KH4-EXP

User manual

Version 1.0
October 2015

Documentation Author

Frédéric Lambercy
K-Team S.A.
Z.I Les Plans-Praz 28
1337 Vallorbe
Switzerland

Email: info@k-team.com
URL: www.k-team.com

1 INTRODUCTION

The hardware of the Khepera IV is based on a modular concept. The expansion turret can be plugged on the Robot to improve the capabilities of the KheperaIV.

This turret will provide many different option to the Robot:

- LRF connection
- StarGazer connection
- Additional battery
- IOs accessibility

LRF connection:

One of the turret function is to adapt the URG-04LX-UG01 from Hokuyo on the KheperaIV. This module is a laser sensor for area scanning. The scan area is a 240° semicircle with a maximal distance of detection of 4m. The sensor retrieves one measure each 0.36° (682 steps exactly). For more information about the sensor, please look at the sensor specifications here:

http://ftp.k-team.com/KheperaIII/LRF/URG-04LX_UG01_spec.pdf

This extension can be purchased with or without the URG-04LX sensor. In case you would like to purchase it by your own way, look at the chapter 4.3 to know how to mount the sensor on the turret.

StarGazer connection:

The second option of the turret is to mount a StarGazer positioning sensor from Hagisonic. This module give a global indoor positioning and bearing solution for the KheperaIV.

This is possible to add both the LRF and the StarGazer on the same KheperaIV. Do not hesitate to ask K-Team for a special configuration.

Additional battery:

An additional battery can be plugged on the turret to improve the whole autonomy of the Robot. This battery is swappable to allow continuous operation. The standard pack include two batteries and one external charger to allow this swap.

This additional battery can be also used with the LRF and/or the StarGazer. In case of an intensive use (LRF, StarGazer and Gripper), it is more than advise to make sure you have more than two batteries to ensure a continuous operation (as soon as the charging time is longer than the discharge time).

IOs accessibility:

The turret give an easy access to several IOs of the KB-250 bus. See section 10 for more details about these connections.

1.1 How to use this manual

This manual introduces the Khepera IV Expansion turret. To learn how to make the best use of your turret, you are urged to read all the chapters 3 through 10.

If this manual does not answer one of the problems you are confronted with, please consult the K-Team web site (www.k-team.com) and especially the Forum and the FAQs.

Introduction: Presentation of the LRF and the way to use it.
Unpacking and Inspection: LRF's package description and first start-up.
The LRF sensor: Description of all the LRF's functionalities.
Connections: Explanation on how to connect (or disconnect) the LRF to the robot.
Programming the LRF: Instructions to program the LRF using the libkorebot.
Warranty: Legal notice of the LRF warranty.

1.2 Safety precautions

Here are some recommendations on how to correctly use the Khepera IV LRF:

- **Keep the turret away from wet area**. Contact with water could cause malfunction and/or breakdown.
- **Store your turret in a stable position**. This will avoid the risks of falls, which could break it or cause damage to a person.
- **Use only the official charger which is delivered with the Khepera IV robot.** Do not try to use another charger; this can cause irreversible damage to the battery.
- **Do not plug or remove the turret while the robot is powered on**. To avoid any damage, make all connections when the robot power is off.
- **Never leave the Khepera IV and the turret powered when it is unused**. When you have finished working with Khepera IV, turn it off. It will save the battery life.

1.3 Recycling

Think about the end of life of your robot! Parts of the robot can be recycled and it is important to do so. It is for instance important to keep batteries out of the solid waste stream. When you throw away a battery, it eventually ends up in a landfill or municipal incinerator. These batteries, which contain Lithium Polymer, can contribute to the toxicity levels of landfills or incinerator ash. By recycling the batteries through recycling programs, you can help to create a cleaner and safer environment for generations to come. For those reasons please take care to the recycling of your robot at the end of its life cycle, for instance sending back the robot to the manufacturer or to your local dealer.

Thanks for your contribution to a cleaner environment!

2 Expansion turret available configuration

The KheperaIV expansion turret can be used for many purpose. The main application is to connect sensor like LRF and Stargazer. But this turret can be also use to extend the autonomy of the Robot with an additional battery.

The Additional battery usage is explain in chapter 3.

The LRF usage is explain in chapter 4 to 6.

Finally, the Stargazer configuration is explain in chapter 7 to 9.

Some GPIOs from the KheperaIV are made available on this turret and are explain in chapter 10.

Note that these configuration can used together on the same robot at the same time. The configuration with StarGazer and LRF is explain in chapter 11.

Do not hesitate to ask K-Team for special configuration.

3 ADDITIONAL BATTERY

3.1 Unpacking and inspection

If you have choose the additional battery as an option with the expansion turret, you must find the following component:

- the Khepera IV Expansion Turret
- 2x Battery pack
- External charger
- External power supply

3.2 Overview

3.2.1 Battery Pack

Figure 2.1 : Battery Pack overview

3.2.2 External charger

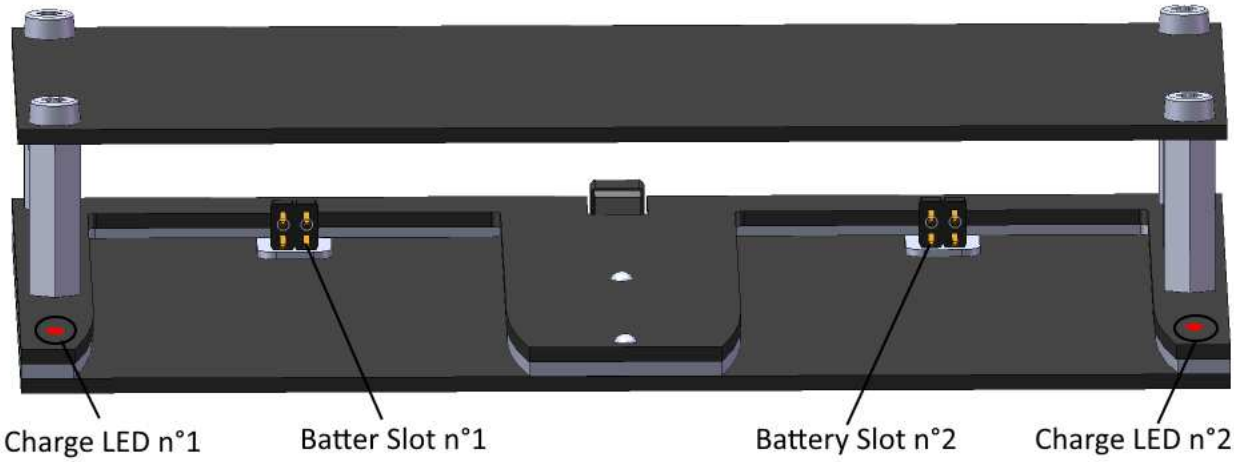


Figure 2.2 : External charger overview (front view)

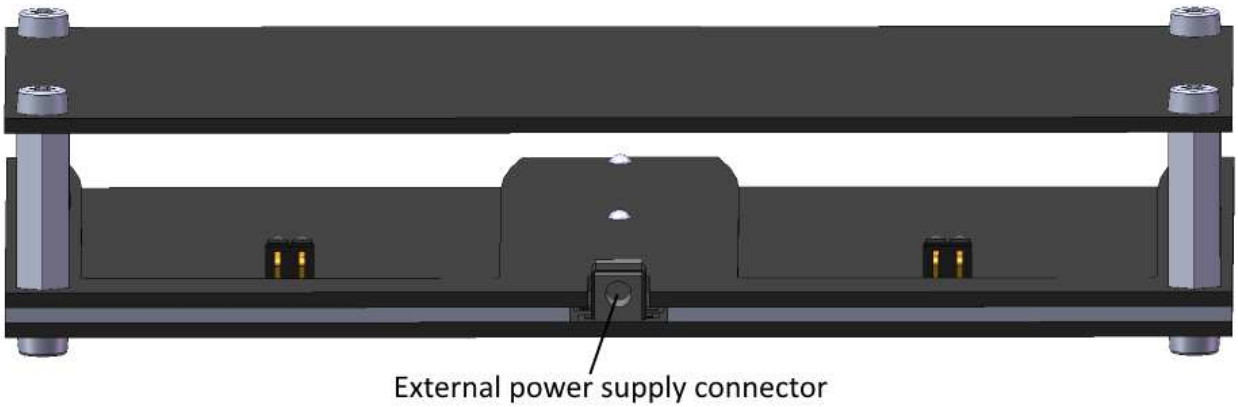


Figure 2.2 : External charger overview (rear view)

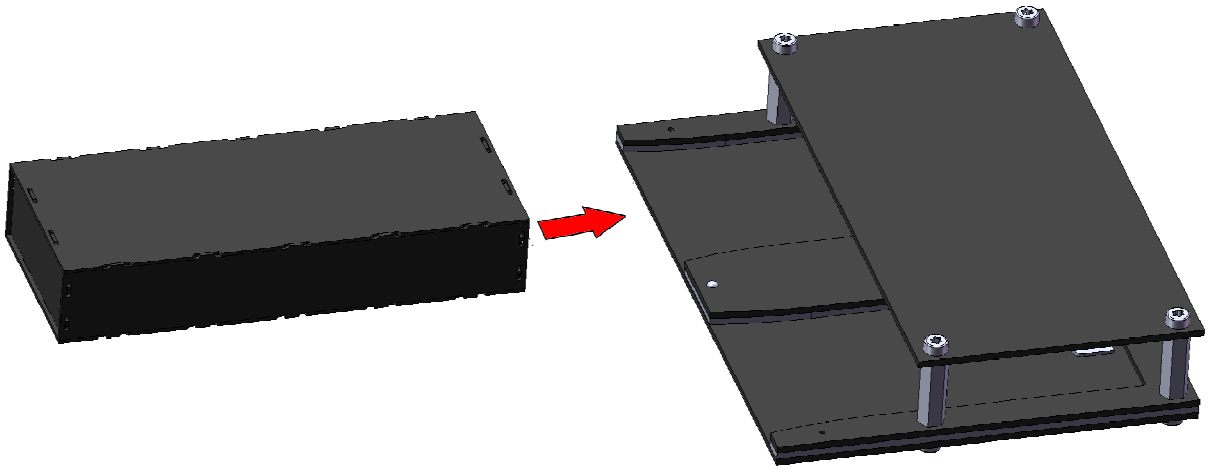


Figure 2.3 : Battery pack insertion in the charger

2.2.3 Turret connection

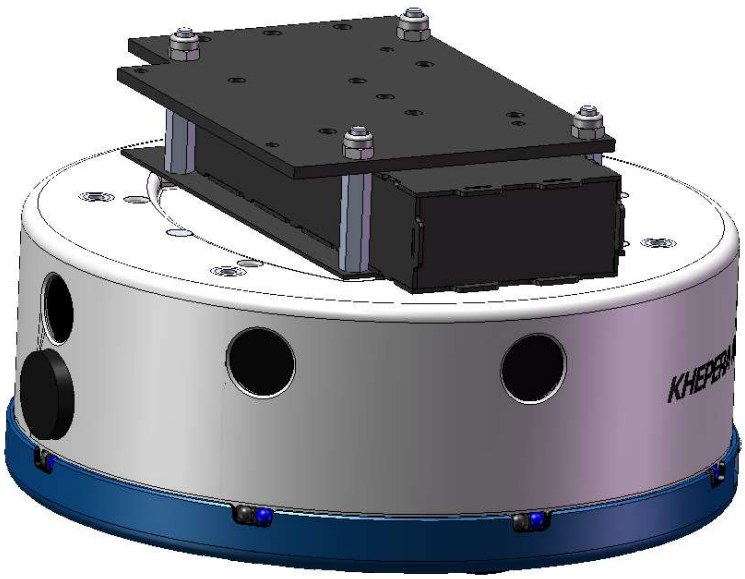


Figure 2.4 : Additionnal battery plugged on the KheperaIV

3.3 Battery pack specification

Dimensions:	107x44.2x19.5mm
Weight:	125g
Nominal Capacity:	3000mAh
Nominal Voltage:	7.4V
Technology:	Li-Pol

3.4 Battery pack assembly

The battery pack can be connected on the expansion turret whenever if the Robot is power ON or OFF. This battery support the hot plug.

