both obstacle and cliff avoidance enabled
- 4: random RGB colors and small green leds on
- 5: robot moving forward with obstacle avoidance enabled and random RGB colors

4.1.2.2 Advanced demo

This is an extension of the *basic demo project*, basically it contains some additional advanced demos. The project source can be downloaded from the repository https://github.com/gctronic/elisa3_firmware_advanced.git; the hex file can be directly downloaded from Elisa-3 advanced firmware hex. To program the robot refer to section Programming.
Selector position and related demo:

- 0: no speed controller activated => free running (all others positions have the speed controller activated)
- 1: obstacle avoidance enabled
- 2: cliff avoidance enabled (currently it will simply stop before falling and stay there waiting for commands)
- 3: both obstacle and cliff avoidance enabled

- 4: random RGB colors and small green leds on
- 5: robot moving forward with obstacle avoidance enabled and random RGB colors
- 6: robot testing and address writing through serial connection (used in production)
- 7: automatic charging demo (refer to section Videos), that is composed of 4 states:
- random walk with obstacle avoidance
- search black line
- follow black line that lead to the charging station
- charge for a while
- 8: autonomous odometry calibration (refer to section Autonomous calibration)
- 9: write default odometry calibration values in EEPROM (hard-

coded values); wait 2 seconds before start writing the calibration values

- 10: robot moving forward (with pause) and obstacle avoidance enabled; random RGB colors and green led effect
- 11: local communication: robot alignment
- 12: local communication: 2 or more robots exchange data sequentially
- 13: local communication: listen and transmit continuously; when data received change RGB color
- 14: local communication: RGB color propagation
- 15: clock calibration (communicate with the PC through the USB cable to change the OSCCAL register); this position could also be used to remote contol the robot through the radio (only speed control is enabled)

4.1.2.3 Atmel Studio 7

If you are working with Atmel Studio 7, you can simply use the provided AVR Studio 4 projects by importing them directly in Atmel Studio 7: `File => Import => AVR Studio 4 Project`, then select `Elisa3-avr-studio.aps` and click on `Convert`.

4.1.3 Arduino IDE project

The project is built with the Arduino IDE 1.x freely available from the official Arduino website. In order to build the Elisa-3 firmware with the Arduino IDE 1.x the following steps has to be performed:

- 1. download the Arduino IDE 1.x (the last known working version is 1.8.9, refer to Arduino Software) and extract it, let say in a folder named `arduino-1.x`
- 2. download the Elisa-3 Arduino library and extract it within the libraries folder of the Arduino IDE, in this case `arduino-`

`1.x\libraries` (see Find-sketches-libraries-board-cores-and-other-files-on-your-computer for more information on Arduino useful paths); you should end up with a `Elisa3` folder within the libraries.
If you start the Arduino IDE now you can see that the `Elisa-3` library is available in the menu `Sketch=>Import Library...` (or `Sketch=>Include Lirary` in later IDE versions).
In later versions of Arduino IDE you can also install the library via menu: `Sketch=>Include Library=>Add .ZIP library`, for more info have a look at importing-a-zip-library.

- 3. the file `boards.txt` in the Arduino IDE folder `arduino-1.x\hardware\arduino` (or `arduino-1.x\hardware\arduino\avr` or `AppData\Local\Arduino15\packages\arduino\hardware\avr\1.8.6` in later IDE versions) need to be changed to contain the definitions for the Elisa-3 robot, add the following definitions at the end of the file:

```
##############################################################
```

```
elisa3.name=Elisa 3 robot

elisa3.upload.tool=avrdude
elisa3.upload.tool.serial=avrdude
elisa3.upload.protocol=stk500v2
elisa3.upload.maximum_size=258048
elisa3.upload.speed=57600

elisa3.bootloader.low_fuses=0xE2
elisa3.bootloader.high_fuses=0xD0
elisa3.bootloader.extended_fuses=0xFF
elisa3.bootloader.path=stk500v2-elisa3
elisa3.bootloader.file=stk500v2-elisa3.hex
elisa3.bootloader.unlock_bits=0x3F
elisa3.bootloader.lock_bits=0x0F
```

elisa3.build.mcu=atmega2560
elisa3.build.f_cpu=8000000L
elisa3.build.board=AVR_ELISA3
elisa3.build.core=arduino
elisa3.build.variant=mega

##

- 4. this step need to be performed only with later IDE versions, when you receive a warning like this `Bootloader file specified but missing...` during compilation.
  In this case place the bootloader hex file (`stk500v2.hex`) you can find in the Bootloader section in the directory `arduino-1.x\Arduino\hardware\arduino\avr\bootloaders\` and name it `stk500v2-elisa3.hex`
- 5. download the Elisa-3 project file and open it with the Arduino IDE

(you should open the file "*elisa3.ino*")

- 6. select `Elisa-3 robot` from the `Tools=>Board` menu; click on the `Verify` button to build the project
- 7. to upload the resulting hex file, attach the micro usb and set the port from the `Tools=>Serial Port` menu consequently; turn on the robot and click on the `Upload` button

    Windows users: if you have problems in uploading the firmware, try opening a command prompt and issue the command `c:\windows\system32\mode.com com62: dtr=on` (beware to change serial port number according to your system) before uploading from the Arduino IDE.

You can download the Arduino IDE 1.0.5 for Linux (32 bits) containing an updated avr toolchain (4.5.3) and the Elisa3 library from the following link arduino-1.0.5-linux32.zip.

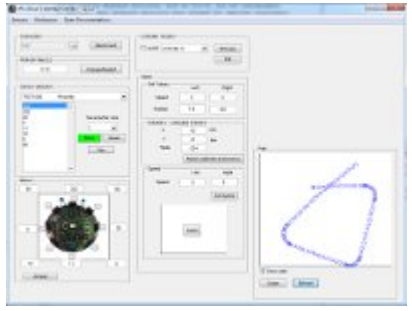
If the `Tools->Serial Port` menu is grayed out then you need to start the Arduino IDE in a terminal typing `sudo path/to/arduino`.

