D4.4 Software package:
Khepera III drivers for Player/Stage

User's/Programmer's Manual for the Software package

Planned date: M24
Actual delivery: 2009-April-06

K-Team SA,
Julien Tharin
jtharin@k-team.com

Contents

1. Introduction:

This document consists of the User’s/Programmer’s manual for the software driver for the Khepera III robot [1], running with Player/Stage.

The presentation and the method used to build this Player/Stage software driver is described in [2]. The driver software can be found from the Wiki website of the Guardians project [6].

Player/Stage is a free robotic framework software and simulator. It was developed by the University of Southern California. It is used in the Guardians project as the main upper level software for all the different robots [3]. Player is a TCP/IP server running on the robot. It furnishes standard interfaces to the different robot components through a network. Stage, which is a Player plug-in, is a 2 D simulator for robots.

2. Implementation:

The driver developed for the Khepera III robot has two different versions:

- A version used for doing simulation.
- Another version running on the real robot.

2.1 Version for simulations purpose:

This version of the software driver is for simulating the Khepera III robot in the Player/Stage environment.

The required hardware and software to install and use the driver are described below:

I. Required hardware:

- Computer with Linux operating system (kernel version 2.6)

II. Required software:

- Player/Stage version 2.1.1 installed on the computer [7]

The files created for the simulator specific to the Khepera III are listed below in table 2.1.

Filename	description
bitmaps\cave.png	map of the virtual world
bitmaps\kh3.png	Image of the virtual Khepera III
kheperaIII.cfg	Player server configuration file
kheperaIII.inc	Definitions of robot devices
kheperaIII.world	Simulation parameters
sonarobstacleavoid.cc	Example of client doing obstacle avoidance
sonarobstacleavoid	Client executable

Table 2.1: files description

2.1.1 Configuration

The configuration of the simulator is done with text files having a specific format. They are parsed by Player/Stage.

In *KheperaIII.cfg*, the simulator is loaded then the components available to the clients are described (fig 2.1).

```
# Desc: Player sample configuration file for controlling Stage devices
# Author: Frederic Benninger <frederic.benninger@lausanne.ch>

# Date: 06.03.2007
# load the Stage plugin simulation driver
driver
(
name "stage"
provides ["simulation:0" ]
plugin "libstageplugin"

# load the named file into the simulator
worldfile "kheperaIII.world"
)


# Create a Stage driver and attach position2d and Sonar(IR) interfaces
# to the model "myKh3"
driver
(
name "stage"
provides ["position2d:0" "sonar:0"]
model "myKh3"
)
```

Fig 2.1 KheperaIII.cfg configuration file

2.1.2 Virtual world configuration file

The configuration of the simulated 2D world is defined in *kheperaIII.world* (fig 2.2). It includes simulation step size, window size, position, … .

An image describing impassible obstacles is used as map (fig 2.3).

```
# defines 'map' object used for floorplans
include "map.inc"

# defines 'KheperaIII' robot
include "kheperaIII.inc"

# the name of the world, as displayed in the window title bar
```

```
name "[worldfile_test_KheperaIII]"
# the amount of real-world time the simulator will attempt to spend on each simulation cycle
interval_real 100
# the length of each simulation update cycle in milliseconds.
interval_sim 100
# the amount of real-world time between GUI updates
gui_interval 100
# specifies the resolution (m) of the underlying bitmap model.
resolution 0.005

# configure the GUI window
window
(
size [600.000 450.000 ]
center [0.0 0.0]
scale 0.015
)

# load an environment bitmap
map
(
bitmap "bitmaps/cave.png"
size [8.5 5.8]
name "cave"
)

# create a kh3 robot
khepera
(
name "myKh3_1"
color "blue"
pose [-1 0 0]
)

model
(
size [0.3 0.2]
color "green"
pose [0.000 -1.250 0.000]
)
```