First, place the battery in front of its slot with the battery connector head low (see Figure 2.5):

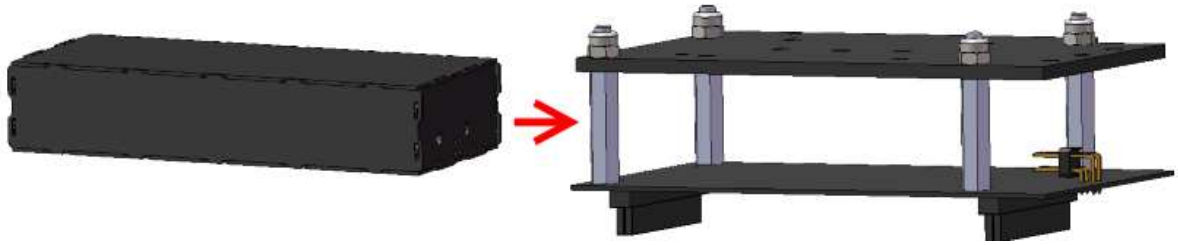


Figure 2.5 : insertion of the battery pack

Then gently insert the battery between the spacer until the battery connector is plugged into the pin of the Expansion turret.

!!! Do not force if the connection is not established easily, verify the orientation of the battery!!!

Once the battery is correctly plugged (and the robot is ON) the green LED on the expansion turret will turn ON meaning the whole power supply of the robot is powered by the battery.

3.5 Usage

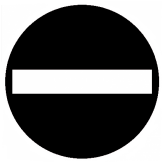
The battery pack was design to be used only with the Expansion Turret. Once the battery connected on the Turret, all the Robot supply will be powered by this battery. In fact, this turret will generate a supply to emulate an external charger, which means that even the charge of the KheperaIV will be active once this turret plugged. Anyway, as this battery has a smaller capacity than the one on the Robot, it cannot supply the Robot and charge completely the KheperaIV battery.

The main use of this additional battery is too improve the autonomy of the Robot and allow hot swap. Which means that, before using this additional battery, you have to charge completely the battery of the KheperaIV first.

Once the additional battery is completely empty, the robot's battery will take over. During this period, you will be able to swap the battery on the turret without stopping your application.

The Expansion turret is equipped with an on board ADC connected to the I2C bus (ADS1015 from Ti). This Analog to Digital Converter allows to measure the Voltage and the Current consumption of the battery.

The battery pack will be completely shut down once the robot is turn OFF. Which means that the battery don't need to be removed once the experimentation is over.



Do not connect the power supply on the KheperaIV when the additional battery is used.

3.6 Charging the battery

The additional battery can be charged only outside the turret. The battery pack need to be removed from its slot, then connected to the external charger.

To connect the battery, insert it inside one of the two slot. The Battery connector must be direct to the low (same direction as the insertion in the Expansion turret).

You can charge up to two battery at the same time on the external charger.

The external charger must be used only with the external power supply provide with the KheperaIV or the one deliver with the charger.

During the charge, the LED on the side of the charger will turn Red (charge LED n°1 for slot n°1, LED n°2 for the slot n°2). Once the charge is complete (~3.5h when battery is completely empty), the LED will turn off.

3.7 Estimation of the added autonomy

Using the additional battery pack to supply a KheperaIV will add a significant autonomy to the Robot. Here's an evaluation of the gain following the configuration (the robot battery need to be fully charged before using this extension to obtain these values):

- KheperaIV alone: ~5.5h
- With LRF: ~3.5h
- With StarGazer: ~3h
- With LRF+StarGazer: ~2.5h
- With Gripper: ~3.5h
- With Gripper+LRF+StarGazer: ~1.5h

3.8 Programming the Battery pack

The Expansion turret is equipped with an ADS1015 ADC converter from Ti which can be read through I2C. A dedicated library has been created especially for this IC in the Libkhepera.1.1. You can d

http://ftp.k-team.com/

If your Khepera IV has already the latest libkhepera installed, jump to section 2.8.2

You can check if the last version is installed:

- Log on the Khepera IV (via ssh, Bluetooth or serial port)
- You can check if you have the 1.1 version in listing the present files:
    - ls –s /usr/lib/libkhepera*
- This should give (end of the two lines):
    - /usr/lib/libkhepera.so -> /usr/lib/libkhepera.so.1.1
    - /usr/lib/libkhepera.so.1.1

3.8.1 Installation of the latest libkhepera and config. file

To be able to read the battery voltage and current with the Khepera IV, it's necessary to install the libkhepera version 1.1 or greater on the Khepera IV.

Normally, if you have received the Khepera IV at the same time than the Expansion Turret, the Khepera IV is ready to be used with the expansion turret; in this case, you can jump to section 2.8.2. Otherwise, if you have bought the turret separately, you will need to execute the step described below:

- Remove the old libkorebot: ***rm /usr/lib/libkhepera****
- Copy or send the ***libkhepera.so.1.1*** file (locate in the ***build-khepera-2.6/lib/*** directory of the libkhepera source files) to the directory ***/usr/lib***.
- Link the libkhepera file as ***libkhepera.so****:*
  - ln -s /usr/lib/libkhepera.so.1.1 /usr/lib/libkhepera.so
- Copy the ***kbattery.knc*** configuration file in the /etc/libkhepera directory

3.8.2 Using the turret with kbattery_test software

Before starting programming to read the battery value, it is important to test the Expansion turret with the ***kbattery_test*** software. If the software is not yet in the */home/root* directory of the Khepera IV, copy it (the executable file is located in the *src/tests/* directory of the *libkhepera*). Then execute it with the command: ***./kbattery_test****.*

The ***kbattery_test*** program waits that the user enters a command to control the lrf. To show all the available commands, type ***help*** and push the "RETURN" key. These commands are described below.

Warning: this software does not support the backspace command. If you have typed a wrong command, push the "RETURN" key and restart typing the correct command. All wrong commands will be return an error message. The parameters of the command are separated by a space, but some commands do not need any parameter. In this case, type only the command and finally push the "RETURN" key.

3.8.3 kbattery_test commands

config

Effect: Configure the ADC to read either the Voltage (param = 1) or the current (param = 0).

Parameter:

- *0 Configure the ADS1015 to read the current*
- *1 Configure the ADS1015 to read the Voltage*

getvolt

Effect: Read continuously the battery voltage. Exit only when a new key is detected. This command will automatically call the "config 1" command before reading repetitively the Voltage.

Parameter: -

getcur

Effect: Read continuously the battery Current. Exit only when a new key is detected. This command will automatically call the "config 0" command.

Parameter: -

getboth

Effect: Read continuously the battery voltage and current. Exit only when a new key is detected. With this command, the system will use the single shot measure to read alternatively the Voltage and the Current (the both other command used the automatic read mode).

Parameter: -

3.8.4 Compiling your own software

To make your own software to read the additional battery value, you need to install the development tools for the Khepera IV on your computer (with Linux OS). If it is not already done, please look at the Khepera IV user manual to know how to install it. Once done, you can start writing your program. The best solution is to copy the kbattery_test.c source code and modify it. In any case, keep your program in the src/tests/ directory, and modify the Makefile to compile your new software: add your new program filename in line 27 of the Makefile in the src/test directory or start with the template program in the template directory.

As for all the Khepera IV extensions, you need to initialize the library before using the functions described below.

The libkhepera provides high-level functions which include all the available capabilities of the ADC used in the Expansion turret. Look at the section 2.8.4.1 to view the different functions provided by the libkhepera.

3.8.4.1 Constants

Constant defined in the header file is described below:

KBAT_RESULT

I2C addr of the result register

```
#define KBAT_RESULT   0x00
```

KBAT_CONFIG

I2C addr of the configuration register

```
#define KBAT_CONFIG   0x01
```

KBAT_LOW_THRES

I2C addr of the low threshold register

```
#define KBAT_LOW_THRES   0x02
```

KBAT_HIGH_THRES

I2C addr of the high threshold register

```
#define KBAT_HIGH_THRES   0x03
```

KBAT_CONFIG_VOLTAGE_AUTO

Value to configure the ADS1015 to read automatically the Voltage.

```
#define KBAT_CONFIG_VOLTAGE_AUTO   0x0483
```

KBAT_CONFIG_VOLTAGE_SINGLE

Value to configure the ADS1015 to read the Voltage in single shot mode.

```
#define KBAT_CONFIG_VOLTAGE_SINGLE   0x0583
```

KBAT_CONFIG_CURRENT_AUTO

Value to configure the ADS1015 to read automatically the Current.

```
#define KBAT_CONFIG_CURRENT_AUTO   0x3483
```

KBAT_CONFIG_CURRENT_SINGLE

Value to configure the ADS1015 to read the Current in single shot mode.

```
#define KBAT_CONFIG_CURRENT_SINGLE   0x3583
```

KBAT_CONFIG_START_MEAS

Mask to start a measure when ADS1015 is configure in single shot mode

#define KBAT_CONFIG_START_MEAS 0x8000

KBAT_CONVERT_VOLTAGE_MULT

Constant to convert the register value to a mV value (need to divide with the *KBAT_CONVERT_VOLTAGE_DIV too)*

#define KBAT_CONVERT_VOLTAGE_MUL 32

KBAT_CONVERT_VOLTAGE_DIV

Constant to convert the register value to a mV value (need to multiply with the *KBAT_CONVERT_VOLTAGE_MULT too)*

#define KBAT_CONVERT_VOLTAGE_DIV 7

*U[mV] = Regvalue * KBAT_CONVERT_VOLTAGE_MULT / KBAT_CONVERT_VOLTAGE_DIV*

KBAT_CONVERT_CURRENT_MULT

Constant to convert the register value to a mA value (need to divide with the *KBAT_CONVERT_CURRENT_DIV too)*

#define KBAT_CONVERT_CURRENT_MULT 250

KBAT_CONVERT_CURRENT_DIV

Constant to convert the register value to a mA value (need to multiply with the *KBAT_CONVERT_CURRENT_MULT too)*

#define KBAT_CONVERT_CURRENT_DIV 97

*I[mA] = Regvalue * KBAT_CONVERT_ CURRENT _MULT / KBAT_ CONVERT_CURRENT_DIV*

3.8.4.2 High-level functions

The **green** words indicate the type of the function (or parameter) and must not be included in your code.
The **blue** words are the parameters. You must declare it in your code before calling the function.
Only the ***black*** words can be directly added to your code.