If you want to have access to the compiler options you can download the following project Elisa3-arduino-makefile.zip that contains an Arduino IDE project with a Makefile, follow the instructions in the "readme.txt" file in order to build and upload to the robot.

4.1.4 Aseba

Refer to the page Elisa-3 Aseba.

4.1.5 Matlab

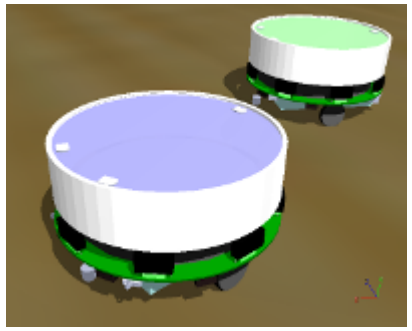


The ePic2 Matlab interface was adapted to work with the Elisa-3 robot. The communication is handled with the radio module. Both Matlab 32 bits and 64 bits are supported (tested on Matlab R2010a). Follow these steps to start playing with the interface:

1. program the robot with the advanced demo
2. place the selector in position 15 (to pilot the robot through the interface with no obstacle and no cliff avoidance)
3. connect the radio base-station to the computer
4. download the ePic2 for Elisa-3 from the repository https://github.com/gctronic/elisa3_epic.git: either from github site clicking on `Code=>Download ZIP` or by issueing the command `git clone https://github.com/gctronic/elisa3_epic.git`
5. open (double click) the file *main.m*; once Matlab is ready type *main+ENTER* and the GUI should start
6. click on the + sign (top left) and insert the robot address (e.g 3307),

then click on *Connect*

4.1.6 Webots simulator



The following features have been included in the Elisa-3 model for the Webots simulator:

- proximity sensors
- ground sensors
- accelerometer
- motors
- green leds around the robot

- RGB led
- radio communication

  You can donwload the Webots project containig the Elisa-3 model (proto) and a demonstration world in the following link Elisa-3-webots.zip.

  You can download a Webots project containing a demonstration world illustrating the usage of the radio communication between 10 Elisa-3 robots and a supervisor in the following link Elisa-3-webots-radio.zip. Here is a video of this demo:

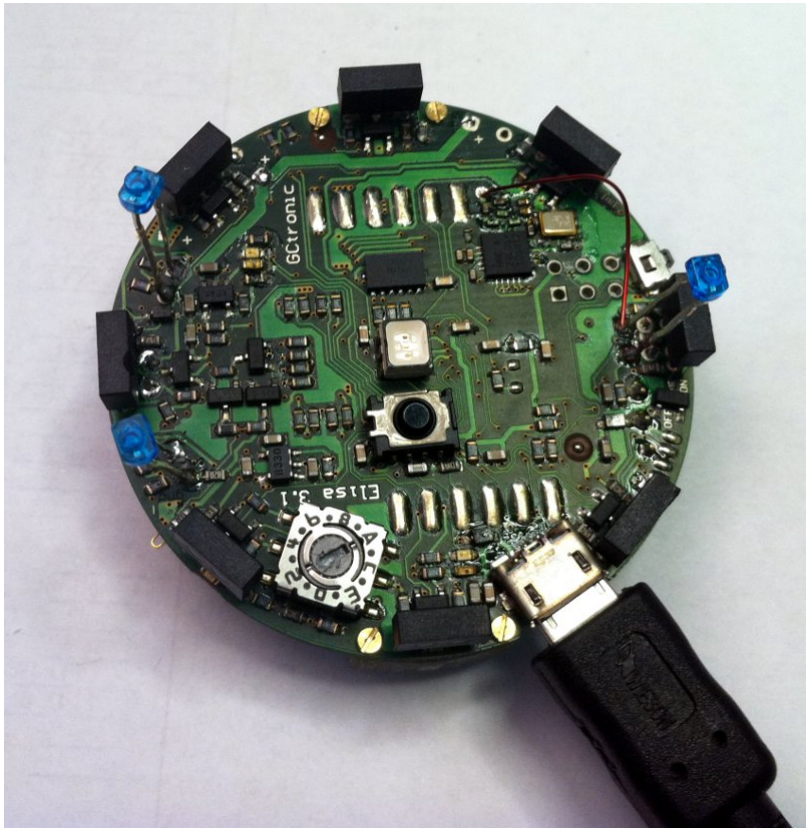
4.1.7 Onboard behaviors

The released firmware contains two basic onboard behaviors: obstacle and cliff avoidance. Both can be enabled and disabled from the computer through the radio (seventh bit of flags byte for obstacle avoidance, eight bit of flags byte for cliff avoidance).

The following videos show three robots that have their obstacle avoidance enabled:

4.1.8 Programming

The robot is pre-programmed with a serial bootloader. In order to upload a new program to the robot a micro USB cable is required. The connection with the robot is shown below:



If you are working with the Arduino IDE you don't need to follow this procedure, refer instead to section Arduino IDE project.

If you encounter some problem during programming (e.g. timeout problems) you can try following this sequence: turn on the robot, unplug the robot from the computer, plug the robot into the computer, it will make some blinks; when the blinks

terminate execute the programming command again.
Beware that every time you need to re-program the robot you need to unplug and plug again the cable to the computer.

4.1.8.1 Windows 7

1. Download the Windows 7 package and extract it. The package contains also the FTDI driver. Beware that starting from robot id 4000 the USB driver might be different, refer to section Requirements, so you need to install it manually in case it isn't an FTDI chip.
2. Execute the script `config.bat` and follow the installation; beware that this need to be done only once. The script will ask you to modify the registry, this is fine (used to save application preferences).
3. Connect the robot to the computer; the COM port will be created.

