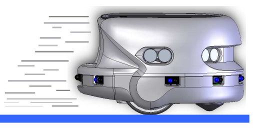
KHIII-Gripper

user manual



K-Team
MOBILE ROBOTICS

version 1.0
APRIL 2010

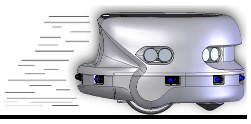
Documentation Author

Frederic Lambercy
K-Team S.A.
Rue Galilee 9, Y-Park
1400 Yverdon-les-Bains
Switzerland

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

TABLE OF CONTENTS

1. INTRODUCTION



The hardware of the KheperaIII is based on a modular concept. The gripper is a turret that can be plugged on the robot **only if a KoreBotII is already plugged**. Due to the configuration of the gripper, other turrets can be plugged on the top of it, such as a KoreUSBCam, a KoreLRF, a KoreIO, etc…

The Gripper was designed to carry objects with different size and weight, there's no minimal limit as the gripper can apply a continuous torque even if the object is very thin. The maximum size and weight are respectively 50mm and 50g.

The KheperaIII Gripper is equipped with a Li-Pol Battery to keep the autonomy of the KheperaIII Robot as the same level as without an extension. The voltage and the remaining capacity are managed by the microcontroller in the base board. This battery must be charged with the same charger as the Robot.

Two microcontrollers installed in the turret (one in the head of the gripper, the other in the base) manage all local functionalities. The one which is in the head, manages the position of the two fingers, the torque and the sensors (optical barrier, two IR sensors and resistivity sensor). A network based on the I2C bus connects the two microcontrollers to the KorebotII processor. Each of the two microcontrollers can be controlled through a library available in the libkorebot version 1.14 or greater.

1.1. How to use this manual

This manual introduces the KheperaIII Gripper extension. To learn how to make the best use of your gripper turret you are urged to read all of chapters 2 through 6.

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

1.2. Safety precautions

Here are some recommendations on how to correctly use the KheperaIII Gripper:

- **Keep the gripper away from wet area**. Contact with water could cause malfunction and/or breakdown.
- **Don’t force mechanical movement of the gripper!** Movements of the gripper mechanical parts have to be controlled ONLY by software. Forcing arm or gripper (open or closed) movements can damage the mechanical parts of the gripper!
- **Store your gripper 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 KheperaIII robot.** Do not try to use another charger; this can cause irreversible damage to the battery.
- **Do not plug or remove the gripper while the robot is powered on**. To avoid any damage, make all connections when the robot power is off.
- **Never leave the KheperaIII and the Gripper powered when it is unused**. When you have finished working with KheperaIII, 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. THE GRIPPER

2.1. Unpacking and inspection

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

- the KheperaIII Gripper Turret
- the support CD with:
  - this User manual
  - the Libkorebot library

2.2. Global View

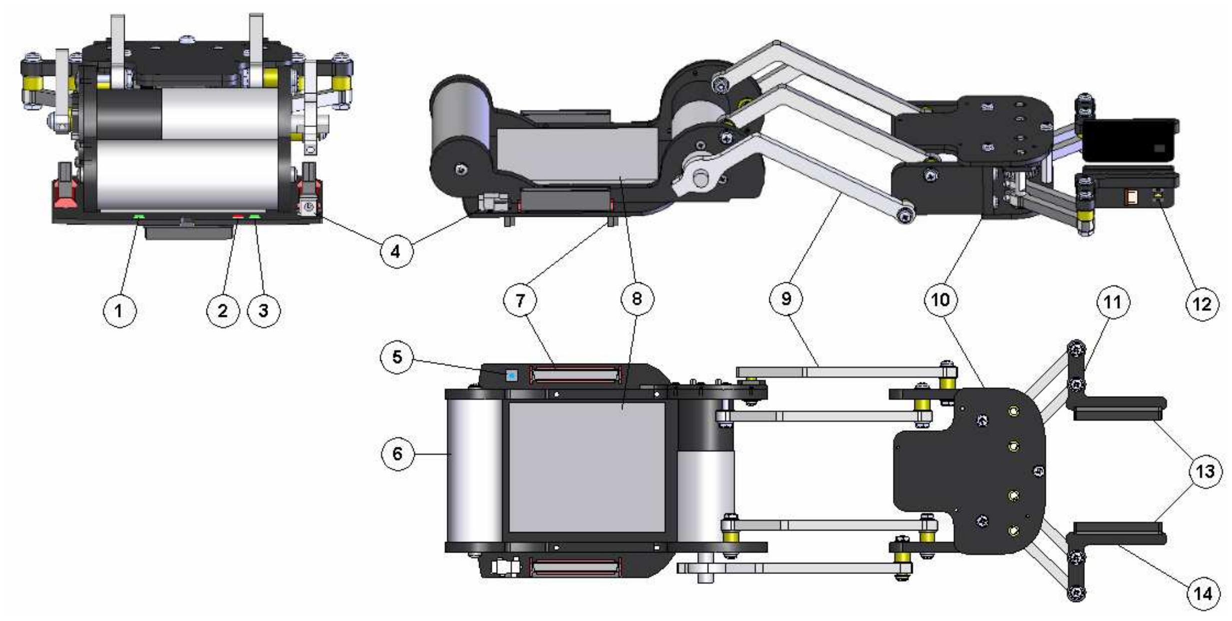


Figure 3.2: Overview of the KheperaIII Gripper

1	Power ON Led	8	Li-Pol Battery 1000mAh 7.4V
2	Charge in progress Led	9	Arm
3	Charge complete Led	10	Head of the Gripper
4	Power input connector	11	Left finger
5	Reset button	12	IR sensor
6	Counterweight	13	Resistivity sensor
7	KB250 extension connector	14	Right finger

2.3. Arm

The arm makes the moving of the gripper possible. The maximum movement is shown in figure 3.3. It allows the Gripper to grab an object in front of the Robot (Ground Position) and lift it above the KheperaIII (High Position) to keep the IR and US sensor free of obstacle and to improve the balance of the Robot.