Note: ***KBAT_socket*** *is the Value handle received by opening the socket with* ***knet_open("Kbattery:ADC" , KNET_BUS_I2C , 0 , NULL);***

int *kbattery_init (void)*

Initializes the library to use the different battery functions. This function must be called at the beginning of each program using the additional battery.

Return: a handle number if success or <0 if error
Parameter: **-**

void *kbattery_Config_Voltage (knet_dev_t *KBAT_socket)*

Configure the ADS1015 to read automatically the voltage.

Return: -

void *kbattery_Config_Current(knet_dev_t *KBAT_socket)*

Configure the ADS1015 to read automatically the current.

Return: -

Unsigned short *kbattery_Get_Battery_Voltage (knet_dev_t *KBAT_socket)*

Read the value of the battery voltage. The ADS1015 must configure to read automatically the voltage to have a correct value

Return: Voltage of the battery in [mV]

Unsigned short *kbattery_Get_Battery_Current (knet_dev_t *KBAT_socket)*

Read the value of the battery voltage. The ADS1015 must configure to read automatically the current to have a correct value

Return: Current of the battery in [mA]

Unsigned short
kbattery_Get_Battery_Voltage_Single_Shot(knet_dev_t *KBAT_socket)

Read the value of the battery voltage in single shot mode (will perform only one conversion).

Return: Voltage of the battery in [mV]

Unsigned short
kbattery_Get_Battery_Current_Single_Shot(knet_dev_t *KBAT_socket)

Read the value of the battery current in single shot mode (will perform only one conversion).

Return: Current of the battery in [mA]

4 THE LRF MODULE

4.1 Unpacking and inspection

First check that you have a complete package. You should find:

- the Khepera IV Expansion Turret
- mini-USB to Micro-Match cable
- URG-04LX-UG01 sensor (optional)

4.2 Global View

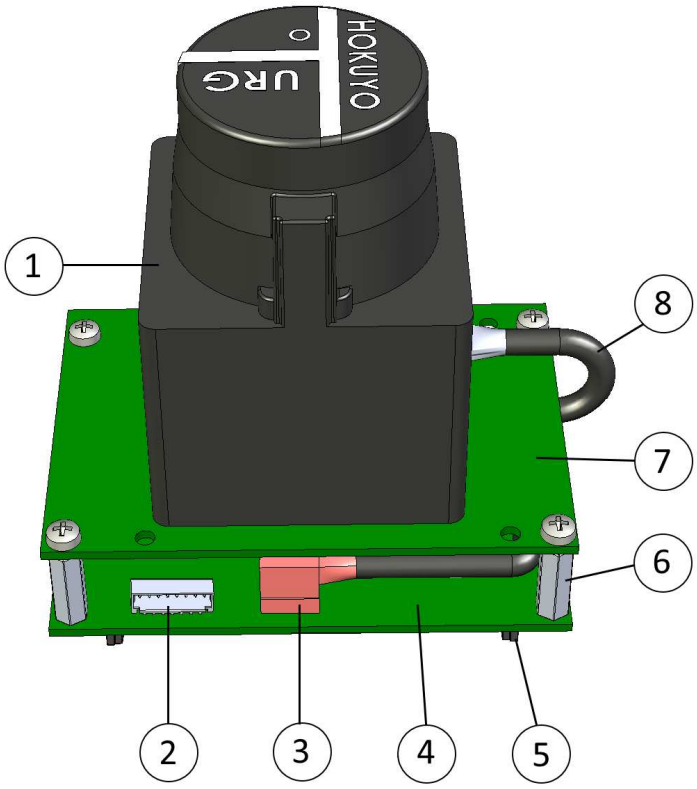


Figure 1: Overview of the Khepera IV LRF

1 URG-04LX-UG01 sensor
2 Serial connector (not used)
3 LRF connector
4 Main PCB
5 KB250 extension connector
6 Mechanical spacer
7 Fixing PCB
8 Mini-USB cable

4.3 LRF sensor assembly

If you have purchased the LRF extension without the URG-04LX sensor, here's the step to follow:

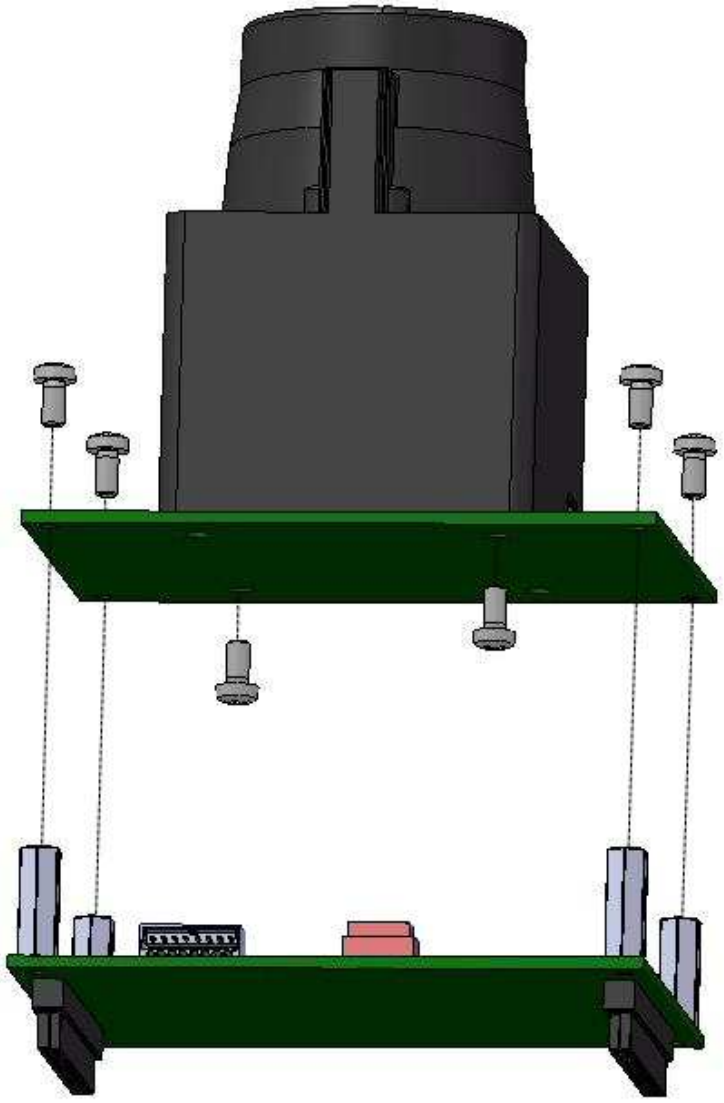


Figure 2: LRF assembly

- unscrew the four screws on the top of the turret to unmount the fixing PCB.
- mount the LRF on the TOP of the PCB (face where the indications FRONT & BACK are visible).
- respect the orientation of the sensor as shown in the picture above.

- use the two holes indicated in the picture below to screw the sensor.

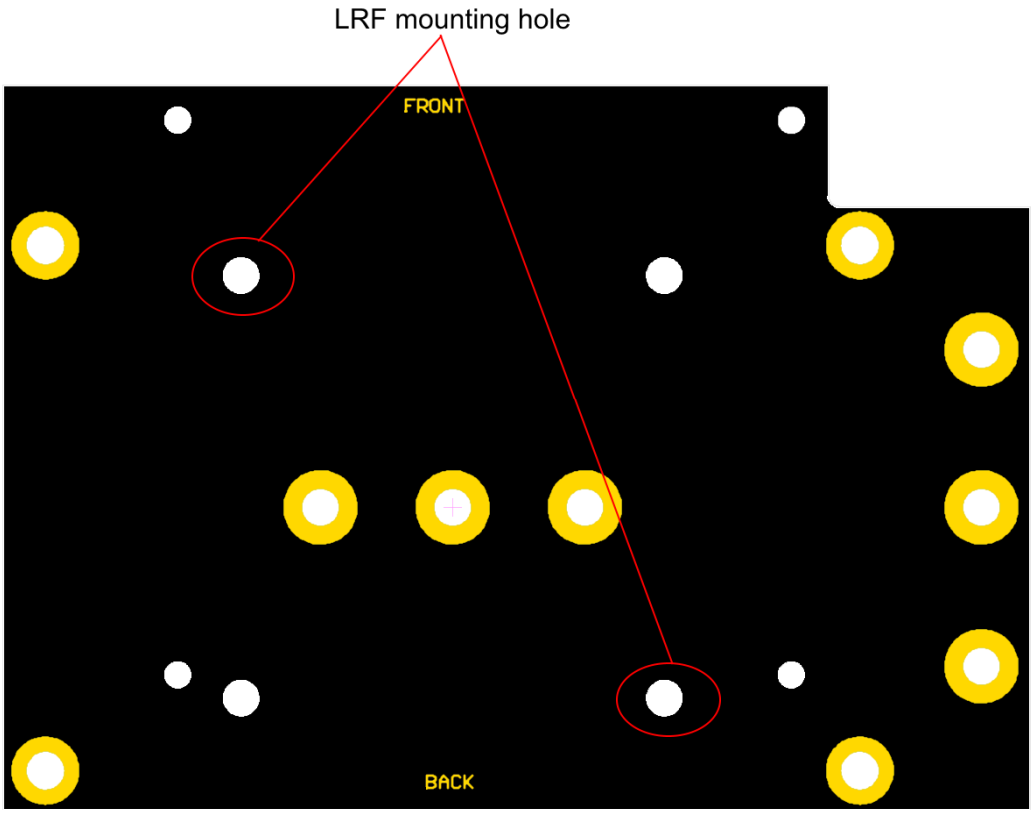


Figure 3: Mounting holes for the Hokuyo sensor

- mount the fixing PCB with the sensor on the Main PCB of the turret and screw it.
- connect the mini-USB/Micro-Match cable between the sensor and the Main PCB.
- You can now mount your fully assembled turret on a Khepera IV to use it.

4.4 URG-04LX-UG01 sensor specifications

Light source	Semiconductor laser diode (λ=785nm),
Laser safety	Class 1 (IEC60825-1)
Power source	5V DC ±5% (USB bus power)
Current consumption	500mA or less (Rush current 800mA)
Detection distance	20mm ~ 4000mm
Accuracy	Distance 20mm ~ 1000mm : ±30mm Distance 20mm ~ 4000mm : ±3% of measurement
Resolution	1 mm
Scan Angle	240°
Angular Resolution	0.36°
Scan Time	100msec/scan
Ambient (Temperature/Humidity)	-10 ~ 50ºC / 85% or less (without dew and frost)
Preservation temperature	-25 ~ 75ºC
Ambient Light Resistance	10000Lx or less
Impact Resistance	196 m/s2, 10 times each in X, Y and Z direction
Insulation Resistance	10MΩ for DC 500Vmegger
Weight Approx.	160 g
Case	Polycarbonate

For more information, please have a look at the sensor datasheet:

http://ftp.k-team.com/KheperaIII/LRF/URG-04LX_UG01_spec.pdf

5 LRF CONNECTIONS

Assembling and disassembling additional turrets is a delicate operation. Try to avoid it as much as possible and perform it carefully. Please follow the instructions below to avoid damage to your modules. K-Team can assume no responsibility for any damage caused by improper manipulation.