4. Run the application `AVR Burn-O-Mat.exe`; you need to configure the port to communicate with the robot:
1. click on `Settings => AVRDUDE`
2. in the `AVRDUDE Options`, on `Port` enter the name of the port just created when the robot was connected to the computer (e.g. COM10); then click `Ok`
5. In the `Flash` section search the hex file you want to upload on the robot.
6. Turn on the robot, wait the blinks terminate and then click on `Write` in the `Flash` section.
   If you get an `Access is denied` error, then run `AVR Burn-O-Mat.exe` as administrator.
7. During the programming the robot will blink; at the end you'll receive a message saying `Flash succesfully written.`

4.1.8.2 Mac OS X

The following procedure is tested in Max OS X 10.10, but should work from Mac OS X 10.9 onwards; in these versions there is built-in support for the FTDI devices.

1. Download the Mac OS X package and extract it.
2. Execute the script `config.sh` in the terminal, it will ask you to install the Java Runtime Environment; in case there is a problem executing the script try with `chmod +x config.sh` and try again. Beware that this need to be done only once.
3. Connect the robot to the computer; the serial device will be created (something like `/dev/tty.usbserial-AJ03296J`).
4. Run the application `AVR Burn-O-Mat`; you need to configure the port to communicate with the robot:

1. click on `Settings => AVRDUDE`

2. in the `AVRDUDE Options`, on `Port` enter the name of the port just created when the robot was connected to the computer; then click `Ok`
5. In the `Flash` section search the hex file you want to upload on the robot.
6. Turn on the robot, wait the blinks terminate and then click on `Write` in the `Flash` section.
7. During the programming the robot will blink; at the end you'll receive a message saying `Flash succesfully written.`

4.1.8.3 Linux

The following procedure was tested in Ubunut 12.04, but a similar procedure can be followed in newer systems and other Linux versions.
You can find a nice GUI for `avrdude` in the following link

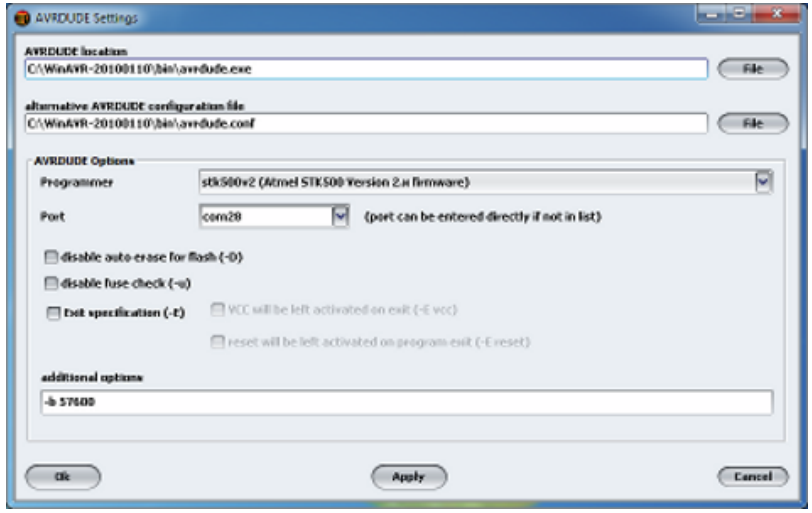
https://burn-o-mat.net/avr8_burn_o_mat_avrdude_gui_en.php; you can download directly the application for Ubuntu from the following link avr8-burn-o-mat-2.1.2-all.deb.
Double click the package and install it; the executable will be `avr8-burn-o-mat`.
Beware that the application requires the Java SE Runtime Environment (JRE) that you can download from the official page https://www.oracle.com/technetwork/java/javase/downloads/index.html, alternatively you can issue the command `sudo apt-get install openjdk-8-jre` in the terminal.

The application need a bit of configuration, follow these steps:

1. connect the robot to the computer, the serial device will be created (something like /dev/USB0)
2. to use the USB port the permissions need to be set to read and write issueing the command `sudo chmod a+rw`

`/dev/ttyUSB0`

3. start the application and click on `Settings => AVRDUDE`
4. set the location of `avrdude` and the related configuration file (refer to the previous section when `avrdude` was installed to know the exact location); the configuration file is in `/etc/avrdude.conf`
3. click `OK`, close the application and open it again (this is needed to load the configuration file information); click on `Settings => AVRDUDE`
4. select `stk500v2` as the `Programmer`
5. set the serial port connected to the robot (`/dev/ttyUSB0`)
6. in `additional options` insert `-b 57600`, you will end up with a window like the following one:



7. click `OK`; select `ATmega2560` in the `AVR type`
8. in the `Flash` section search the hex file you want to upload on the robot; select `Intel Hex` on the right
9. connect the robot to the computer, turn on the robot, wait the blinks terminate and then click on `Write` in the `Flash` section
10. during the programming the robot will blink; at the end you'll receive a message saying `Flash succesfully written.`

4.1.8.4 Command line

The avrdude utility is used to do the upload, you can download it directly from the following links depending on your system:

- Windows (tested on Windows 7 and 10); `avrdude` will be installed in the path `C:\WinAVR-20100110\bin\avrdude`; avrdude version 5.10
- Mac OS X; `avrdude` will be installed in the path `/usr/local/CrossPack-AVR/bin/avrdude`; to check the path issue the commmand `which avrdude` in the terminal; avrdude version 6.0.1
- Ubuntu (12.04 32-bit): issue the command `sudo apt-get install avrdude` in the terminal; `avrdude` will be installed in the path `/usr/bin/avrdude`; to check the path issue the commmand `which avrdude` in the terminal; avrdude version 5.11.1

Open a terminal and issue the following commands:

1. only for windows users: `c:\windows\system32\mode.com com10: dtr=on`.

2. `avrdude -p m2560 -P COM10 -b 57600 -c stk500v2 -D -Uflash:w:Elisa3-avr-studio.hex:i -v`

   where `COM10` must be replaced with your com port and `Elisa3-avr-studio.hex` must be replaced with your application name; in Mac OS X the port will be something like `/dev/tty.usbserial-...`, in Ubuntu will be `/dev/ttyUSB0`. The Basic demo and Advanced demo have this command contained in the file `program.bat` in the `default` directory within the project, this can be useful for Windows users.