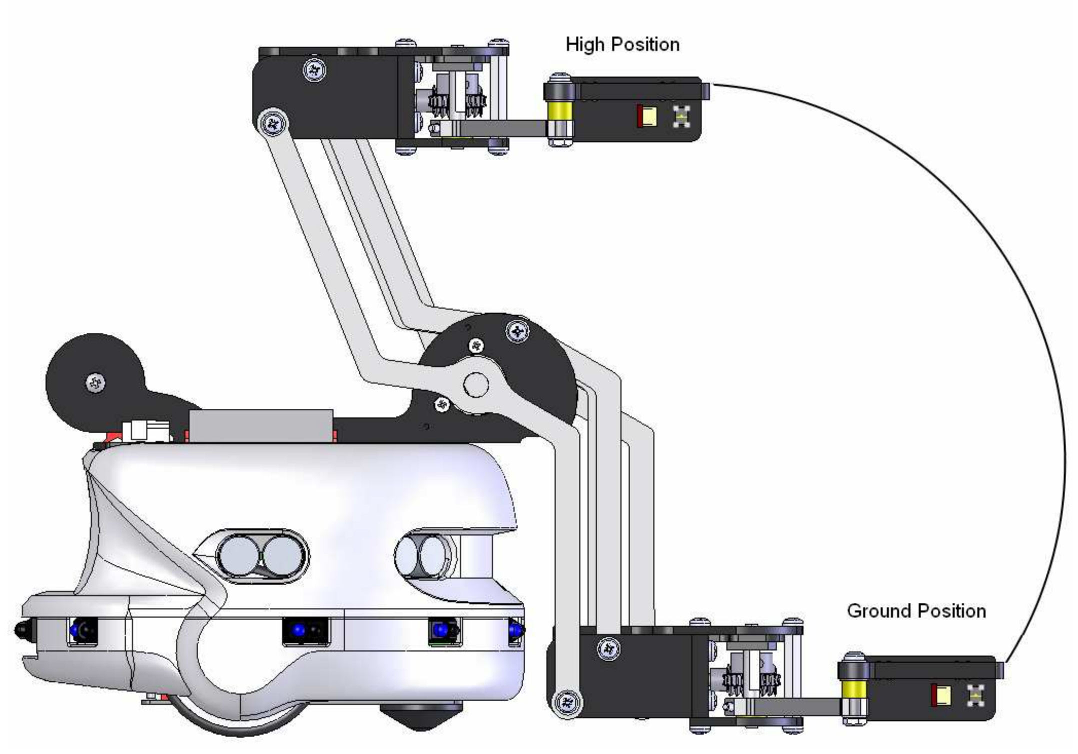


Figure 3.3: Possible positions of the Arm

The arm is moved by a DC motor coupled with a position sensor (potentiometer) which provides an absolute positioning. The regulation is made by the microcontroller in the base board. The maximum positions (limits of movement) of the arm are limited by the software, the two limits positions (High and Ground) are saved in the EEPROM of the microcontroller as soon as a search limits function is made. These two limits can be modified by the user if a new search limits is done, this is useful if an extension is plugged above the gripper, to avoid the Gripper to hit the extension during each movement.

The user can read at any time the actual position, speed and order of the arm. It's also possible to get minimum and maximum position (limits of movement) of the arm and the maximum speed parameter. As the gripper uses the current in the motor to detect the mechanical limits during the search limits function, this value is also accessible for the user. The value can be converted in a real unit using the following formula:

$$\mathbf{M}ax\mathbf{C}urrent = 2[A]$$

$$\mathbf{C}urrent[A] = \frac{\mathbf{V}alue}{1023} \cdot \mathbf{MC} = \frac{\mathbf{V}alue}{1023} \cdot 2 = \frac{Value}{511.5}$$

For example, if the value of the motor current is 255, the real current in the motor will be 500[mA]. Please note that this value is not an average but a crest current, as the measure is made whenever the PWM is turned off.

The user will be able to configure the maximum speed of the arm (0-15) and the position. If the position order is out of the movement limits, the gripper will limit automatically the order to avoid any damage. The position is given by a potentiometer mounted on the arm. Its value is read by a 10 bits ADC (0-1023). But as the movement are mechanically limited, the limits value are approximately 300 (High Position) and 900 (Ground Position), these values can change a little bit if you make a new search limits function.

Each increment of the position sensor corresponds to 0.325° in the position of the arm. For example, to move the arm 30° in direction of the ground, add 92 to the position order of the arm.