5.1 Assembling

Assembling is an easy operation, but it is also necessary to perform it carefully:

- Be sure that the Khepera IV is turned off.
- Insert the LRF on the Khepera IV (the connectors of the Main PCB must be at the back of the robot). Do not try to mount the LRF the other way round; this can cause irreversible damages to the KB-250 bus connectors.
- When the LRF is correctly engaged in the KB-250 bus connectors, push the turret straight to plug it. If it is too hard to plug the turret, do not force on it. The connectors are certainly not correctly aligned.

5.2 Disassembling

This operation must be done very carefully and as infrequently as possible:

- First, switch off the Khepera IV
- Take the LRF turret with one hand and maintain the Khepera IV with the other. **Do not pull on the URG-04LX sensor to unplug the LRF. Place your fingers on the Main PCB**.
- Pull the LRF straight and very carefully. Once unplugged, place the LRF in its case to store it.

5.3 Unpacking test

After unpacking it is important to test the functionality of the LRF:

- Plug the LRF module on a Khepera IV
- Connect the Khepera IV to a computer (see Khepera IV user manual).
- Open a terminal on your computer and turn the robot on.
- Once the login passed, run the ***klrf_test*** program by typing ***./klrf_test***. If it is not yet installed on your Khepera IV, follow the steps in the section 5.1.
- Type the command ***lrfinit*** and push the RETURN key.
- It should return the model, the motor speed and the connected port as follows:
    - model: URG-04LX(Hokuyo Automatic Co.,Ltd.)
    - scan_rpm: 600
    - URG is detected, port /dev/ttyACM0
- Type the command ***lrfmeasure 0*** and push the RETURN key.
    - It should display continuously LRF data in a x-y graphic and you should recognize the environment shape around the LRF module.
- Push any key then type the command ***exit*** and push the RETURN key. This will close the program.

If the LRF does not correctly perform this sequence of actions, please contact your local dealer.

More commands are described in the section 6.2.

6 PROGRAMMING THE LRF

The LRF is an extension that can be used only once installed on the Khepera IV. That means that the Khepera IV controls all the functions of the LRF. As all the Khepera IV extensions, a library including all the available functions is provided with the libkhepera version 1.0 or greater. If you already have a libkhepera installed in your computer but with an older version, you can download the latest version on our FTP:

http://ftp.k-team.com/

If your Khepera IV has already the latest libkhepera installed, jump to section 5.2.

You can check if the last version is installed:

- Log on the Khepera IV (via ssh, Bluetooth or serial port)
- You can check if you have the 1.0 version in listing the present files:
  - ls –s /usr/lib/libkhepera*
- This should give (end of the two lines):
  - /usr/lib/libkhepera.so -> /usr/lib/libkhepera.so.1.0
  - /usr/lib/libkhepera.so.1.0

6.1 Installation of the latest libkhepera and configuration file

To use the LRF with the Khepera IV, it's necessary to install the libkhepera version 1.0 or greater on the Khepera IV.

Normally, if you have received the Khepera IV at the same time than the LRF, the Khepera IV is ready to be used with the LRF; in this case, you can jump to section 5.2. Otherwise, if you have bought the LRF separately, you will need to execute the step described below:

- Log on the Khepera IV (via ssh, Bluetooth or serial port)
- You have different ways to update the library:
- Autonomously with the package:
  - Upload the package file ***libkhepera-1.0-r0_armv5te.ipk*** by ssh, Bluetooth or Serial to the Korebot II.
- Remove the old one with the command: ***ipkg remove libkhepera***
- Install the new one with the command:
  - ipkg install libkhepera-1.0-r0_armv5te.ipk

By hand:

- Remove the old libkhepera: ***rm /usr/lib/libkhepera****
- Copy or send the ***libkhepera.so.1.0*** file (locate in the ***build-khepera-2.6/lib/*** directory of the libkhepera source files) to the directory ***/usr/lib***.
- Link the libkhepera file as ***libkhepera.so****:*
    - ln -s /usr/lib/libkhepera.so.1.0 /usr/lib/libkhepera.so

6.2 Using the LRF with klrf_test software

Before starting programming the LRF, it is important to test the LRF with the ***klrf_test*** software to understand the different functionalities and capabilities of the LRF. If the software is not yet in the */home/root* directory of the Khepera IV, copy it (the executable file is located in the *src/tests/* directory of the *libkhepera*). Then execute it with the command: ***./klrf_test***.

The ***klrf_test*** program waits that the user enters a command to control the lrf. To show all the available commands, type ***help*** and push the "RETURN" key. These commands are described below.

Execute ***lrfinit*** : a message appears as in the unpacking test chapter 3.1. If an error appears return to section 6.1 and try to install a new version of the libkhepera library.

Warning: this software does not support the backspace command. If you have typed a wrong command, push the "RETURN" key and restart typing the correct command. All wrong commands will be return an error message. The parameters of the command are separated by a space, but some commands do not need any parameter. In this case, type only the command and finally push the "RETURN" key.

6.2.1 klrf_test commands

poweron

Effect: not used. The LRF turn on automatically.
Parameter: -

lrfinit

Effect: Initializes the LRF communication and turns the laser beam on.
Parameter: -

laseron

Effect: sets the laser beam on. The LRF green led becomes steady on.
Parameter: -

laseroff

Effect: sets the laser beam off (module is still active and consumes battery). The LRF green led starts blinking.
Parameter: -

poweroff

Effect: Not used. The power supply is always active.
Parameter: -

setfile

Effect: sets the filename where the output of the LRF is saved. If not set, no data will be saved. File format is text. First and second lines are the header. Next are the data, with tab between fields: current measurement index, distance index in the current measurement, angle, distance, x position, y position. See figure 4.3 in chapter 6.3 for more details.

Parameter: name of the file in a char-array format

lrfmeasure

Effect: gets LRF data, saves raw data to file if specified by the ***setfile*** command and plot processed data in a x-y graphic. Push any key to stop or it stops after the number of measurements taken from the first parameter. During the measurement the LRF led is green and not blinking.

Parameter: first: number of measurements to take (0 = runs forever)
second (optional): number for averaging each measurement.

quit , ***exit*** or ***bye***

Effect: closes the software and return to the Khepera IV main prompt.
Parameter: -

help

Effect: displays the list of commands above.
Parameter: -

6.3 Compiling your own program using the libkorebot

To make your own software to control the LRF, you need to install the development tools for the Khepera IV on your computer (with Linux OS). If it is not already done, please look at the Khepera IV user manual to know how to install it. Once done, you can start writing your program. The best solution is to copy the ***klrf_small_ex.c*** source code (see chapter 0) and modify it. The source code ***klrf_test.c*** is a more complete example. In any case, keep your program in the ***src/tests/*** directory, and modify the Makefile to compile your new software: add your new program filename in line 27 of the Makefile in the *src/test* directory or start with the template program in the *template* directory.

As for all the Khepera IV extensions, you need to initialize the library before using the functions described below.

The libkhepera provides high-level functions which include all the available capabilities of the LRF. Look at the section 4.3.1 to view the different functions provided by the libkhepera.

6.3.1 Constants

Constant defined in the header file is described below:

LRF_DATA_NB

Number of data in one measurement defined like this:

#define LRF_DATA_NB 682

6.3.2 Variables

Variable for transferring data that is already declared in the header file is described below:

kb_lrf_DistanceData

Array of long containing measurements of one scan in [mm], declared like this:

long kb_lrf_DistanceData [LRF_DATA_NB]

You can transform from Polar to Cartesian coordinates systems, with the LRF module at the origin by looking at figure 5.3. Values between 0 and 19 are error codes

(see http://www.hokuyo-aut.jp/02sensor/07scanner/download/data/URG_SCIP20.pdf, table 3 "ERROR CODES").

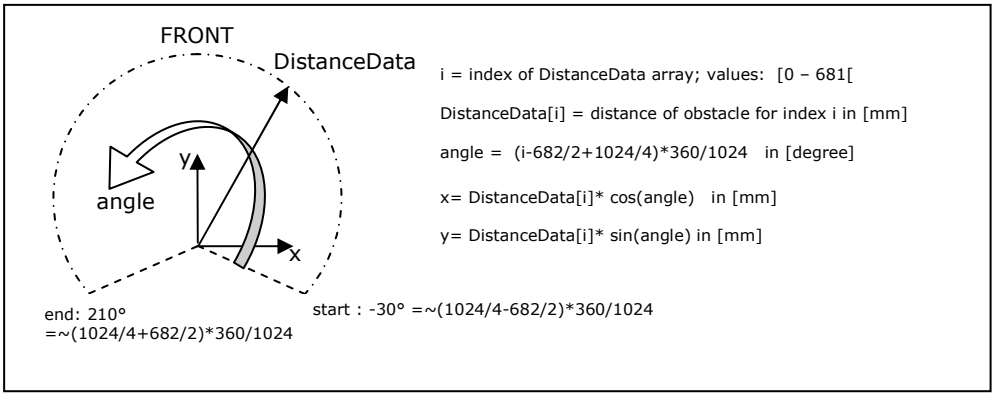


Figure 5.3 : Data description

6.3.3 High-level functions of the libkorebot

The **green** words indicate the type of the function (or parameter) and must not be included in your code.

The **blue** words are the parameters. You must declare it in your code before calling the function.

Only the ***black*** words can be directly added to your code.

int *kb_lrf_Init(char *LRF_DeviceName)*

Initializes the library to use the different LRF functions and power the laser on. This function must be called at the beginning of each program using the LRF.

Return: a handle number if success or <0 if error

Parameter: ****LRF_DeviceName:*** name of the device where the lrf is connected. Should be "/dev/ttyACM0".

int *kb_lrf_GetDistances(int LRF_DeviceHandle)*

Gets one set of radius distances between objects and the center of the LRF.

Return: a handle if success or <0 if error
Parameter: ***LRF_DeviceHandle*:** Value handle received by ***kb_lrf_Init*** function.

int *kb_lrf_GetDistances_Averaged(int LRF_DeviceHandle , int average)*

Gets multiple set of radius distances between objects and the center of the LRF averaged data.

Return: a handle if success or <0 if error
Parameters: ***LRF_DeviceHandle*:** Value handle received by ***kb_lrf_Init*** function.
average: number of times the measure is averaged.

int *kb_lrf_Close(int LRF_DeviceHandle)*

It powers off the LRF and closes the port device.

Return: -
Parameter: ***LRF_DeviceHandle*:** Value handle received by ***kb_lrf_Init*** function.

void *kb_lrf_Laser_On(int LRF_DeviceHandle)*

Sets the LRF laser beam on.