4.1.9 Internal EEPROM

The internal 4 KB EEPROM that resides in the microcontroller is pre-programmed with the robot ID in the last two bytes (e.g. if ID=3200 (0x0C80), then address 4094=0x80 and address 4095=0x0C). The ID represents also the RF address that the robot uses to communicate with the computer and is automatically read

at startup (have a look a the firmware for more details).
Moreover the address 4093 is used to save the clock calibration value that is found during production/testing of the robots; this value hasn't to be modified otherwise some functionalities such as tv remote control could not work anymore. For more information on clock calibration refers to the applicaiton note AVR053: Calibration of the internal RC oscillator.
The Elisa-3 robot supports an autonomous calibration process and the result of this calibration is saved in EEPROM starting at address 3946 to 4092.
The size of usable EEPROM is thus 3946 bytes (0-3945) and the remaining memory must not be modified/erased.

In order to program the eeprom an AVR programmer is required, we utilize the Pocket AVR Programmer from Sparkfun (recognized as USBtiny device); then with the avrdude utility the following command has to be issued:

```
avrdude -p m2560 -c usbtiny -v -U eeprom:w:Elisa3-eeprom.hex:i
-v -B 1
```

where *Elisa3-eeprom.hex* is the EEPROM memory saved as Intel Hex format (eeprom example); a possible tool to read and write Intel Hex format is Galep32 from Conitec Datensysteme. Alternatively a program designed to writing to these EEPROM locations can be uploaded to the robot, in case an AVR programmer isn't available. The project source is available in the repository https://github.com/gctronic/elisa3_eeprom.git; it is simply needed to modify the address, rebuild and upload to the robot.

4.1.10 Bootloader

In case the bootloader of the Elisa-3 is erased by mistake, then you can restore it by using an AVR programmer. You can download the bootloader from here stk500v2.hex; the source code is available from the repository https://github.com/gctronic

/elisa3_bootloader.git.

Avrdude can be used to actually write the bootloader to the robot with a command similar to the following one:

```
avrdude -p m2560 -c stk500v2 -P COM348 -v -U lfuse:w:0xE2:m -U hfuse:w:0xD8:m -U efuse:w:0xFF:m -V -U flash:w:stk500v2.hex:i -v -B 2
```

Here we used a programmer recognized as a serial device (port COM348) that utilizes the `stk500v2` protocol.

4.2 Base-station

This chapter explains informations that aren't needed for most of the users since the radio module is ready to be used and don't need to be reprogrammed. Only if you are interested in the firmware running in the radio module and on how to reprogram it then refer to section https://www.gctronic.com/doc/index.php/Elisa#Base-station (chapter 4.2) of the Elisa robot wiki.

4.3 PC side

This section gives informations related to the radio module connected to the computer; if you don't have a radio module you can skip this section.

4.3.1 Requirements

Refer to the section https://www.gctronic.com/doc/index.php/Elisa#1._Install_the_radio_base-station_driver.

4.3.2 Elisa-3 library

This library simplify the implementation of applications on the pc side (where the radio base-station is connected) that will take control of the robots and receive data from them. Some basic examples will be provided in the following sections to show how to use this library.

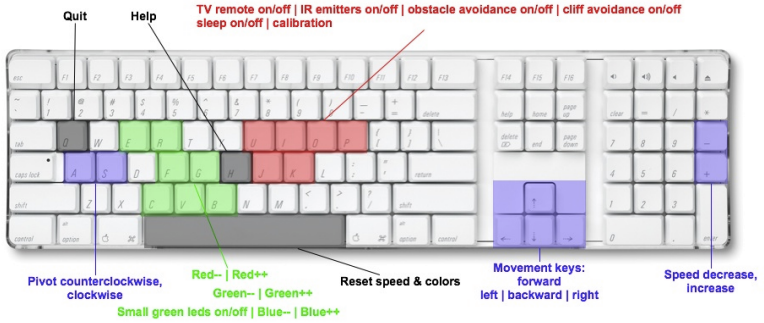
The source code of the library is available in the repository https://github.com/gctronic/elisa3_remote_library; follow the instructions in the repository to build the library.

4.3.3 Multiplatform monitor

The demo is a command line monitor that shows all the sensors information (e.g. proximity, ground, acceleromter, battery, ...) and let the user move the robot and change its colors and behavior with the keyboard. The data are sent using the protocol described in the previous section.
The following figures show the monitor on the left and the available commands on the right.

```
*****************************
*** RECEPTION robot -> pc ***          Press h for help
*****************************
PROXIMITY
Prox0    Prox1    Prox2    Prox3    Prox4    Prox5    Prox6    Prox7
  1        1        1        2        0        0        0        1
PROXIMITY AMBIENT
Prox0    Prox1    Prox2    Prox3    Prox4    Prox5    Prox6    Prox7
 992     1003      997     1000      997      982      968      960
GROUND
ground0  ground1          ground2          ground3
 514      515              514              515
GROUND AMBIENT
ground0  ground1          ground2          ground3
1017     1016             1016             1018
ACCELEROMETER
X:   2   Y:  -1  Z:  -1
angle: 116 degrees (on vertical wall)
BATTERY
adc:  884 (~ 67%)
FLAGS
0x00
TV REMOTE
00
SELECTOR
15
MOTORS ENCODERS
l: +0000000000 r: +0000000000
ODOMETRY
X:   0   Y:   0  theta:   0
*****************************
*** TRANSFER robot <- pc ***
*****************************
SPEED
speed to send:  +0
left:  +0        right:  +0
RGB LEDS
r:   0   g:   0  b:   0
FLAGS
0x00

RF LINK QUALITY
 96.89 %
```



The source can be downloaded from the repository https://github.com/gctronic/elisa3_remote_monitor.