The speed is also calculated with this position. At each control loop (every 10ms) the speed is updated. So the speed value corresponds to the movement made by the arm in 10ms. To calculate the real speed with the value given by the Gripper, use this:

$$\mathbf{P}osition\mathbf{I}ncrement = 0.325[°]$$

$$\mathbf{C}ontrol\mathbf{L}oop\mathbf{T}ime = 0.01[s]$$

$$\mathbf{S}peed[°/s] = \frac{\mathbf{V}alue \cdot \mathbf{PI}}{\mathbf{CLT}} = \frac{\mathbf{V}alue \cdot 0.325}{0.01} = \mathbf{V}alue \cdot 32.5$$

For example, for a speed value of 5, the arm is currently moving at 162.5[°/s]. Please note that this value is the default speed after each boot. The arm won't move all the distance at this speed, it will start with an acceleration ramp and stop with a deceleration ramp. The maximum speed parameter is very important if you want to carry heavy object (50g or greater). The Gripper is designed to carry up to 50g, but the motor can lift more if the Robot doesn't topple over. In this case, it's useful to limit the maximum speed. In the opposite, if you carry only light object, you can increase the speed up to 15 (= 487.5[°/s]). Above this value, the microcontroller will limit automatically the speed to 15.

All values returned by the gripper are refreshed every control loop; i.e. every 10ms.

2.4. Gripper

The Gripper is made of two fingers which are moved by the same DC motor. These two fingers are parallel in every position to improve gripping ability. The maximum opening of the gripper is 55mm as shown in figure below:

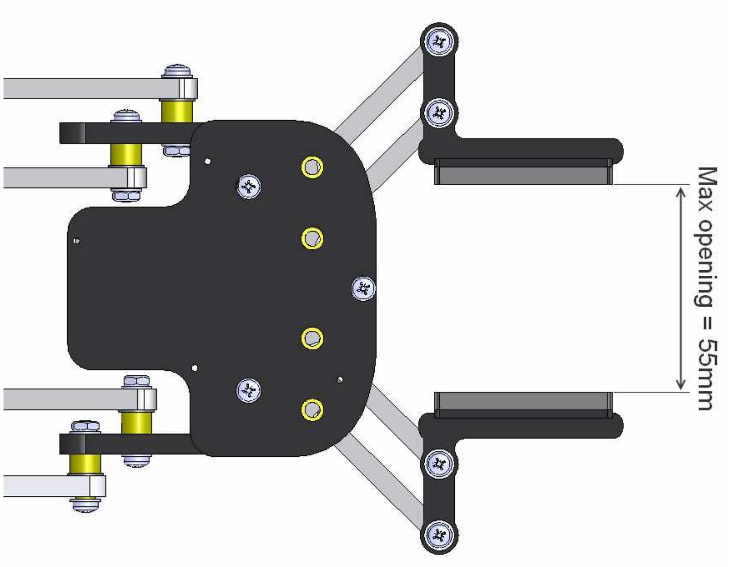


Figure 3.4: Max opening of the gripper.

The gripper is controlled in position and torque by the microcontroller in the head board. The user can set every position between the closed and the max opening position. As every product is a little bit different than each other, the gripper has the opening limit position saved in the EEPROM of the microcontroller, which is approximately a value of 170. This limit is updated when the search limits function is execute. The minimal position which can be set is 1. When the position is set to 0, the gripper will automatically enter in Torque control to ensure a constant torque even if the object is very small.

The user can also read the position of the gripper to measure the size of the object gripped. There are two possibilities: The simplest solution is to linearize the position of the Gripper; the problem of this solution is the error that can be greater than 5mm.

$$\mathbf{M}ax\mathbf{O}pening = 170 = 55[mm]$$

$$\mathbf{O}bject\mathbf{S}ize = \frac{\mathbf{V}alue}{170} \cdot 55 = \frac{\mathbf{V}alue}{3.1}[mm]$$

Formula to calculate the size of the object with linearization

The second possibility is to take care of the entire mechanical characteristic to get a better conversion. Please note that even with this formula, the measures won’t be very precise (+/- 1mm) due to the mechanical play.

$$\mathbf{O}bject\mathbf{S}ize[mm] = 2 \cdot [35 \cdot \sin(0.325 \cdot (\mathbf{V}alue + 13.6)) - 2.7]$$

Formula to calculate the size of the object with all parameters (more accurate)

Once the fingers are blocked by an object, the microcontroller switches to the torque control mode and limits the motor current to a given value. Configuring a small torque can avoid damaging a fragile object. In opposition, it's necessary to configure a high torque if you want to carry a heavy object. To convert the measured value to a real current, use the following formula:

$$\mathbf{M}axCurrent = 217[mA]$$

$$\mathbf{C}urrent[mA] = \frac{\mathbf{V}alue}{1023} \cdot 217 = \frac{\mathbf{V}alue}{4.7}[mA]$$

2.5. Battery

The KheperaIII Gripper has an integrated battery to keep the autonomy of the robot at the same level than usual. The Li-Pol Battery of the gripper has a 1000mAh capacity under 7.4V to ensure autonomy up to 4 hours. The battery status is managed by the microcontroller in the base board (same as arm management). The user can view the voltage and the remaining capacity. The capacity is evaluated from the voltage, so the two measures are linked. To convert the returned value to a real voltage, use the following formula:

$$\mathbf{B}attery\mathbf{V}oltage[V] = \frac{\mathbf{V}alue}{102.3}$$

This battery doesn't replace the KheperaIII robot battery; you must plug a charged battery in the robot to use the robot and the gripper. The gripper will be supplied only if it's mounted on a KheperaIII which is correctly powered and turn on. As soon as the KheperaIII is turn off or the battery is empty, the gripper supply will be cut.