Return: -
Parameter: ***LRF_DeviceHandle*:** Value handle received by ***kb_lrf_Init*** function.

void *kb_lrf_Laser_Off(int LRF_DeviceHandle)*

Sets the LRF laser beam off (module still initialized and motor keeps running).

Return: -
Parameter: ***LRF_DeviceHandle*:** Value handle received by ***kb_lrf_Init*** function.

long *kb_lrf_Get_Timestamp (void)*

Gets the timestamp of the last measurement set.

Return: Return the timestamp of the last measurement set in [ms].
Parameter: -

6.4 Software example

A small example of software controlling the LRF is enclosed below. You can find its source code file in ***libkhepera-VERSION/src/tests/krlf_small_ex.c***.

```
#include <math.h>
#include <khepera/khepera.h>

// device where the LRF is connected: here USB port
#define LRF_DEVICE "/dev/ttyACM0"

int main( int argc , char * argv[] )
{
  int rc,i;
  int LRF_DeviceHandle; // serial port handle for lrf

  float angle,x,y;

  /* reset the screen */
  printf("\033[2J");           /* erase the whole console */
  printf("\033[0;0f");         /* Move cursor to the top left */

  printf("Led Range Finder Small Example Program  (C) K-Team S.A\r\n");

  /* Set the libkhepera debug level - Highly recommended for development. */
  kb_set_debug_level(2);

 // initialise the libkhepera
  if((rc = kb_init( argc , argv )) < 0 )
  {
      printf("\nERROR: port %s could not initialise libkorebot!\n");
      return -1;
  }

   kb_lrf_Power_On(); // activate the power supply battery module

   // initialise LRF device
   if ((LRF_DeviceHandle = kb_lrf_Init(LRF_DEVICE))<0)
  {
     printf("\nERROR: port %s could not initialise LRF!\n");
     return -2;
  }
  // get distances
  if (kb_lrf_GetDistances(LRF_DeviceHandle)<0)
  {
     printf("\nERROR: port %s could not initialise LRF!\n");
     kb_lrf_Close(LRF_DeviceHandle);
     return -3;
  }
  printf("index dist[mm] angle[deg] x[mm] y[mm]\n");

  // process distances:
  // You have the distances radii from the center of the robot in [mm],
  // starting at -30 deg and rotating to counterclockwise direction
  // inside kb_lrf_DistanceData array. Values < 20 are errors.
  // You can get the distances average with function kb_lrf_GetDistances(lrfHandle,average).
  for (i=0;i<LRF_DATA_NB;i++)
  {
      angle= (i- LRF_DATA_NB /2+1024/4)*360.0/1024.0; // angle of each data

      // convert from polar to cartesian
      x=kb_lrf_DistanceData[i]*cos(angle*M_PI/180.0); // direction: right side of robot
      y=kb_lrf_DistanceData[i]*sin(angle*M_PI/180.0); // direction: front of robot
      printf("%3d\t%4ld\t %+6.1f \t%+7.1f \t%+7.1f\n",i,kb_lrf_DistanceData[i],angle,x,y);
  }
   // close the lrf device
   kb_lrf_Close(LRF_DeviceHandle);
   return 0;
}
```

7 THE STARGAZER MODULE

7.1 Package Contents

Figure 6.1: View of the StarGazer module

Your package should contain the following items:

1. KSG board
2. DVD with software and this User Manual*

* Updates can be found at http://www.k-team.com/

7.2 Inspection

First, check that you have a complete package. You should find:

- the KSG module (mounted on the Kh4-EXP turret)

Then, check that the wires connecting the Stargazer to the expansion turret are connected and not in bad shape.

8 Description

8.1 Overview

An overview of the KSG hardware is depicted in the Figure 7.1. The locations of various key elements are indicated for later references.

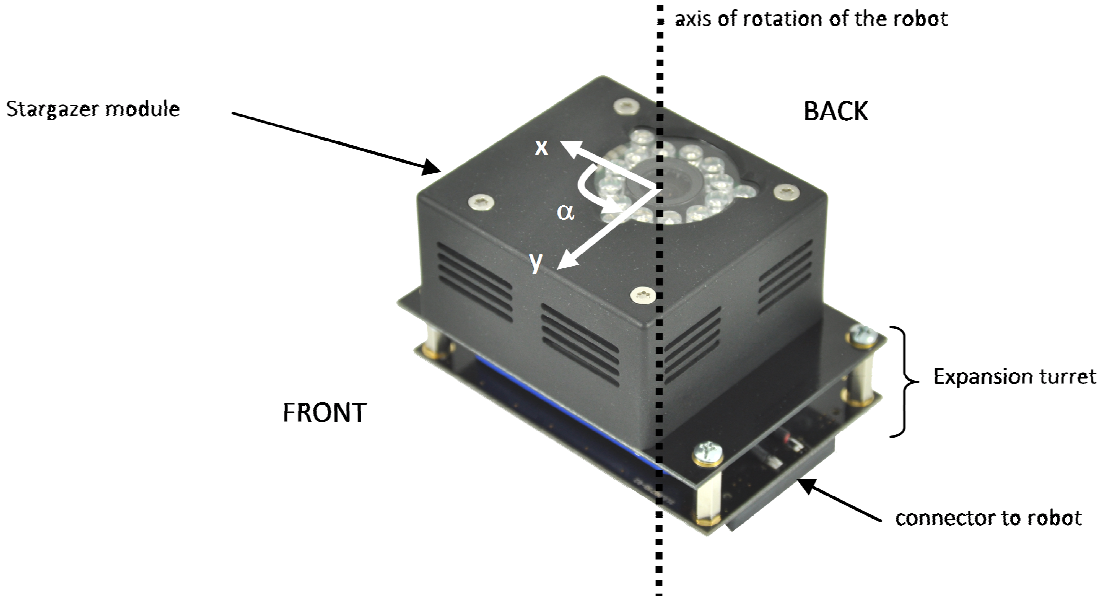


Figure 7.1: KSG overview

8.2 KSG Hardware

The KSG module is composed of a Stargazer position module from Hagisonic co. Ltd and the K4LFSG battery module. There are also landmarks to put on the ceiling.

The Stargazer module has IR LEDs for lighting landmarks and a camera for capturing the image, which is processed by onboard electronics. It computes position and bearing related to landmarks stuck on the ceiling.

The K4LFSG battery module is composed of a battery and electronics for managing its charge and connections to the robot. The module is ON at startup, and OFF when the Khepera IV is OFF.

8.3 KSG Software

The position and bearing are computed internally in the Stargazer and they are communicated through the serial port (/dev/ttyS2) to the robot using several commands.

For the ease of use, functions were developed and included in the Libkhepera library. A test program named ***kgazer_test*** is also provided. It initializes the module and displays the position in an ASCII x-y graph. The robot can be controlled with the keyboard arrows.

The parameters of the KSG are saved into the KSG and remain even after power OFF/ON. The exception is the calibration parameter.

9 Usage

9.1 Required hardware / software

The required hardware and software to use the board and develop programs are described below.

9.1.1 Required hardware

- Computer with Bluetooth or Wi-Fi access
- KSG module
- Khepera IV robot

9.1.2 Required software

- Khepera IV light toolchain installed (see Khepera IV User Manual)
- Libkhepera library

Remarks:
You may find updated version of these software at:

http://ftp.k-team.com/

9.2 Assembly

9.2.1 KSG on the robot

- The assembly of the board with the robot is depicted in Figure 3Figure .
- The assembly and disassembly must be done while the robot is switched OFF; also the robot and KSG must not be charging.
- Insert the KSG module on top of Khepera IV (or Khepera IV extensions stack).
- Pay attention to the orientation of the module (front/back).

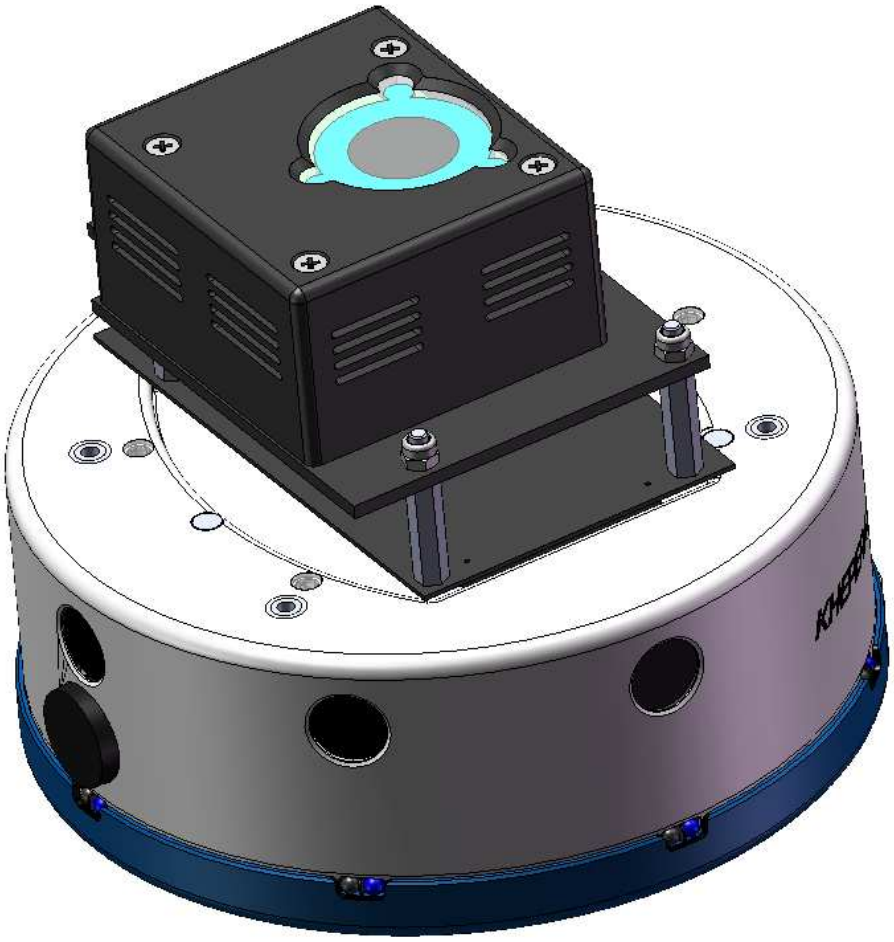


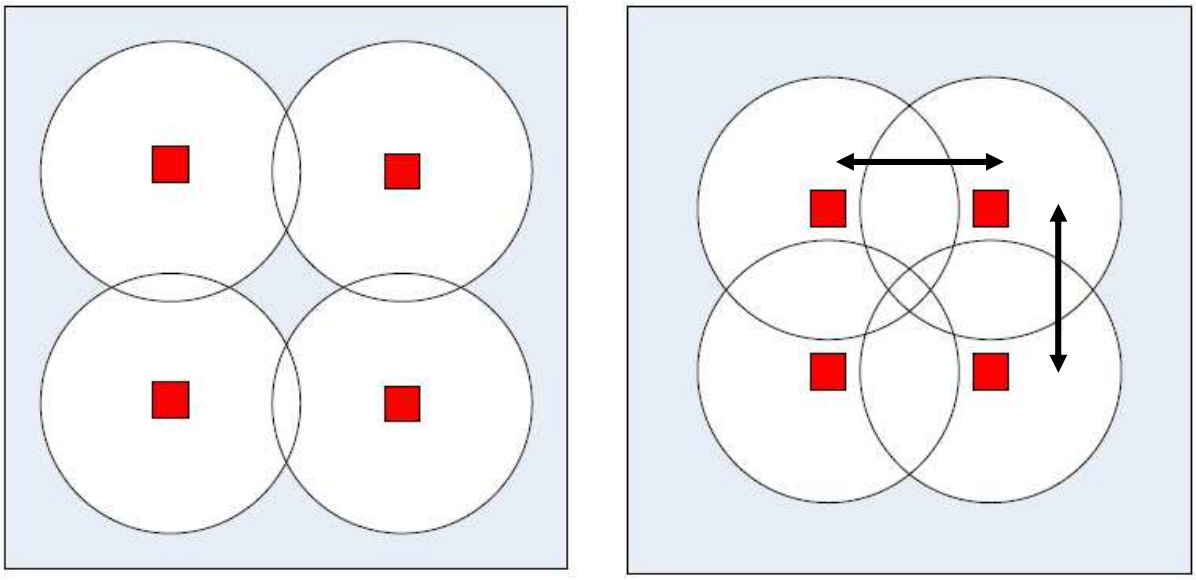
Figure 3: Assembly

9.2.2 Landmarks setting

The landmarks should be placed at maximum of ***x*** m interval on the ceiling for the height of about ***h*** m in order that any dead zone may not occur (figure 4b), with a relation of:

Equation 4.1: $x = 0.8 \cdot h$

Example: h=2.5 m => x = max 2m



a) with dead zone (b) without dead zone

Figure 4: The placement of landmarks

You don't need to install all the 4 landmarks if your area is smaller. And they don't need to be placed in a square shape because during the map building process, the relative position of the landmarks are computed.

The default landmarks delivered with the KSG are of type ***HLD1S***. They can be used with a height range of 1.1-2.9 m. See chapter 3 A and appendix C of "Stargazer User's Guide" for more information about the landmark:

http://ftp.k-team.com/KheperaIII/KSG/StarGazer_Guide_02.0904.16(English).pdf

9.3 Power-up and test

The KSG module is switched ON when the Khepera IV robot is switched ON. It starts autonomously sending position through the serial port, if it sees a landmark.

9.3.1 Test

You can also launch the ***kgazer_test*** program (see chapter 6.2) or test with the instructions below:

- put the robot so that it can see landmarks (for placing landmarks, see chapter 9.2.2).
- after the robot is switched ON, you should see some red LED ON inside the Stargazer, from its top window.
- change ssh escape character on your PC . If it is ~, it will conflict only with the Stargazer command for the test below, not for the normal use of that manual for Linux, run:
  - sudo sh -c 'echo "EscapeChar ^" >>/etc/ssh/ssh_config'
- open a ssh connections terminal to the robot (see ***Khepera IV User Manual***).
- run the following command to set the serial port the terminal:
  - stty -F /dev/ttyS2 115200 -parenb cs8 clocal –crtscts cat /dev/ttyS2

  => it should display lines like below without interruption if the robot is under landmarks on the first terminal:

  ~^I560|+79.05|-10.94|-106.12|190.00`
  ~^I560|+79.05|-10.93|-106.11|190.00`
  ~^I560|+78.39|-10.97|-106.15|189.16`
  ~^I560|+79.05|-10.94|-106.15|190.00`
  ~^I560|+79.05|-10.94|-106.13|190.00`
  ~^I560|+79.05|-10.93|-106.12|190.00`
  ~^I560|+79.05|-10.93|-106.11|189.82`

The data is respectively landmark ID (560), angle (~79 deg), x position (~-10cm), y position (~-106cm), height to landmark (~190cm).

To stop, push keys ***CTRL*** and ***C*** together at the first terminal.

To test sending commands, please use minicom on your Khepera IV:

- launch minicom with the command: ***minicom –o***
- set its parameters with the sub-menu "**Serial port**" setup of the menu **[configuration]**

(Figure **4.4**) (keys "**Ctrl-A + O**") as described in Figure 4.3.

```
+-----------------------------------------------------------+
| A - Serial Device : /dev/ttyS2                            |
| B - Lockfile Location : /var/lock                         |
| C - Callin Program :                                      |
| D - Callout Program :                                     |
| E - Bps/Par/Bits : 115200 8N1                             |
| F - Hardware Flow Control : No                            |
| G - Software Flow Control : No                            |
|                                                           |
| Change which setting?                                     |
+-----------------------------------------------------------+
```

Figure 5: Minicom serial parameters

- Save the settings with the sub-menu "***Save setup as dfl***"

```
+-----[configuration]------+
| Filenames and paths      |
| File transfer protocols  |
| Serial port setup        |
| Modem and dialing        |
| Screen and keyboard      |
| Save setup as dfl        |
| Save setup as..          |
| Exit                     |
+--------------------------+
```

Figure 6: Minicom configuration menu

- push ***CTRL a*** and ***e*** keys to have local echo.
- write the following command to stop receiving position: ***~#CalcStop`***
  => return should be: ***~!CalcStop`***

Multiple stops may be needed. The command ***~#CalcStart`*** starts again the position computation.

- write the following command to get Stargazer firmware version: ***~#Version`***
  => return should be: ~!Version`~$Version|2.1101.18`

See ***Stargazer User's Guide*** for more information on chapter 6 for commands and on chapter 7 for landmarks:
http://ftp.k-team.com/KheperaIII/KSG/StarGazer_Guide_02.0904.16(English).pdf

9.4 Programming the KSG

The KSG is an extension that can be used only with the Khepera IV. That means that the Khepera IV controls all the functions of the KSG. As all the Khepera IV extensions, a library including all the available functions is provided with the Libkhepera version 1.0 or greater. If you already have a Libkhepera installed in your computer but with an older version, you can download the latest version on our ftp:

http://ftp.k-team.com/

If your Khepera IV has already the latest Libkhepera installed, jump to section 9.4.2

You can check if the latest version is installed:

- Log on the Khepera IV (via ssh, Bluetooth or serial port)
- You can check if you have the 1.0 version in listing the present files:

  ls –s /usr/lib/libkhepera*

  - This should give this, where the number after ***.so.*** is the current version:

    0 /usr/lib/libkhepera.so 315
    /usr/lib/libkhepera.so.1.0

9.4.1 Installation of the latest libkhepera and config. file

To use the KSG with the Khepera IV, it's necessary to install the libkhepera version 1.0 or greater.

Normally, if you have received the Khepera IV at the same time than the KSG, the Khepera IV is ready to be used with the KSG; in this case, you can jump to section 9.4.2. Otherwise, if you have bought the KSG separately, you will need to execute the step described below:

- Log on the Khepera IV (via ssh, Bluetooth or serial port)

You have different ways to update the library:

Autonomously with the package:

- Upload the package file ***libkhepera-1.0-r0_armv5te.ipk*** by ssh, Bluetooth or Serial.
- Remove the old one with the command: ***ipkg remove libkhepera***
- Install the new one with the command:

  ipkg install libkhepera-1.0-r0_armv5te.ipk

By hand:

- Remove the old libkhepera: ***rm /usr/lib/libkhepera****
- Copy or send the ***libkhepera.so.1.0*** file (located in the ***build-khepera-2.6/lib/*** directory of the libkhepera source files) to the directory ***/usr/lib***.
- Link the libkhepera file as ***libkhepera.so***:
  ln -s /usr/lib/libkhepera.so.1.0 /usr/lib/libkhepera.so

9.4.2 Using the KSG with kgazer_test software

The ***kgazer_test*** is a test and example program given also in source code in directory ***src/tests/*** of the Libkhepera library.

After copying this program to the Khepera IV and launching it with the command ***./kgazer_test*** , it will initialize the Stargazer, display the version of its internal software, ask if you would like to change its parameters (number, type and reference of landmark) . Then, it will rotate to do a calibration due to the fact that the ceiling may not be exactly parallel to the KSG module. Then it displays an ASCII interface (figure 4.5).

You can move the robot with the keyboard arrows keys. The robot is represented by the character ***A***,**<**,**>** or ***V*** depending of its orientation. Other keys are defined below:

+/-	:	zoom in / out
Page Up/Down	:	speed up/down the robot
c	:	redo position calibration
g	:	go to xy (choose goal, then go to goal using a very basic algorithm)
m	:	if several landmarks, starts map building
k	:	use / don't use correction from calibration
p	:	change stargazer parameters
s	:	start/stop saving data to file *data_corr.csv*
t	:	display / undisplay old robot position
q	:	exit program

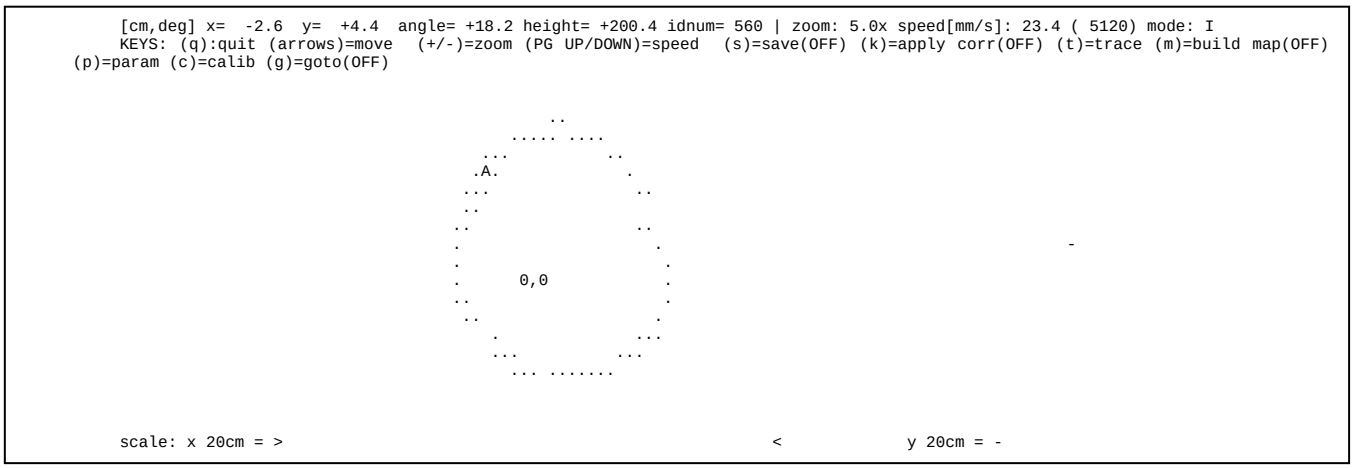


Figure 8.1: kgazer_test ASCII interface

9.4.3 Compiling your own program using the libkhepera

To make your own software to control the KSG, you need to install the development tools for the Khepera IV on your computer (with Linux OS). If it is not already done, please look at the Khepera IV User Manual to know how to install it.