4.3.3.1 Windows

Execution:

- install the driver contained in the nRFgo Studio tool if not already done; this let the base-station be recognized as a WinUSB device (bootloader), independently of whether the libusb library is installed or not
- once the driver is installed, the pre-compiled "exe" (under `\bin\Release` dir) should run without problems; the program will prompt you the address of the robot you want to control

Compilation:
the Code::Blocks project should already be setup to reference the Elisa-3 library headers and lib files, anyway you need to put this project within the same directory of the Elisa-3 library, e.g. you should have a tree similar to the following one:

- Elisa-3 demo (parent dir)
- `elisa3_remote_library` (Elisa-3 library project)
- `elisa3_remote_monitor` (current project)

4.3.3.2 Linux / Mac OS X

The project was tested to work also in Ubuntu and Mac OS X (no driver required).
Compilation:

- you need to put this project within the same directory of the Elisa-3 library
- build command: go under "linux" dir and type `make clean && make`

Execution:

- `sudo ./main`

4.3.4 Communicate with 4 robots simultaneously

This example shows how to interact with 4 robots simlutaneously, basically it shows the sensors information (proximity and ground) coming from 4 robots and let control one robot at a time through the keyboard (you can change the robot you want to control). The source can be downloaded from the repository https://github.com/gctronic/elisa3_remote_multiple. For building refer to the section Multiplatform monitor.

4.3.5 Obstacle avoidance

This demo implements the *obstacle avoidance* behavior controlling the robot from the pc through the radio; this means that the robot reacts only to the commands received using the basic communication protocol and has no "intelligence" onboard. The demo uses the information gathered from the 3 front proximity

sensors and set the motors speed accordingly; moreover the RGB LED is updated with a random color at fixed intervals.
The source can be downloaded from the repository https://github.com/gctronic/elisa3_remote_oa. For building refer to the section Multiplatform monitor.
The following video shows the result:

It is available also the same example but with 4 robots controlled simultaneously; the source can be downloaded from the branch `4robots` of the repository https://github.com/gctronic/elisa3_remote_oa
It is easy to extend the previous example in order to control many robots, the code that controls 8 robots simultaneously can be downloaded from the branch `8robots` of the repository https://github.com/gctronic/elisa3_remote_oa.

4.3.6 Cliff avoidance

This demo implements the *cliff avoidance* behavior controlling the robot from the pc through the radio; as with the *obstacle avoidance* demo, the robot reacts only to the commands received from the radio. The demo uses the information gathered from the 4 ground sensors to stop the robot when a cliff is detected (threshold tuned to run in a white surface); moreover the RGB LED is updated with a random color at fixed intervals.
The source can be downloaded from the repository https://github.com/gctronic/elisa3_remote_cliff. For building refer to the section Multiplatform monitor.
The following video shows the result:

4.3.7 Set robots state from file

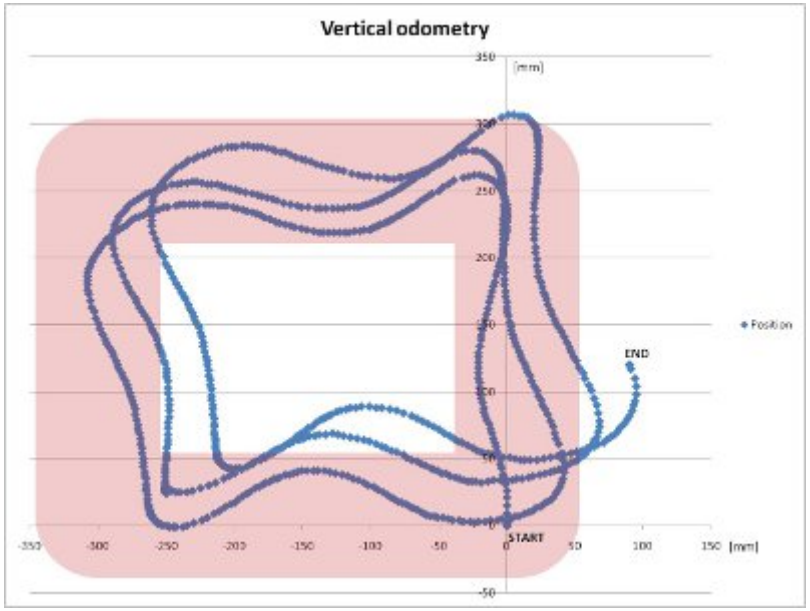
This project show how to send data to robots for which we will know the address only at runtime, in particular the content of the packets to be transmitted is parsed from a csv file and the interpreted

commands are sent to the robots one time. The source can be downloaded from the repository https://github.com/gctronic/elisa3_remote_file. For building refer to the section Multiplatform monitor.

5 Odometry

The odometry of Elisa-3 is quite good even if the speed is only measured by back-emf. On vertical surfaces the absolute angle is given by the accelerometer measuring g... quite a fix reference without drifting ;-)
A fine calibration of the right and left wheel speed parameters might give better results. However the current odometry is a good estimate of the absolute position from a starting point. The experiments are performed on a square labyrinth and the robot advances doing obstacle avoidance. The on-board calculated (x,y,theta) position is sent to a PC via radio and logged for further

display.



Details about the code can be found in the advanced-demo project, in particular the *motors.c* source file. The PC application used for logging data is the monitor.

5.1 Autonomous calibration

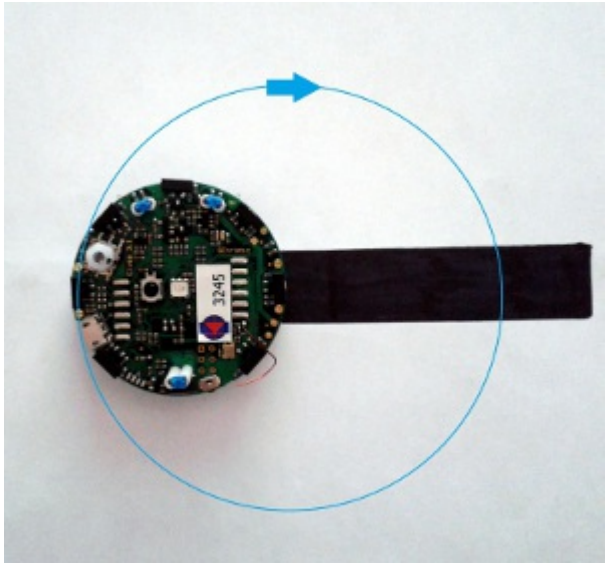
Since the motors can be slightly different a calibration can improve the behavior of the robot in terms of maneuverability and odometry