Fig. 2.2 kheperaIII.world configuration file

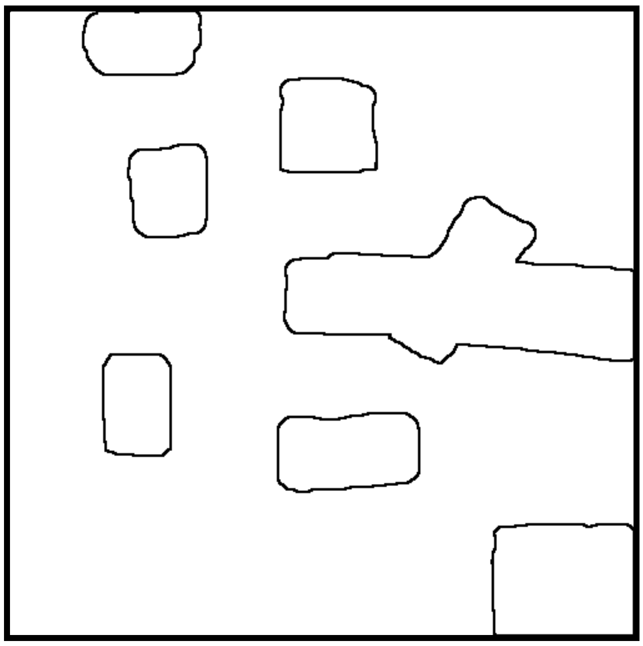


Fig. 2.3 World map

2.1.3 Robot geometry and sensors positions

In the *KheperaIII.inc*, the robot definition is described. It contains the number and position of the IR sensors (Fig 2.4).

```
define khepera_ir ranger
(

# The number of range transducers
scount 9

# Poses of the IRs (m m deg for each one)
spose[0] [-0.043 0.054 128]
spose[1] [0.019 0.071 75]
spose[2] [0.056 0.050 42]
spose[3] [0.075 0.017 13]
spose[4] [0.075 -0.017 -13]
spose[5] [0.056 -0.050 -42]
spose[6] [0.019 -0.071 -75]
spose[7] [-0.043 -0.054 -142]
spose[8] [-0.061 0 180]
# define the size of each TCRT5000 transducer [xsize ysize] in meters
# Has no effect on the data, but controls how the sensor looks in the Stage
ssize [0.007 0.017]
# define the field of view of each transducer [range_min range_max view_angle]
# view_angle has no effect on player/stage 2.0.1
sview [0 0.20 20]
# per-transducer version of the sview property. Overrides the common setting.
sview[9] [0.02 0.04 3]
)
```

Fig 2.4 IR sensors definitions in KheperaIII.inc file

The Ultrasonic sensors (US) of the robot are described in an analogue way to the infrared sensors (fig 2.6).

```
define khepera_us ranger
(

# The number of range transducers
scount 5

# Poses of the US (m m deg for each one)
spose[0] [0.000 0.075 90]
spose[1] [0.053 0.053 45]
spose[2] [0.075 -0.000 0]
spose[3] [0.053 -0.053 -45]
spose[4] [0.000 -0.075 -90]
# define the size of each 400ST100 and 400SR100 [xsize ysize] in meters
# Has no effect on the data, but controls how the sensor looks in the Stage
ssize [0.010 0.020]

# define the field of view of each transducer in meters
# [range_min range_max view_angle] view_angle has no effect on
#player/stage 2.0.1
sview [0.20 4 35]
)
```

Fig 2.5 US sensors definitions in KheperaIII.inc file

Finally the robot size and motor type, here selected as a differential drive for the Khepera III, and an image representing the robot in the simulation was (Fig 2.6 and 2.7).

```
define khepera position
(
# drive "diff","omni","car"
# differential steering model
drive "diff"
# actual size
size [0.156 0.156]
# the khepera's center of rotation is at its center of
origin [0.0 0.0 0]

bitmap "bitmaps/kh3.png"

# use the ir array defined above
khepera_ir()
# and the us array
khepera_us()
)
```