The charger is not provided with the gripper as it's the same as the KheperaIII Robot. To charge the gripper battery, you must use the same charger as the robot and connect it to the power connector (see n°4 in the overview). The red led (n°2 in the overview) will turn on to indicate that the battery is in charge. As soon as the battery is completely charged, the red led will turn off, and the green led (n°3 in the overview) will turn on.

You can charge the battery even if the gripper is mounted or not on a KheperaIII. The battery can also be charged if the KheperaIII and the gripper are turn on. The only difference will be the charged time. It's faster to charge the battery when the robot is turned off than the opposite.

Warning, use only the official charger provided with the KheperaIII robot to charge the gripper.

2.6. IR sensors and optical barrier

The KheperaIII Gripper is equipped with two IR sensors (one on each finger) to create an optical barrier capable to detect if an object is between the two fingers. Each IR sensor is, in fact, composed by an emitter and a receiver. Along the optical barrier, the user can read the value of the reflected light on the object, which gives the opportunity to detect the colour of the grabbed object. This measure must be done only when the gripper is closed to ensure that the distance (object <=> sensor) is always the same. Besides this, you can use these sensors value to measure the distance between the object and the fingers. The second option is only viable if the colour is known, and useful if you want to place the object in the centre of the gripper before grabbing it.

2.7. Electrical resistivity sensor

Two conductive foams are mounted on the internal side of each finger (point 13 in the overview) to measure the electrical resistivity of the gripped object.

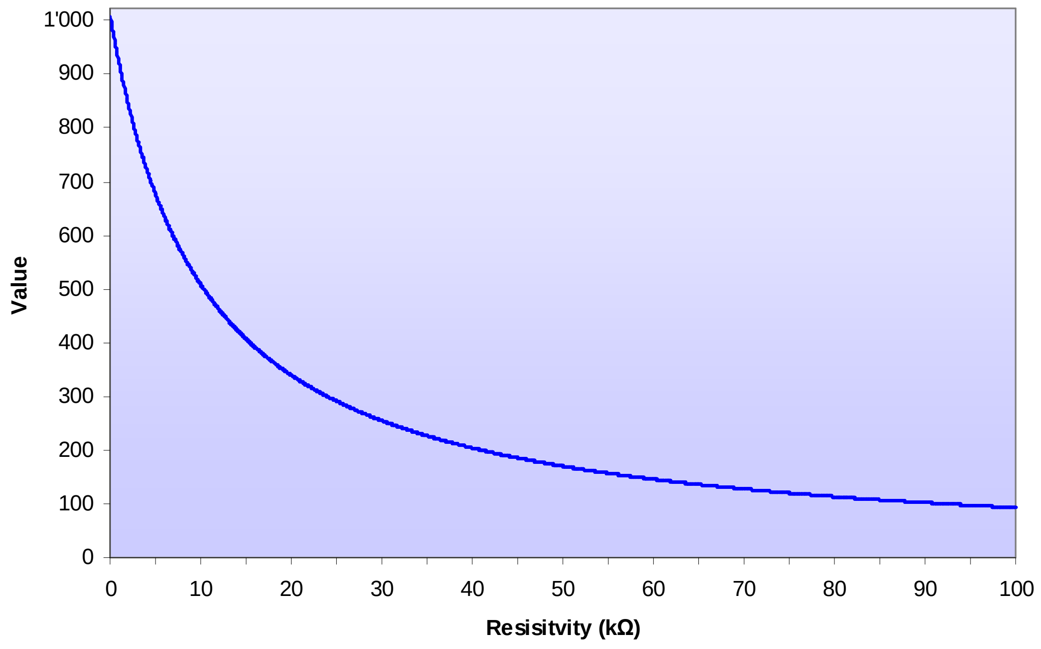


Figure 3.7: Measured value of the sensor versus resistivity of the gripped object

To get the exact resistivity of the gripped object, the formula below shows how to calculate the real value with the returned value:

$$\mathbf{R}esistivity[k\Omega] = \frac{10'230}{\mathbf{V}alue} - 10.15$$

A returned value of 1000 indicates that a conductive object is currently gripped (metal, for instance), a value near of 0 indicates that the object has a high electrical resistivity (plastic, wood, …). Note that if there isn't any object in the gripper and the fingers are closed, the value will be at the maximum level (1000) because the two conductive foams will be directly in touch. The same thing can happen if the gripped object is too small.

3. 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.

3.1. Assembling

Assembling is the easier operation, but it is also necessary to perform it carefully:

- Before assembling the gripper on the KheperaIII you must be sure that a KoreBotII is already mounted inside. If not, look at the KheperaIII User manual to know how to open the KheperaIII and plug the KoreBotII. You must mount only the main board of the KoreBotII. The top PCB (the one with only connectors) must be removed before closing the KheperaIII. This top board is not useful with the gripper.
- If the black protection plate is mounted on KheperaIII, remove it before connecting the gripper and, very important, be sure that the KheperaIII robot is turn off.
- Then insert the gripper on the KheperaIII with the arm directed to the front of the Robot. Do not try to mount the gripper the other way round; this can cause irreversible damage to the KoreBotII connector.
- When the gripper is correctly engaged in the KB-250 bus connectors, push the gripper straight to plug it. If it's too hard to plug the gripper, do not force on it. The connectors are certainly not correctly aligned.
- Once the gripper is plugged, the three magnets of the KheperaIII body coupled with the gripper magnets will ensure a good maintenance of the turret.