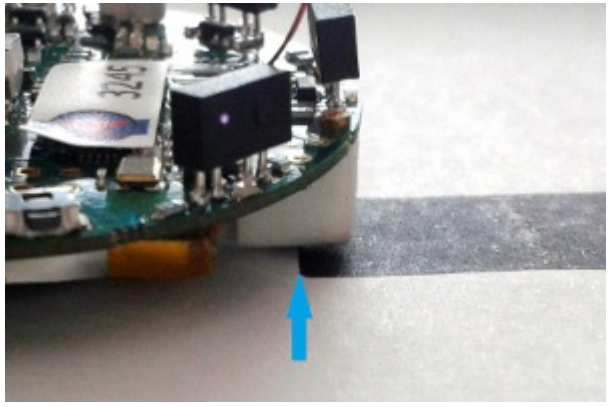
accuracy. An autonomous calibration process is implemented onboard: basically a calibration is performed for both the right and left wheels in two modes that are forward and backward with speed control enabled. In order to let the robot calibrate istelf a white sheet in which a black line is drawed is needed; the robot will measure the time between detection of the line at various speeds. The calibration sheet can be downloaded from the following link calibration-sheet.pdf.

In order to accomplish the calibration the robot need to be programmed with the advanced firmare and a specific command has to be sent to the robot through the radio module or the TV remote; if you are using the radio module you can use the monitor application in which the letter *l (el)* is reserved to launch the calibration, otherwise if you have a TV remote control you can press the button *5*. The sequence is the following:

1. put the selector in position 8

2. place the robot near the black line as shown below; the left motor is the first to be calibrated. Pay attention to place the right wheel as precise as possible with the black line

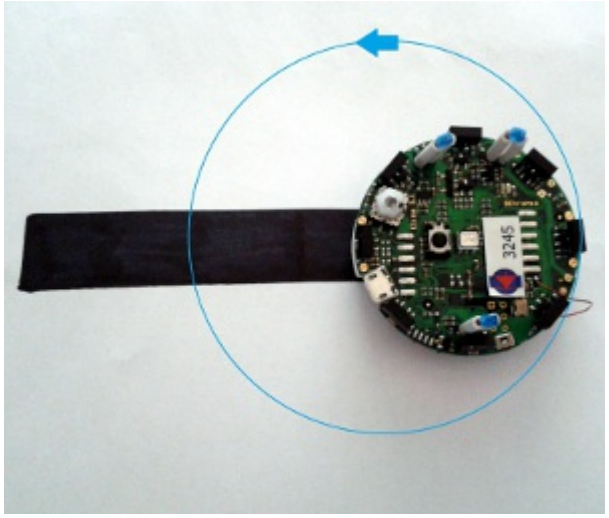




3. once the robot is placed you can type the *l (el)* command (or press the button *5*); wait a couple of minutes during which the robot will do various turns at various speed in the forward direction and then in the backward direction

4. when the robot terminated (robot is stopped after going backward at high speed) you need to place it in the opposite

direction in order to calibrate the right motor, as shown below.



5. once the robot is placed you can type again the *l (el)* command (or press the button *5*)
6. when the robot finish, the calibration process is also terminated.

The previous figures show a robot without the top diffuser, anyway you don't need to remove it!

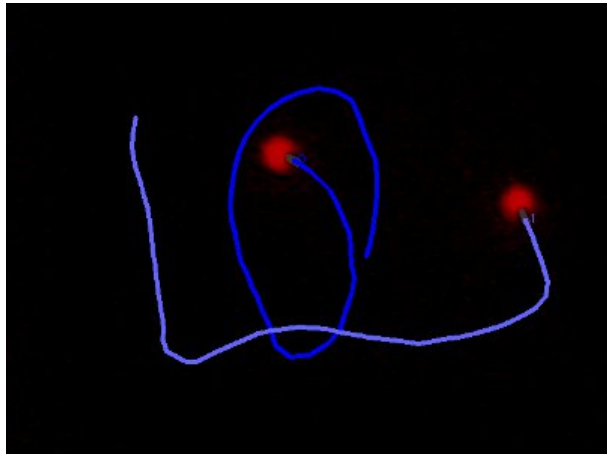
6 Tracking

6.1 Assembly documentation

You can download the documentation from here tracking-doc.pdf.
Have a look also at the video:

6.2 SwisTrack

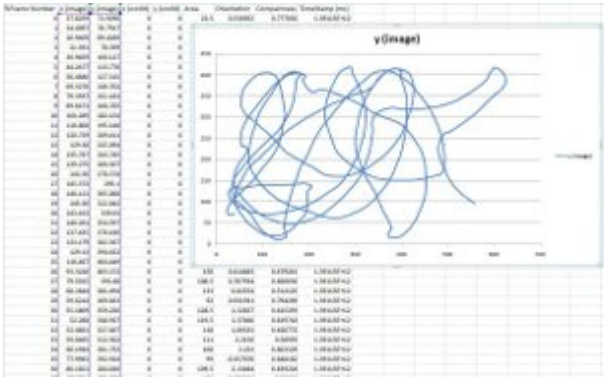
Some experiments are done with the SwisTrack software in order to be able to track the Elisa-3 robots through the back IR emitter, here is a resulting image with 2 robots:



The pre-compiled SwisTrack software (Windows) can be downloaded from the following link SwisTrack-compiled.