Fig 2.6 robot description in KheperaIII.inc file

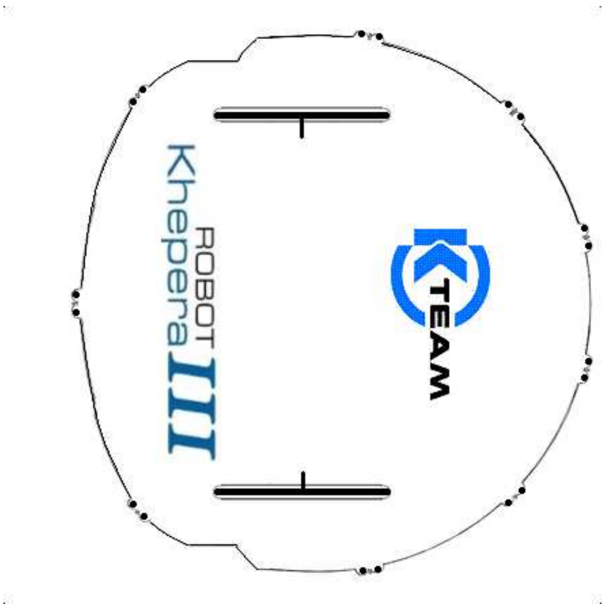


Fig 2.7 Khepera III robot image for simulator

2.1.4 Map defining simulated world limits

In the *map.inc* file, the definition of the simulated world are defined

```
define map model
(
# sombre, sensible, artistic
color "black"

# most maps will need a bounding box
boundary 1
gui_nose 1
gui_grid 1
gui_movemask 0
gui_outline 0
gripper_return 0
)
```

Fig 2.8 virtual world definition for simulator

2.1.5 Client compilation

After having installed Player/Stage, the client software ***sonarobstacleavoid*** can be compiled with the following command:

```
g++ -o sonarobstacleavoid `pkg-config --cflags playerc++` sonarobstacleavoid.cc `pkg-config --libs playerc++`
```

2.1.6 Running the simulation

You can run a simulation with the commands lines below:

Export the library path:

```
export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
```

Run the player server:

```
player KheperaIII.cfg&
```

Open the simulation window (fig 2.9): with the command below

```
playerv –p 6665 --position2d --sonar:0 --sonar:1
```

Run the player client, which contains an algorithm of obstacle avoidance:

```
./sonarobstacleavoid
```

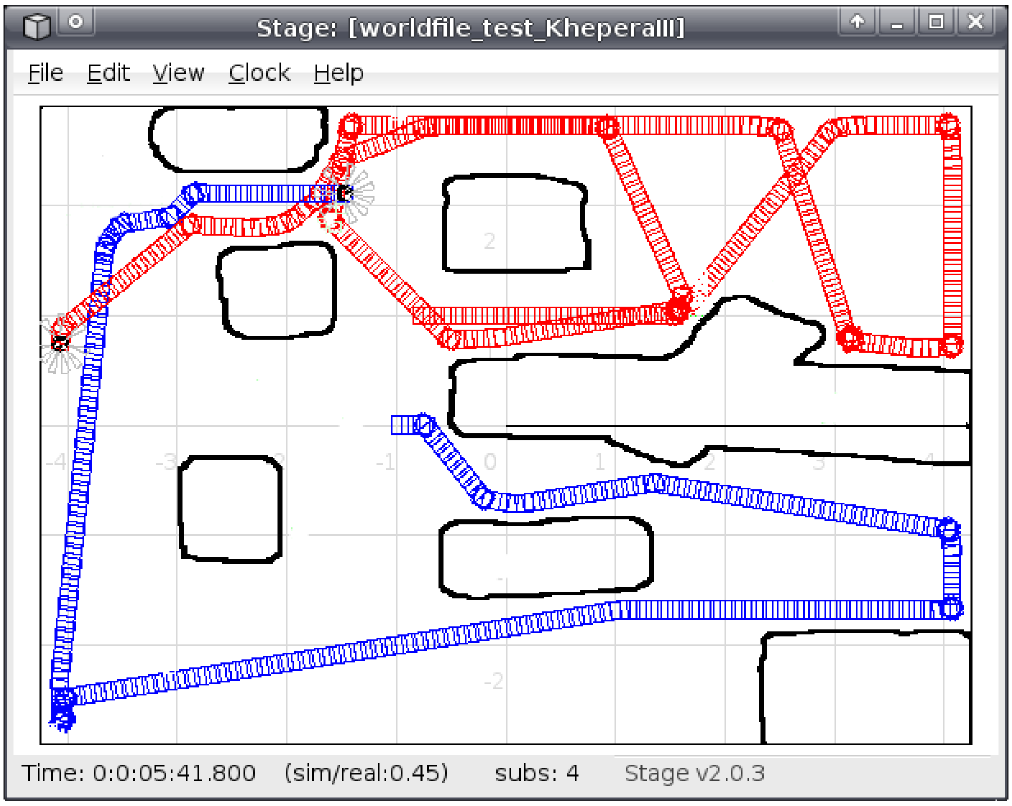

Stage: [worldfile_test_KheperaIII]
File Edit View Clock Help
2
-4
-3
-1
0
1
2
3
-2
Time: 0:0:05:41.800 (sim/real:0.45) subs: 4 Stage v2.0.3

Fig. 2.9 Simulation window

2.2 Version running on the real robot:

This version of the software driver is for using the real Khepera III robot.