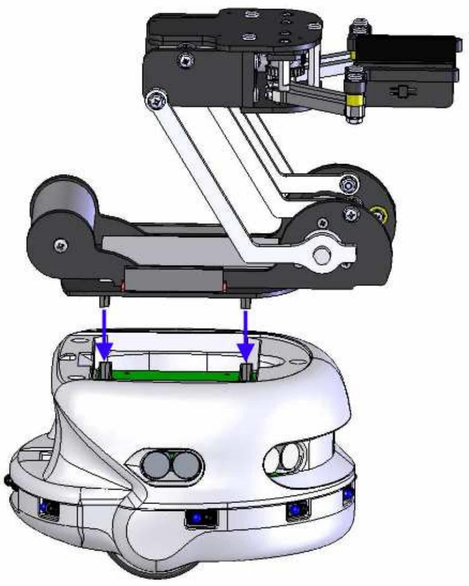


Figure 4.1: How to plug the gripper on the KheperaIII

3.2. Disassembling

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

- First switch OFF the KheperaIII robot. To remove the battery is not enough.
- Take the gripper with one hand and maintain the KheperaIII with the other. Do not pull on the arm to unplug the gripper, place your fingers on the motor and on the counter weight to pull correctly the gripper out of the KheperaIII connectors.
- Pull the gripper straight and very carefully. Once unplugged, place the gripper in its case to store it.

3.3. Unpacking test

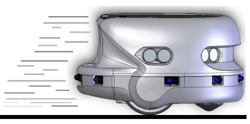
After unpacking it is important to test the functionality of the gripper. A test that uses both motors and all the sensors can be performed if you operate as following:

- Plug the gripper on a KheperaIII equipped with a KoreBotII
- Connect the KheperaIII to a computer using a KoreConnect and a serial cable
- Open a terminal on your computer and turn on the robot
- Once the login passed, run the kgripper_test program. If it is not yet installed on your KoreBotII, follow the step in the section 4.1.
- Type the command “get 1” and enter.

After this command, the gripper will open its finger, and the arm will go in ground position. Then the gripper will wait to see an object between its fingers. Place a little object in the finger, the gripper will close its finger and grab the object to place it in high position. Once done, the resistivity of the object will appear on the terminal. You can try with one object made of wood (or plastic) and one made of metal to compare and verify that the sensors work correctly.

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

4. PROGRAMMING THE GRIPPER



The Gripper is an extension that can be used only with the KoreBotII mounted in the KheperaIII. That means that the KoreBotII controls all the functions of the Gripper. In fact, the gripper does nothing without an order from the KoreBotII. As all the KoreBotII extensions, a library including all the available functions is provided with the libkorebot version 1.14 or greater. If you already have a libkorebot installed in your computer but with an older version, you can download the latest version with the following link:

http://ftp.k-team.com/korebot/libkorebot/libkorebot-*VERSION*.zip
(Replace the VERSION by the version number, 1.14 or greater)

If your KoreBotII has already the latest libkorebot and configuration file (kgripper.knc) installed, jump to section 4.2.

4.1. Installation of the latest libkorebot and configuration file

To use the gripper with the KheperaIII and KoreBotII, it's necessary to install the libkorebot version 1.14 or greater on the KoreBotII. A configuration file (kgripper.knc) is also necessary to use the gripper.

Normally, if you have received the KoreBotII at the same time than the gripper, the KoreBotII is ready to be used with the gripper; in this case, you can jump to section 4.2. Otherwise, if you have bought the gripper separately, you will need to execute the step described below:

- Log on the KoreBotII (via ssh, Bluetooth or serial port)
- Go in the /usr/lib directory *(cd /usr/lib/)*
- Remove the old libkorebot *(rm libkorebot.so)*
- Copy or send the libkorebot-1.14.so file (locate in the */build-korebot/lib/* directory of the libkorebot).
- Rename the libkorebot file as libkorebot.so *(mv libkorebot-1.14.so libkorebot.so)*
- Go in the /etc/libkorebot directory *(cd /etc/libkorebot/)*
- Copy or send the kgripper.knc file (locate in the */config/* directory of the libkorebot)

4.2. Testing the gripper with kgripper_test software

Before starting with the programming of the gripper, it's important to test the gripper with the kgripper_test software to understand the different functionalities and capabilities of the gripper. If the software is not yet in the /home/root directory of the KoreBotII, copy it (the executable file is located in the /src/tests/ directory of the libkorebot). Then execute it with the command "*./kgripper_test*". A message will indicate that the software is running correctly. If not, return to section 4.1 and install the libkorebot and the configuration file.

The kgripper_test program waits that the user enters a command to control the gripper. To show all the available commands, type *"help"* and enter. These commands are described below. Warning, this software does not support the backspace command, if you have typed a wrong command, type enter and restart. All wrong commands will be returned by 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 enter.