The following video shows the tracking of 5 robots:

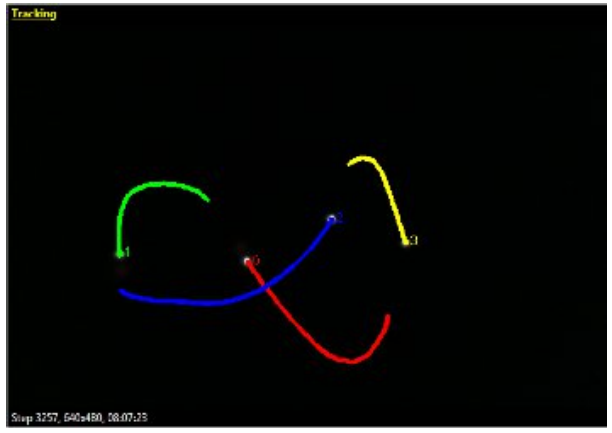
The SwisTrack software lets you easily log also the resulting data that you can then elaborate, here is an example taken from the experiment using 5 robots:



The following video shows the test done with 20, 30 and 38 Elisa-3 robots, the tracking is still good; it's important to notice that we stopped to 38 Elisa-3 robots because are the ones we have in our lab.

6.3 Position control

We developed a simple position control example that interacts with Swistrack through a TCP connection and control 4 robots simultaneously; the orientation of the robots is estimated only with the Swistrack information (delta position), future improvements will integrate odometry information. The following video shows the control of 4 robots that are driven in a *8-shape*.



All the following projects require the Elisa-3 library, for building refer to the section Multiplatform monitor.

- Horizontal position control (4 robots): the source code can be downloaded from position-control-pattern-horizontal-4-robots.zip

(Code::Blocks project).

One of the characteristics of the Elisa-3 robot is that it can move in vertical thanks to its magnetic wheels, thus we developed also a vertical position control that use accelerometer data coming from the robot to get the orientation of the robot (more precise) instead of estimating it with the Swistrack information, you can download the source code from the following link:

- Vertical position control (4 robots): position-control-pattern-vertical-4-robots.zip (Code::Blocks project).

  We developed also an example of position control that control a single robot (code adapted from previous example) that can be useful during the initial environment installation/testing; you can download the source code from the following link:

- Horizontal position control (1 robot): position-control-pattern-horizontal-1-robot.zip (Code::Blocks project).

Another good example to start playing with the tracking is an application that lets you specify interactively the target point that the robot should reach; you can download the source code of this application from the following link:

- Go to target point: position-control-goto-pos-horizontal-1-robot.zip (Code::Blocks project).

6.4 Utilities

In order to adjust the IR camera position it is useful to have an application that turn on the back IR of the robots. The following application back-IR-on-4-robots-rev245-15.01.21.zip is an example that turn on the back IR of 4 robots, their addresses are asked to the user at the execution.

7 Local communication

The advanced firmware is needed in order to use the local communication. You can find some examples on how to use this module in the main, refers to demos in selector position from 11 to 14.

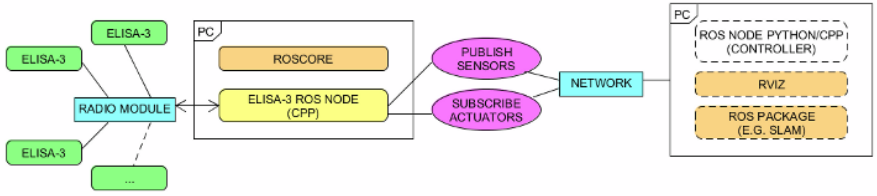
Here are some details about the current implementation of the communication module:

- use the infrared sensors to exchange data, thus during reception/transmission the proximity sensors cannot be used to avoid obstacles; in the worst case (continuous receive and transmit) the sensor update frequency is about 3 Hz
- bidirectional communication
- id and angle of the proximity sensor that received the data are available
- the throughput is about 1 bytes/sec
- maximum communication distance is about 5 cm

- no reception/transmission queue (only one byte at a time)
- the data are sent using all the sensors, cannot select a single sensor from which to send the data. The data isn't sent contemporaneously from all the sensors, but the sensors used are divided in two groups of 4 alternating sensors (to reduce consumption)

8 ROS

This chapter explains how to use ROS with the elisa-3 robots; the radio module is needed here. Basically all the sensors are exposed to ROS and you can also send commands back to the robot through ROS. The ROS node is implemented in cpp. Here is a general schema:



Click to enlarge

First of all you need to install and configure ROS, refer to https://wiki.ros.org/Distributions for more informations. Alternatively you can download directly a virtual machine pre-installed with everything you need, refer to section virtual machine; this is the preferred way.

- This tutorial is based on ROS Hydro. The same instructions are working with ROS Noetic, beware to use `noetic` instead of `hydro` when installing the packages.
- If you downloaded the pre-installed VM you can go directly to section Running the ROS node.

The ROS elisa-3 node based on roscpp can be found in the following repository https://github.com/gctronic/elisa3_node_cpp.

8.1 Initial configuration

The following steps need to be done only once after installing ROS:

1. If not already done, create a catkin workspace, refer to https://wiki.ros.org/catkin/Tutorials/create_a_workspace. Basically you need to issue the following commands:

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/src
catkin_init_workspace
cd ~/catkin_ws/
catkin_make
source devel/setup.bash
```

2. You will need to add the line `source ~/catkin_ws/devel/setup.bash` to your `.bashrc` in order to automatically have access to the ROS commands when the system is started
3. Clone the elisa-3 ROS node repo from https://github.com

/gctronic/elisa3_node_cpp inside the catkin workspace source folder (~/catkin_ws/src): `git clone https://github.com/gctronic/elisa3_node_cpp.git`

4. Install the dependencies:

ROS:

- `sudo apt-get install ros-hydro-slam-gmapping`
- `sudo apt-get install ros-hydro-imu-tools`

  If you are using a newer version of ROS, replace `hydro` with your distribution name.

cpp:

- install OpenCV: `sudo apt-get install libopencv-dev`

  If you are working with OpenCV 4, then you need to change the header include from `#include <opencv/cv.h>` to `#include <opencv2/opencv.hpp>`

5. Rebuild the `elisa-3 library`: go to `~/catkin_ws`

`/src/elisa3_node_cpp/src/pc-side-elisa3-library/linux`, then issue `make clean` and `make`

6. Open a terminal and go to the catkin workspace directory (`~/catkin_ws`) and issue the command `catkin_make`, there shouldn't be errors

7. The USB radio module by default requires root priviliges to be accessed; to let the current user have access to the radio we use `udev rules`:

- in the udev rules file you can find in `/etc/udev/rules.d/name.rules` add the following string changing the `GROUP` field with your current user group:

  ```
  SUBSYSTEMS=="usb", ATTRS{product}=="nRF24LU1P-F32 BOOT LDR", GROUP="viki"
  ```

  To know which groups your user belongs to issue the command `id`
- disconnect and reconnect the radio module

8. Program the elisa-3 robot with the last advanced firmware (>= rev.221) and put the selector in position 15

8.2 Running the ROS node

First of all get the last version of the elisa-3 ROS node from github:

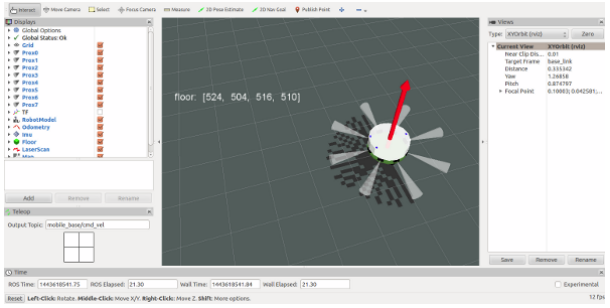
- clone the repo https://github.com/gctronic/elisa3_node_cpp and copy the `elisa3_node_cpp` directory inside the catkin workspace source folder (e.g. ~/catkin_ws/src)
- build the driver by opening a terminal and issueing the command `catkin_make` from within the catkin workspace directory (e.g. ~/catkin_ws).

  Now you can start the ROS node, for this purposes there is a launch script (based on roslaunch), as explained in the following section. Before starting the ROS node you need to start `roscore`, open another terminal tab and issue the command `roscore`.

8.2.1 Single robot

Open a terminal and issue the following command: `roslaunch elisa3_node_cpp elisa3_single.launch elisa3_address:='1234'` where `1234` is the robot id (number on the bottom).

If all is going well rviz will be opened showing the informations gathered from the topics published by the elisa ROS node as shown in the following figure:



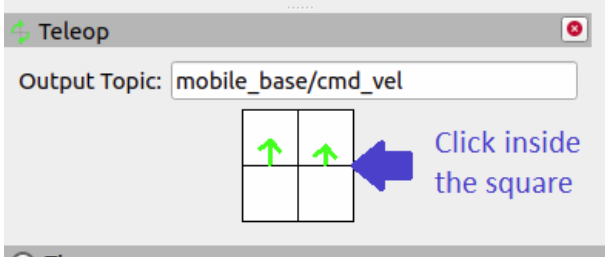
Click to enlarge

The launch script is configured also to run the gmapping (SLAM) node that let the robot construct a map of the environment; the map is visualized in real-time directly in the rviz window. Here is a video:

8.3 Move the robot

You have two options to move the robot.

The first one is to use the `rviz` interface: in the bottom left side of the interface there is a `Teleop` panel containing an *interactive square* meant to be used with differential drive robots. By clicking in this square you'll move the robot, for instance by clicking on the top-right section, then the robot will move forward-right.



Click to enlarge

The second method is by directly publishing on the `/mobile_base/cmd_vel` topic, for instance by issueing the following command `rostopic pub -1 /mobile_base/cmd_vel geometry_msgs/Twist -- '[0.0, 0.0, 0.0]' '[0.0, 0.0, 1.0]'` the robot will rotate on the spot, instead by issueing the

following command `rostopic pub -1 /mobile_base/cmd_vel geometry_msgs/Twist -- '[4.0, 0.0, 0.0]' '[0.0, 0.0, 0.0]'` the robot will move straight forward.
Beware that there shouldn't be any other node publishing on the `/mobile_base/cmd_vel` topic (e.g. Rviz), otherwise your commands will be overwritten.

8.4 Troubleshooting

8.4.1 Robot state publisher

If you get an error similar to the following when you start a node with roslaunch:

ERROR: cannot launch node of type [robot_state_publisher/state_publisher]: Cannot locate node of type [state_publisher] in package [robot_state_publisher]. Make sure file exists in package path and permission is set to executable (chmod

+x)

Then you need to change the launch file from:

<node name="elisa3_state_publisher" pkg="robot_state_publisher" type="state_publisher" />

To:

<node name="elisa3_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher" />

This is due to the fact that `state_publisher` was a deprecated alias for the node named `robot_state_publisher` (see https://github.com/ros/robot_state_publisher/pull/87).

8.5 Virtual machine

To avoid the tedious work of installing and configuring all the system we provide a virtual machine which includes all the system requirements you need to start playing with ROS and elisa. You can

download the image as *open virtualization format* from the following link ROS-Hydro-12.04.ova (based on the VM from https://nootrix.com/2014/04/virtualized-ros-hydro/); you can then use VirtualBox to import the file and automatically create the virtual machine. Some details about the system:

- user: gctronic, pw: gctronic
- Ubuntu 12.04.4 LTS (32 bits)
- ROS Hydro installed
- Webots 8.0.5 is installed (last version available for 32 bits linux)
- git-cola (git interface) is installed
- the `catkin workspace` is placed in the desktop

9 Videos

9.1 Autonomous charge

The following videos show 3 Elisa-3 robots moving around in the environment avoiding obstacles thanks to their proximity sensors and then going to the charging station autonomously; some black tape is placed in the charging positions to help the robots place themselves thanks to their ground sensors. The movement and charging is indipendent of the gravity. It works also vertically and up-side-down.

9.2 Remote control

The following video shows 38 Elisa-3 robots moving around with onboard obstacle avoidance enabled; 15 of them are running autonmously, the remaining 23 are controlled from one computer with the radio module.