2.2.1 Concept and principle of functioning of the driver

A Player driver is a class inherited from the Driver class. When the Player server is launched, the configuration file is read, the KheperaIII driver launched. The class constructor is called, which declares the different devices defined in the configuration file. Player is listening on the network tcp/ip ports and wait until client are connected (fig 2.10).

At every request from the client to use an interface, the subscribe method is invoqued. The interface is used to define the synthax and the semantic of all the message exchanged with entities of a same class. The interface defined by the Guardians project is depicted in table 2.2.

Khepera III	Player/Stage interface	Message	Description
Motor speed cmd/ wheel odometry robot geometry	Position2D	player_position2d_data_state	return speed, odometry data, motor state
		player_position2d_reg_get_geom	return robot geometry
		player_position2d_cmd_vel	set a new speed value
IR sensors	IR	player_ir_data_range	return IR sensors value
		player_ir_pose	return IR sensors geometry
Sonar sensors	Sonar	player_sonar_data_range	return US sensors value
		player_sonar_pose	return US sensors geometry

Table 2.2 Player/Stage interfaces and messages

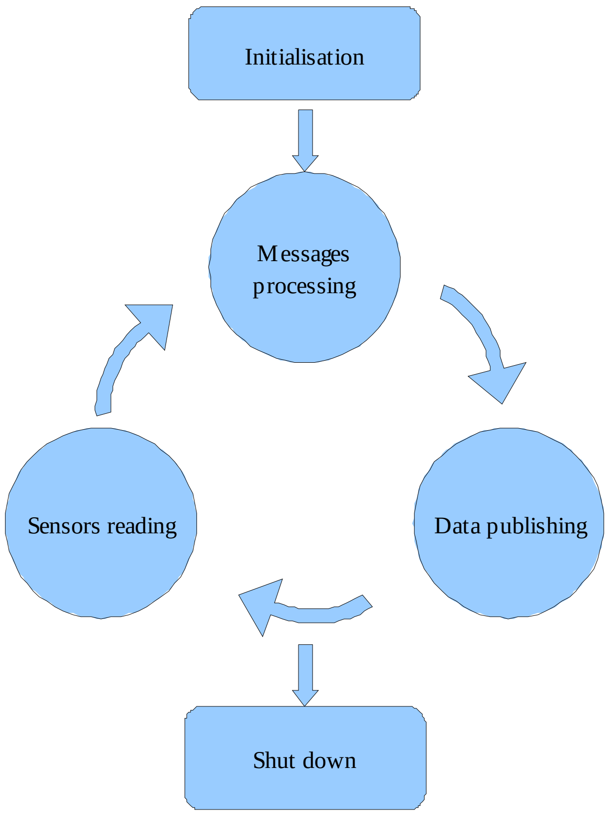


Fig 2.10 Player/Stage driver functioning principle

Two structures are used, defining the robot geometry and another containing the sensors data, which will be send to the client.

At the first request, the *setup* method is called to initialise the hardware then the *main* method is launched in a separate thread.

The unsubscribe method is invoqued when the client does not need anymore the ressource and deconnects itself from the interface.

When all the interfaces are freed, the shutdown method is call, which stops the thread and stops the robot.