4.2.1. kgripper_test commands

quit or ***exit***

Effect:	Close the software and return to the KoreBotII main prompt
Parameter:	-

status

Effect:	display the different status of the arm and the gripper (position, speed, limits, OnTarget flag, order, etc…). This function can be repeated by setting the parameter bigger than 1 (up to 65535 times)
Parameter:	number of repeated status display

getrev

Effect:	display the revision of the firmware in the arm and gripper microcontroller.
Parameter:	-

movearm

Effect:	move the arm to a given position
Parameter:	Position to reach. This value must be between the two mechanical limits (use the status command to view these limits).

movegrip

Effect:	move the finger of the gripper to a given value
Parameter:	Position to reach. This value must be between 0 and the open limit (use the status command to view the limit). Where a parameter of 0 will force the gripper to close till the torque limit is reached.

sensor

Effect:	display the value of the different sensors in the gripper (distance sensor left and right, ambient light, object detection and electrical resistivity).
Parameter:	number of repeated sensors display.

maxspeed

Effect:	set the maximum speed of the arm use in the position regulation.
Parameter:	max speed of the arm (1-15 where 1=32.5[°/s] and 15=487.5[°/s]).

get

Effect:	release or grab on object using the optical barrier.
Parameter:	1 = open the gripper, and move the arm in the ground position. Wait in this position until an object is detected by the optical barrier. 0 = move to the ground position and release the object. Return to the high position before closing.

demo

Effect:	run a demo mode where the gripper and the arm move together at the same time.
Parameter:	number of repeated cycle (1-65535 times). If the parameter is set to 0, the demo mode will run continuously. To stop the demo, push *CTRL+C* keys to close the software.

torque

Effect:	set the maximum torque used by the gripper to grab an object.
Parameter:	max current in the gripper motor used in the regulation. This parameter could be set between 100 and 900 (22mA to 190mA). Warning, if the parameter is too small (< 200), sometimes the gripper will not close anymore.

search

Effect:	start a search limit protocol to detect the mechanical limits of the arm and the gripper. Once executed, this function will update the limits on the EEPROM of the both microcontroller. **Warning, before executing this command; be sure that the gripper path is free of obstacle. If the gripper does not reach correctly its mechanical limits, restart the protocol.**
Parameter:	-

4.3. Compiling your own program using the libkorebot

To make your own software to control the gripper, you need to install the development tools for the KoreBotII on your computer (with Linux OS). If it is not already done, please look at the KoreBotII User Manual to know how to install it. Once done, you can start writing your program. The best solution is to copy the kgripper_test.c code source 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 program name in line 30 of the makefile).

As for all the KoreBotII extension, you need to initialize the library and open the I^{2}C bus socket before using the functions described below. In every program using the gripper, you will need to include these three lines in the beginning of your program:

```
kgripper_init();
Arm = knet_open( "Kgripper:Arm" , KNET_BUS_I2C , 0 , NULL );
Gripper  = knet_open( "Kgripper:Gripper" , KNET_BUS_I2C , 0 , NULL );
```

The libkorebot provides high-level functions which include all the available capabilities of the gripper. Look at the section 4.3.1 to view the different functions provided by the libkorebot. If you want to access directly to the I^{2}C register, look at the section 4.3.2.

4.3.1. 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.

unsigned int *kgripper_init(* void *)*

Initialize the library to use the different gripper functions. This function must be called at the beginning of each program using the gripper.

Return: 0 if success or <0 if error

Parameter: -

4.3.1.1. Arm functions

unsigned char ***kgripper_Arm_Get_Version(Arm)***

Read the firmware version of the arm microcontroller.

Return: firmware version (0x00-0xFF)

Parameter: -

unsigned short ***kgripper_Arm_Get_Position(Arm)***

Read the actual position of the arm.

Return: arm position (0-1023), where 1 = 0.325[°]

Parameter: -

unsigned char ***kgripper_Arm_Get_Speed(Arm)***

Read the actual speed of the arm.

Return: arm speed (0-255), where 1 = 32,5[°/s].

Parameter: -

unsigned short ***kgripper_Arm_Get_Current(Arm)***

Read the actual current of the arm motor.

Return: arm motor current (0-1023), where 1 = 1,95[mA]

Parameter: -

unsigned char ***kgripper_Arm_OnTarget(Arm)***

Read if the arm has reached its position order. Actually, this value will be set if the difference between the actual position and the order is less than 4 (=1.3[°]).

Return: 1 if the arm has reached its position order, and 0 if not.

Parameter: -

unsigned short ***kgripper_Arm_Get_Voltage(Arm)***

Read the actual voltage of the battery.

Return: 0-1023 battery voltage value, where 1 = 9,77[mV]

Parameter: -

unsigned char ***kgripper_Arm_Get_Capacity(Arm)***

Read the battery relative remaining capacity.

Return: 0-100[%]

Parameter: -

void *kgripper_Arm_Get_Limits(Arm,* unsigned short ** Min_Position* , unsigned short ** Max_Position)*

Read the two mechanicals limits of the arm. This function uses two pointers to get both results in only one call. You need to pass the address of the variables to the function.

Return: -

Parameter: ****Min_Position*:** Value of the high mechanical limit
****Max_Position*:** Value of the ground mechanical limit

unsigned short *kgripper_Arm_Get_Order(Arm)*

Read the actual position order of the arm

Return: arm position order (0-1023), where 1 = 0,325[°]

Parameter: -

unsigned char *kgripper_Arm_Get_Max_Speed(Arm)*

Read the maximum speed used by the arm regulation.