Once done, you can start writing your program. The best solution is to copy the ***kgazer_small_ex.c*** source code (see chapter 0) and modify it. The source code ***kgazer_test.c*** is a more complete example. In any case, keep your program in the ***src/tests/*** directory, and modify the Makefile to compile your new software: add your new program filename in line 35 at the variable ***TARGETS*** of the file in the *src/test/Makefile* or start with the template program in the *template* directory.

As for all the Khepera IV extensions, you need to initialize the library before using the functions described below.

The Libkhepera provides high-level functions which include all the available capabilities of the KSG. Look at the following sections to view the different exported constants variables and functions provided by the Libkhepera for that KSG extension.

9.4.3.1 Constants

NB_MARK_TYPES

Number of types of different landmarks:

#define NB_MARK_TYPES 6

HLD1S

Index of landmark type HLD1S:

#define HLD1S 0

HLD1L

Index of landmark type HLD1L:

#define HLD1L 1

HLD2S

Index of landmark type HLD2S:

#define HLD2S 2

HLD2L

Index of landmark type HLD2L:

#define HLD2L 3

HLD3S

Index of landmark type HLD3S:

#define HLD3S 4

HLD3L

Index of landmark type HLD3L:

#define HLD3L 5

NB_MARK_MODES

Number of landmark modes:

#define NB_MARK_MODES 2

MARK_ALONE

Alone landmark mode index:

#define MARK_ALONE 0

MARK_MAP

Map landmark mode index:

#define MARK_MAP 1

NB_HEIGHT_FIX_MODES

Number of landmark height modes:

#define NB_HEIGHT_FIX_MODES 2

HEIGHT_FIX_NO

Index of the non-fix height:

#define HEIGHT_FIX_NO 0

HEIGHT_FIX_YES

Index of the fixed height:

#define HEIGHT_FIX_YES 1

ANGLE_CORRECTION

Angle correction for calibration:

#define ANGLE_CORRECTION 137.0

CALIB_STDEV_MAX

Maximum standard deviation before calibration error [cm]:

#define CALIB_STDEV_MAX 3.0

9.4.3.2 Variables

External variables for transferring data that are declared in the header file and defined in the library are described below:

kb_gazer_landmark_types

Array of array of char containing landmark types, defined like this:

*const char *kb_gazer_landmark_types[] = {"HLD1S","HLD1L","HLD2S","HLD2L","HLD3S","HLD3L"};*

kb_gazer_landmark_modes

Array of array of char containing landmark modes, defined like this:

*const char *kb_gazer_landmark_modes[] = {"Alone","Map"};*

kb_gazer_height_fix_modes

Array of array of char containing height modes, defined like this:

*const char *kb_gazer_height_fix_modes[] = {"No","Yes"};*

9.4.3.3 High-level functions of the libkhepera

There below you will find details about declared functions for the KSG extensions in the Libkhepera libraries.

The **green** words indicate the type of the function (or parameter) and must not be included in your code.
The **blue** words are the parameters. You must declare it in your code before calling the function.
Only the ***black*** words can be directly added to your code.

int *kb_stargazer_Init (char * DeviceName)*

Initializes the library to use the different KSG functions and stops the module sending position. This function must be called at the beginning of each program using the KSG.

Return: *0* : no error
-1 : error initialising gpio
-2 : cannot open serial port
-3 : cannot communicate with the Stargazer

Parameter: **** DeviceName***: name of the device where the KSG is connected. Should be "/dev/ttyS2".

int *kb_gazer_get_version (char * version)*

Gets Stargazer firmware version.

Return: *0* : no error
-1 : serial port not open
-2 : command not acknowledged
-3 : command not acknowledged
Parameter: **** version***: firmware version.

int *kb_gazer_set_landmark_number (int number)*

Sets the numbers of landmarks to be used.

Return: *0* : no error
-1 : serial port not open
-2 : command not acknowledged
Parameter: ***number***: numbers of landmarks.

int *kb_gazer_get_landmark_number (int * number)*

Gets the numbers of landmarks to be used.

Return: *0* : no error
-1 : serial port not open
-2 : command not acknowledged
Parameter: **** number***: numbers of landmarks.

int *kb_gazer_set_ref_if (int refid)*

Sets the landmark id as reference.

Return: *0* : no error
-1 : serial port not open
-2 : command not acknowledged
Parameter: ***refid***: reference id.

int *kb_gazer_get_ref_if (int * refid)*

Gets the landmark id as reference.

Return: *0* : no error
-1 : serial port not open
-2 : command not acknowledged
Parameter: ****refid***: reference id.

int *kb_gazer_set_landmark_type (int type)*

Sets the landmark type index.

Return: *0* : no error
-1 : serial port not open
-2 : command not acknowledged
-3 : type not valid
Parameter: ***type***: landmark type index.

int *kb_gazer_get_landmark_type (int * type)*

Gets the landmark type index.

Return: *0* : no error
-1 : serial port not open
-2 : command not acknowledged
-3 : unknown landmark type
Parameter: ****type***: landmark type index.

int *kb_gazer_set_landmark_mode (int mode)*

Sets the landmark mode index.

Return: *0* : no error
-1 : serial port not open
-2 : command not acknowledged
-3 : mode not valid
Parameter: ***mode***: landmark mode index.

int *kb_gazer_get_landmark_mode (int * mode)*

Sets the landmark mode index.

Return: *0* : no error
-1 : serial port not open
-2 : command not acknowledged
-3 : unknown landmark mode
Parameter: ****mode***: landmark mode index.

int *kb_gazer_set_height_fix_mode (int mode)*

Sets the height fix mode index.

Return: *0* : no error
-1 : serial port not open
-2 : command not acknowledged
-3 : mode not valid
Parameter: ***mode***: height fix mode index.

int *kb_gazer_get_height_fix_mode (int * mode)*

Gets the height fix mode index.

Return: *0* : no error
-1 : serial port not open
-2 : command not acknowledged
-3 : unknown height fix mode
Parameter: ****mode***: height fix mode index.

int *kb_gazer_start_map_mode (void)*

Starts the map building mode

Return: *0* : no error
-1 : serial port not open
-2 : could not stop receiving position
-3 : command not acknowledged
Parameter: -

int *kb_gazer_set_end_command (void)*

Sets end of commands for update.
Return: *0* : no error
-1 : serial port not open
-2 : command not acknowledged
-3 : data not updated
Parameter: -

int *kb_gazer_start_computation (void)*

Starts computation of position.
Return: *0* : no error
-1 : serial port not open
-2 : command not acknowledged
Parameter: -

int *kb_gazer_wait_stop_computation(void)*

Stops computation of position and wait until it stops (retry MAX_STOP).
Return: *0* : no error
-1 : serial port not open
-2 : command not acknowledged
Parameter: -

int *kb_gazer_stop_computation(void)*

Sends the computation stop of position.
Return: *0* : no error
-1 : serial port not open
-2 : command not acknowledged
Parameter: -

void *kb_stargazer_Close (void)*

Releases the Stargazer.
Return: -
Parameter: -

int *kb_stargazer_read_data* (*double* * x, *double* * y, *double* * z, *double* * angle, *int* * idnum, *char* * cmode, int corr)

Reads and interprets data from the Stargazer. Should be called periodically (up to 10 times/s). ***kb_gazer_start_computation*** must be called once before.

Return: *0* : no error
-1 : cannot serial port not open
-2 : cannot communicate with the Stargazer
-3 : buffer overrun (try to call more often this function!)
-4 : read command not acknowledged
-5 : data error
-6 : no landmark found
-7 : data error
-8 : no received data
-9 : mapid error
1 : update parameters after map building mode
>1: MAPID in map building mode

Parameter:
***x**: x position relative to the reference landmark in [cm], right dir.
***y**: y position relative to the reference landmark in [cm], forward dir.
***z**: height to the landmark in [cm]
***angle**: angle relative to the reference landmark orientation in [degree] (counterclockwise) in 0..360 range
***idnum**: id number of the currently used landmark
***cmode**: current mode: 'F' = map building mode, 'I' map mode, 'Z' height calculation mode
corr: 1 apply position correction; 0 do not apply it

int *kb_gazer_calibration* (*knet_dev_t* * mot1,*knet_dev_t* * mot2,*double* *_center_x0,*double* *_center_y0,*double* *_angle_rot,*double* *_a_axis, *double* *_b_axis, *double* *_stddev_x, *double* *_stddev_y)

Configures the Stargazer rotation compensation by moving the robot around itself then fitting the result ellipse.

Return: *0* : no error
-1: timeout while computing calibration
-2: data buffer too short
-3: error computing ellipse parameters
-4: error: data are too scattered
< -4+ (return kb_stargazer_read_data): error getting Stargazer data

Parameter: ***mode**: height fix mode index.
***mot1**: left motor pointer
***mot2**: right motor pointer
***_angle_rot**: angle of rotation of the fitted ellipse
***_a_axis**: half major axis of the fitted ellipse
***_b_axis**: half minor axis of the fitted ellipse
***_stddev_x**: standard deviation error of x
***_stddev_y**: standard deviation error of y

9.4.4 Software example

A small example of software controlling the KSG is enclosed below. You can find its source code file in ***libkhepera-VERSION/src/tests/kgazer_small_ex.c***.

This sample example does not use calibration of the sensor due to parallelism error of the ceiling and sensor plan. This results in x/y error when the robot rotates. See ***libkhepera-VERSION/src/tests/kgazer_test.c*** for a full example with calibration.