Return: max_speed value, where 1 = 32,5[°/s]

Parameter: -

unsigned char *kgripper_Arm_Get_Search_Limit(Arm)*

Read the search limit flag. If set, the arm is currently searching the mechanical limits. If zero, the search limits protocol is finished, and the arm works as usual (waiting commands).

Return: arm Search Limit flag (0 = done, 1 = in progress).

Parameter: -

void *kgripper_Arm_Set_Order(Arm,* unsigned short *Order)*

Set the position order of the arm to move it. Once this function is called, the arm will move automatically to the given position using the max_speed parameter. If this value is outside the mechanical limits, the microcontroller will limit it automatically.

Return: -

Parameter: ***Order:*** position that the arm must reach

void *kgripper_Arm_Set_Max_Speed(Arm,* unsigned char *Max_Speed)*

Set the maximum speed used by the arm regulation during a movement.

Return: -

Parameter: ***Max_Speed:*** Maximum speed value (1 = 32,5[°/s]).

void *kgripper_Arm_Set_Search_Limit(Arm,* unsigned char *Search_Limit_Flag)*

Start or stop a search of the arm mechanical limits.

Return: -

Parameter: ***Search_Limit_Flag :*** 1 to start a limit search, 0 to stop it.

4.3.1.2. *Gripper functions*

unsigned char *kgripper_Gripper_Get_Version(Gripper)*

Return the firmware version of the gripper microcontroller.
Return: firmware version (0x00-0xFF)
Parameter: -

unsigned char *kgripper_Gripper_Get_Position(Gripper)*

Read the actual position of the gripper. If an object is gripped, this will indicate its size.
Return: gripper position (0-255)
Parameter: -

unsigned char *kgripper_Gripper_Get_Order(Gripper)*

Read the actual position order of the gripper.
Return: gripper position order (0-255)
Parameter: -

unsigned char *kgripper_Gripper_Get_Speed(Gripper)*

Read the actual speed of the gripper.
Return: gripper speed (0-255)
Parameter: -

unsigned short *kgripper_Gripper_Get_Current(Gripper)*

Read the actual current of the arm motor.
Return: actual gripper motor current (0-1023), where 1 = 0,22[mA]
Parameter: -

unsigned short *kgripper_Gripper_Get_Torque(Gripper)*

Read the maximal current (~Torque) acceptable in the gripper motor. The regulator will limit the motor torque when grabbing an object using this parameter.
Return: max gripper motor current (0-1023), where 1 = 0,22[mA]
Parameter: -

unsigned short *kgripper_Gripper_Get_Resistivity(Gripper)*

Read the value of the object resistivity (only if an object is gripped, other wise value will be 0).
Return: object resistivity (0-1023).
Parameter: -

unsigned char ***kgripper_Gripper_OnTarget(Gripper)***

Read the flag which indicates if the gripper has reached its position order. Actually, this value will be set if the difference between the actual position and the order is less than 2.

Return: 1 if the gripper has reached its position order, and 0 if not.
Parameter: -

unsigned char ***kgripper_Gripper_Object_Detected(Gripper)***

Read the state of the optical barrier to know if an object is between the two fingers.

Return: 1 if the gripper has detected an object, and 0 if not.
Parameter: -

unsigned char ***kgripper_Gripper_Get_Search_Limit(Gripper)***

Read the search limit flag. If set, the gripper is currently searching the mechanical limits. If zero, the search limits protocol is finished, and the gripper works as usual (waiting commands).

Return: gripper Search Limit flag (0 = done, 1 = in progress).
Parameter: -

unsigned char ***kgripper_Gripper_Get_Limits(Gripper)***

Read the open mechanical limit. The closed limit is always 0.

Return: gripper mechanical limit (0-255).
Parameter: -

void ***kgripper_Gripper_Get_Ambiant_IR_Light(Gripper,*** **unsigned short *** ***Amb_IR_Left*** **,** **unsigned short *** ***Amb_IR_Right)***

Read the ambient IR light measured by the two finger sensors. This function uses two pointers to get both results in only one call. You need to pass the address of the variables to the function.

Return: -
Parameter: ***** ***Amb_IR_Left*****:** ambient light measured by the left sensor
***** ***Amb_IR_Right*****:** ambient light measured by the right sensor

void ***kgripper_Gripper_Get_Distance_Sensors(Gripper,*** **unsigned short *** ***Dist_IR_Left*** **,** **unsigned short *** ***Dist_IR_Right)***

Read the distance measured by each IR sensor. The value depends on the distance and the colour. If one of these two parameters are constant, you can use these values to measure the unknown parameter.

Return: -
Parameter: ***** ***Dist_IR_Left*****:** distance measured by the left sensor
***** ***Dist_IR_Right*****:** distance measured by the right sensor

void *kgripper_Gripper_Set_Order(Gripper,* unsigned char *Order)*

Set the position order of the gripper to open or close it. Once this function is called, the gripper will move automatically to the given position using the Torque parameter. If this value is greater than the mechanical limit, the microcontroller will limit it automatically. You can force the gripper to use the torque control by setting the order to 0.

Return: -

Parameter: ***Order:*** position that the gripper must reach (0-255).

void *kgripper_Gripper_Set_Torque(Gripper,* unsigned short *Torque)*

Set the maximum current in the gripper motor used by the regulator in the torque control mode.