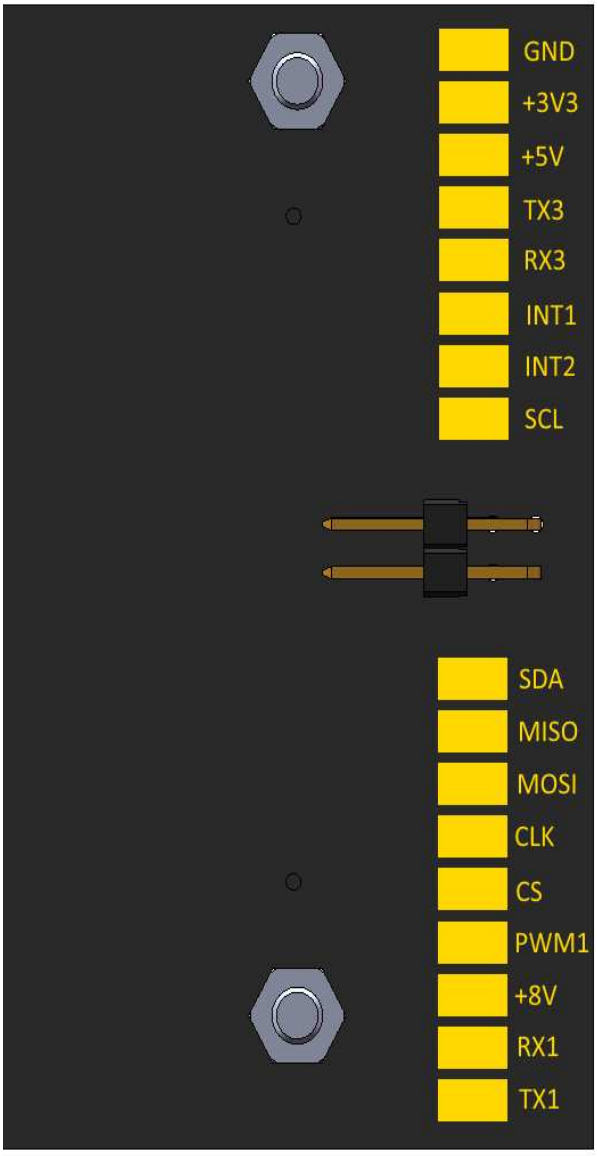
```
#include <khepera/khepera.h>

int main(int argc, char *argv[])
{
    char version[128];
    int ret=0,c=0,i,idnum;
    float angle,x,y,z,xc,yc; // stargazer returned variables and corrected position
    char cmode;     // current mode of the Stargazer

    /* reset the screen */
    kb_clrscr();

    printf("\nKhepera III Stargazer small example program\n");

    /* Set the libkhepera debug level - Highly recommended for development. */
    kb_set_debug_level(2);

    // initialise the libkhepera
    if((ret = kb_init( argc , argv )) < 0 )
      return -1;

    printf("\nInitialising Stargazer module; please wait!\n");

    // initialise the Stargazer module
    if ((ret=kb_stargazer_Init())!=0)
   {
      printf("\nError initialising the Stargazer  (error = %d)!\n",ret);
      kb_stargazer_Close();
      return -2;
   }

    // get Stargazer firmware version
    kb_gazer_get_version(version);
    printf("\nStargazer version is: %s\n",version);

    // read parameters values
    printf("\nRead current Stargazer parameters:\n");
    kb_gazer_get_landmark_number(&c);
    printf("  landmark number: %d\n",c);

    kb_gazer_get_ref_id(&c);
    printf("  reference id   : %d\n",c);

    kb_gazer_get_landmark_type(&c);
    printf("  landmark type  : %s\n",kb_gazer_landmark_types[c]);

    kb_gazer_get_landmark_mode(&c);
    printf("  landmark mode  : %s\n",kb_gazer_landmark_modes[c]);

    kb_gazer_get_height_fix_mode(&c);
    printf("  height fix mode: %s\n",kb_gazer_height_fix_modes[c]);

    kb_gazer_start_computation(); // start computation of position


  // continues on next page
```

```
        for (i=0;i<10; i++) // read 10 data
        {
            // read sensor values
              ret=kb_stargazer_read_data(&x,&y,&z,&angle,&idnum,&cmode,0);

              switch(ret)
              {
                  case 0:
                          printf("data %d: [cm,deg] x= %+6.1f  y= %+6.1f  angle= %+6.1f
    height= %+6.1f idnum= %4d mode: %c\n",i,x,y,angle,z,idnum,cmode);
                  break;
                  case -3:
                      fprintf(stderr,"\nERROR:   read   error:   buffer   too   short,
    leaving!\n");
                  break;
                  case -6:
                      printf("The sensor does not see any landmark!\n");
                  break;
                  case -8:
                      fprintf(stderr,"\nERROR: no data received!\n");
                  break;
                  default:
                      fprintf(stderr,"\nERROR: read error number %d!\n",ret);
              }

              usleep(100000); // wait for next data
          }

          kb_stargazer_Close();
          return 0;
      }
```

10 GPIOs

The expansion turret give access to several GPIOs of the KheperaIV gumstix. These Ios are directly connected from the KB-250 bus, no voltage translator or any protection has been added on this board.

All GPIOs level are at 3.3V, there's a voltage translator on the KheperaIV (to convert the 1.8V level of the gumstix). The GPIOxxx denomination correspond to the Gumstix IO number.



: Main ground
: +3.3V supply output (max 2A)
: +5V supply output (max 2A)
: COM3 Tx (ttyS2), GPIO166
: COM3 Rx (ttyS2), GPIO165
: Interrupt 1 input, GPIO10
: Interrupt 2 input, GPIO186
: I2C SCL, GPIO184

: I2C SDA, GPIO185
: SPI MISO, GPIO173
: SPI MOSI, GPIO172
: SPI CLK, GPIO171
: SPI CS1, GPIO175
: PWM1, GPIO144
: Kh4 battery Voltage
: COM1 Rx (ttyS0), GPIO151
: COM1 Tx (ttyS0), GPIO148

Figure 9.1 : GPIOs connection

11 LRF & STARGAZER SETUP

If you need both the LRF and StarGazer function at the same time, the expansion turret can customize to use these two module on the same Robot. The usage of the module will be exactly as describe above for each turret.

Just ask K-team for this special stack-up.

!!! When using this stack-up, the Robot need to be driven very smoothly to avoid balancing problem due to the height of the turret!!!

11.1 Overview

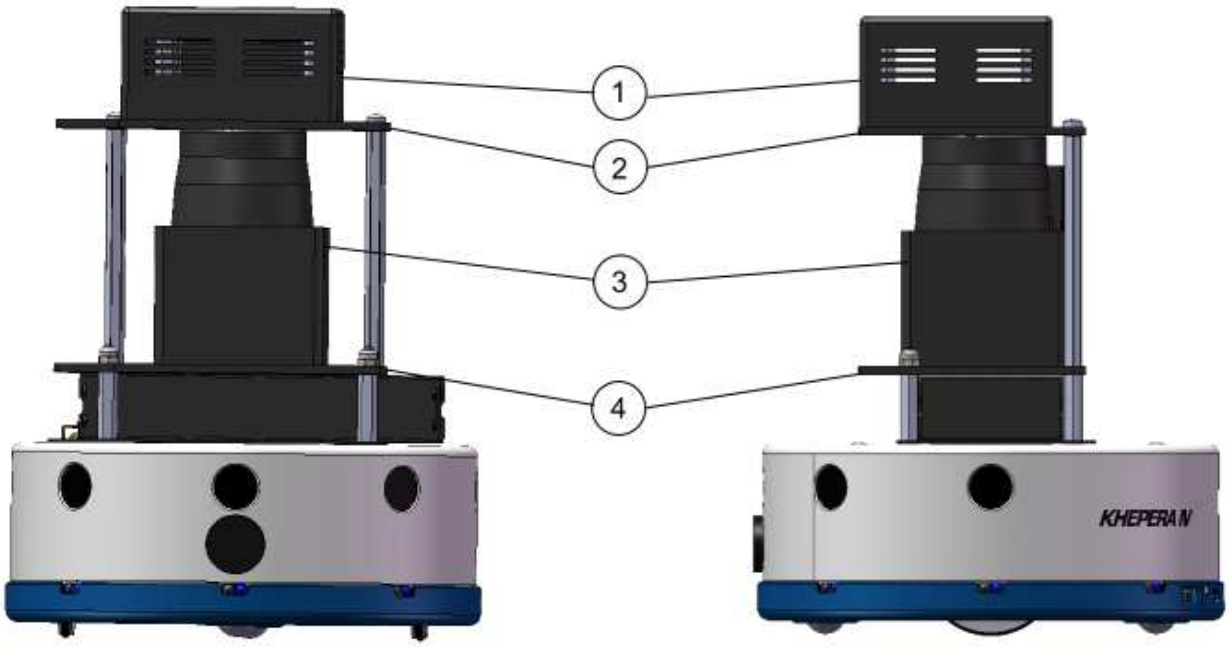


Figure 9.1 : LRF & StarGazer setup on the KheperaIV

1 StarGazer Module (with its cable)
2 Fixing PCB with its two spacer
3 LRF module (with its mini-USB cable)
4 Expansion turret

11.2 Assembling

This stack-up will be delivered fully assembled and ready to used. You just need to plug it on the KheperaIV. Be sure that the Robot is turn OFF during the operation and be careful of the Extension connector.

If you have already a Gripper, this extension can be also plugged on Top of the Gripper turret. In this setup, the LRF will be mask by the Gripper when the Arm is in high position.

12 WARRANTY

K-TEAM warrants that this product is free from defects in materials and workmanship and in conformity with the respective specifications of the product for the minimal legal duration, respectively one year from the date of delivery, under normal use conditions.

Upon discovery of a defect in materials, workmanship or failure to meet the specifications in the Product during the aforementioned period, Customer must request help on K-Team Internet forum on http://www.k-team.com/forum/ by detailing:

The type of the product used (package, version, & serial number).

The extension modules.

The programming environment of the robot (standard, version, OS).

The standard use of Product before the appearance of the problem.

The description of the problem.

If no answer is received within two working days, Customer can contact K-TEAM support by phone or by electronic mail with the full reference as stated below

If the defect is identified as a “warranty” related problem, K-TEAM shall then, at K-TEAM's sole discretion, either repair such Product or replace it with the equivalent product without charging any technical labour fee and repair parts cost to Customer, under the condition that Customer brings such Product to K-TEAM within the period mentioned before. Repair or replacement under warranty does not entitle to original warranty team extension.

This limited warranty is invalid if the factory-applied serial number has been altered or removed from the Product.

This limited warranty covers only the hardware and software components contained in the Product. It does not cover technical assistance for hardware or software usage and it does not cover any software products contained in the Product.

This limited warranty is non-transferable.

It is likely that the contents of Customer's flash memory will be lost or reformatted in the course of the service and K-TEAM will not be responsible for any damage to or loss of any programs, data or other

information stored on any media or any part of the Product serviced hereunder or damage or loss arising from the Product not being available for use before, during or after the period of service provided or any indirect or consequential damages resulting therefore.

If during the repair of the product the contents of the flash memory are altered, deleted or in any way modified, K-Team is not responsible whatever. Customer’s product will be returned to customer configured as originally purchased (subject to availability of software).

Be sure to remove all third parties' hardware, software, features, parts, options, alterations, and attachments not warranted by K-TEAM prior to Product service. K-TEAM is not responsible for any loss or damage to these items.

This warranty is limited as set out herein and does not cover, any consumable items (such as batteries) supplied with the Product; any accessory products which is not contained in the Product; cosmetic damages; damage or loss to any software programs, data, or removable storage media; or damage due to (1) acts of God, accident, misuse, abuse, negligence, commercial use or modifications of the Product; (2) improper operation or maintenance of the Product; (3) connection to improper voltage supply; or (4) attempted repair by any party other than a K-TEAM authorized robot service facility.

This limited warranty does not apply when the malfunction results from the use of the Product in conjunction with any accessories, products or ancillary or peripheral equipment, or where it is determined by K-Team that there is no fault with the Product itself.

K-Team expressly disclaims all other warranties than stated hereinbefore, expressed or implied, including without limitation implied warranties of merchantability and fitness for a particular purpose to the fullest extent permitted by law.

Limitation of Liability: In not event shall either party be liable to the other for any indirect, special, incidental or consequential damages resulting from performance or failure to perform under the contract, or from the furnishing, performance or use of any goods or service sold or provided pursuant hereto, whether due to a breach of contract, breach of warranty, negligence, or otherwise. Save that nothing herein shall limit either party’s liability for death or personal injury arising from its negligence, neither party shall have any liability to the other for indirect or punitive damages or for any claim by any third party except as expressly provided herein.

K-Team S.A.
Z.I. Les Plans-Praz 28
1337 Vallorbe
Switzerland