Return: -

Parameter: ***Torque:*** Maximum motor current (100-900).

void *kgripper_Gripper_Set_Search_Limit(Gripper,* unsigned char *Search_Limit_Flag)*

Start or stop a search of the gripper mechanical limits.

Return: -

Parameter: ***Search_Limit_Flag:*** 1 to start a limit search, 0 to stop it.

4.3.2. I²C Register summary

In fact, the library access to the gripper functionalities with simple read and writes command to I²C register. If you do not want to use the high-level functions you can access directly to the register of the arm and the gripper. When a parameter size is two bytes, you must read or write the LSB first. A command such as a new order will be executed as soon as the MSB byte is updated.

Arm	Device address:	0x14
Register address	Read / Write	Name
0x30	R	Firmware version
0x31 – 0x32	R	Actual position of the arm
0x33	R	Actual speed of the arm
0x34 – 0x35	R	Actual motor current
0x36	R	On Target flag
0x37 – 0x38	R	Battery voltage
0x39	R	Remaining capacity of the battery
0x3A – 0x3B	R	Mechanical limit of the arm (Ground position)
0x3C – 0x3D	R	Mechanical limit of the arm (high position)
0x3E – 0x3F	R/W	Position order.
0x40	R/W	Maximum speed of the arm
0x41	R/W	Search Limit flag. Set it to start a research.

Table 4.1: Register summary of the Arm microcontroller

Gripper	Device address:	0x15
Register address	Read / Write	Name
0x30	R	Firmware version
0x31	R	Actual position of the gripper
0x32	R/W	Position order. Set to 0 to close the gripper.
0x33	R	Actual speed of the gripper.
0x34 – 0x35	R	Actual motor current
0x36 – 0x37	R/W	Maximum current allowed in the motor
0x38 – 0x39	R	Object resistivity
0x3A – 0x3B	R	Ambient light measured by the left sensor
0x3C – 0x3D	R	Ambient light measured by the right sensor
0x3E – 0x3F	R	Distance measured by the left sensor
0x40 – 041	R	Distance measured by the right sensor
0x42	R	Object detection (0 = no object)
0x43	R	On Target flag
0x44	R/W	Search Limit flag. Set it to start a research.
0x45	R	Mechanical limit of the gripper (max opening)

Table 4.2: Register summary of the Gripper microcontroller

4.4. Software example

Here’s a small example of software controlling the Gripper. Once this example is compiled and executed on your KheperaIII, the gripper will open its fingers, move to the ground position, and wait that an object is detected by the optical barrier. As soon as the object is between the two fingers, the gripper will grab it, move to the high position and print the resistivity value of the object.

```
/*!
 * \file   kgripper_small_ex.c  Khepera3 Gripper small example software
 *
 * \brief
 *         This is an small application which control the gripper to grip an object
 *
 *
 * \author   Frederic Lambercy (K-Team SA)
 *
 * \note     Copyright (C) 2010 K-TEAM SA
 * \bug      none discovered.
 * \todo     nothing.
 */

#include <korebot/korebot.h>

/*! handle to the various Gripper devices (knet socket, i2c mode)
 */
static knet_dev_t * Arm;
static knet_dev_t * Gripper;

int main( int arc, char *argv[])
{
 unsigned short Min_Position, Max_Position, Data16;

 /* Init the Gripper library */
 kgripper_init();
 /* open the both gripper and arm sockets and store the handle in their respective pointers */
 Arm = knet_open( "Kgripper:Arm" , KNET_BUS_I2C , 0 , NULL );
 Gripper  = knet_open( "Kgripper:Gripper" , KNET_BUS_I2C , 0 , NULL );
 printf("Khepera3 Gripper example (C) K-Team S.A\r\n");
 printf("Get an object\n");
 kgripper_Gripper_Set_Order( Gripper,  kgripper_Gripper_Get_Limits(Gripper));          // Open the gripper to its maximal limit
 usleep(10000);                                                                        // Wait at least that a control loop is made
 while(kgripper_Gripper_OnTarget(Gripper) == 0);                                       // Wait that the gripper is on position
 kgripper_Arm_Get_Limits(Arm, &Min_Position ,  &Max_Position );
 kgripper_Arm_Set_Order( Arm, Max_Position);                                           // Move the arm to the ground
 usleep(10000);                                                                        // Wait at least that a control loop is made
 while(kgripper_Arm_OnTarget(Arm) == 0);                                               // Wait that the arm is on position
 while(kgripper_Gripper_Object_Detected(Gripper) == 0);                                // Wait an object
 kgripper_Gripper_Set_Torque( Gripper, 600);                                           // Set the maximal torque
 kgripper_Gripper_Set_Order( Gripper, 0);                                              // Close the gripper
 while(kgripper_Gripper_Get_Current( Gripper) < 600);                                  // Wait that the object is grip
 kgripper_Arm_Set_Order( Arm, Min_Position);                                           // Move the arm up the Robot
 usleep(10000);                                                                        // Wait at least that a control loop is made
 while(kgripper_Arm_OnTarget(Arm) == 0);                                               // Wait that the arm is on target
 Data16 = kgripper_Gripper_Get_Resistivity( Gripper);
 printf(" Object resistivity  %u\n", Data16);
}
```

5. 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 afore mentioned period, Customer must request help on K-Team Internet forum on http://www.k-team.com/kforum/ 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.
Rue Galilee 9
1400 Yverdon-les-Bains
Switzerland