Here after are the two types of structure defined in *KheperaIII.h*. These structures are derived from standards structures defined in Player (fig. 2.11 and 2.12).

```
struct player_kh3_geom
{
double scale;
double encoder_res;
player_position2d_geom_t position;
player_ir_pose_t ir;
player_sonar_geom_t sonar;
// double * ir_calib_a;
// double * ir_calib_b;
} __attribute__ ((packed));

typedef struct player_kh3_geom player_kh3_geom_t;
```

Fig 2.11 player_kh3_geom in KheperaIII.h

```
struct player_kh3_data
{
player_position2d_data_t position; //.pos .vel .stall
player_ir_data_t ir;
player_sonar_data_t sonar; // .ranges_count .ranges
player_power_data_t power;
//player_aio_data_t aio;
} __attribute__ ((packed));

typedef struct player_kh3_data player_kh3_data_t;
```

Fig 2.12 player_kh3_data in KheperaIII.h

2.2.2 Data read from the configuration file

In the *libplayercore* of Player, the *configFile* class includes methods to parse the configuration file.

Here after is an example reading a real number:

```
kh3Geom->scale = cf->ReadFloat(section, "scale_factor", KH3_DEFAULT_SCALE);
```

And here is the way for reading the IR parameters.

```
for (unsigned int i = 0; i < kh3Geom->ir.poses_count; ++i)
{
kh3Geom->ir.poses[i].px =
cf->ReadTupleFloat(section,"ir_poses",3*i+0,0) * kh3Geom->scale;

kh3Geom->ir.poses[i].py =
cf->ReadTupleFloat(section,"ir_poses",3*i+1,0) * kh3Geom->scale;
kh3Geom->ir.poses[i].pa =
DTOR(cf->ReadTupleFloat(section,"ir_poses",3*i+2,0));
}
```

Fig 2.13 reading IR parameters

These data will be returned to the client from the following way by using the PLAYER_IR_POSE message.

```
Publish(ir_addr, resp_queue,
PLAYER_MSGTYPE_RESP_ACK,
PLAYER_IR_POSE,
(void*)&kh3Geom->ir, sizeof(kh3Geom->ir),NULL);
```

Fig 2.14 publishing IR parameters

For returning the position to the client, the *Publish* method is used (fig 2.15).

```
Publish(position_addr, NULL,
PLAYER_MSGTYPE_DATA,
PLAYER_POSITION2D_DATA_STATE,
(void*)&kh3Data->position, sizeof(kh3Data->position), NULL);
```

Fig 2.15 publishing position parameters

2.2.3 Installing, using and modifying/rebuilding the driver

Thereafter the instructions for installing, using and modifying/rebuilding the driver Player-Stage for Khepera III are described.

Three parts are available:

- Installing the compiled version and usage
- Modifying and rebuilding the driver of Player
- Rebuilding the player server for Khepera3 (for example adding a build-in driver)

2.2.3.1 Installing and using Player on the Khepera 3

The required hardware and software to install and use the driver are described below:

I. Required hardware:

To use the Khepera 3 driver, you must have these components:

- Khepera 3 with KorebotLE
- Computer and wireless (Wifi) network

II. Required software:

- linux operating system version 2.6.x on the computer
- Player/Stage version 2.1.1 installed on the computer [7]
- Kernel 2.6.x on the KorebotLE
- from the binaries directory of the Khepera 3 Player/Stage driver [9]:
  - KheperaIII.cfg
  - KheperaIII.so
  - libltdl3_1.5.10-r3_armv5te.ipk
  - libstdc++6_4.1.1-r16_armv5te.ipk
  - player_2.1.1-r0_armv5te.ipk

Installation procedure:

Here after the procedure for installing and using the driver is described.

1) Power up the Khepera 3.

2) Establish a network connection with the Khepera 3 either by wifi [4.1] or with the serial cable (see [8] chapter 4.2).

3) Copy and install the 3 following packages on the korebot:

   command for copying with ssh (see Annexe 4.3):

   scp FILE root@KHEPERA_IP_ADDRESS:/home/root

   installation procedure : ***ipkg install PACKAGE_NAME***

   packages files:

   - ***libstdc++6_4.1.1-r16_armv5te.ipk***
   - ***libltdl3_1.5.10-r3_armv5te.ipk***
   - ***player_2.1.1-r0_armv5te.ipk***

Remark: If there is not enough free space, delete each package after installation.

4) Copy KheperaIII.so and KheperaIII.cfg on the korebot

Usage:

1) On the KoreBot, launch the server:

player KheperaIII.cfg

2) On the computer with Player 2.1.1 installed in a terminal, export the library path and launch the viewer:

export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH

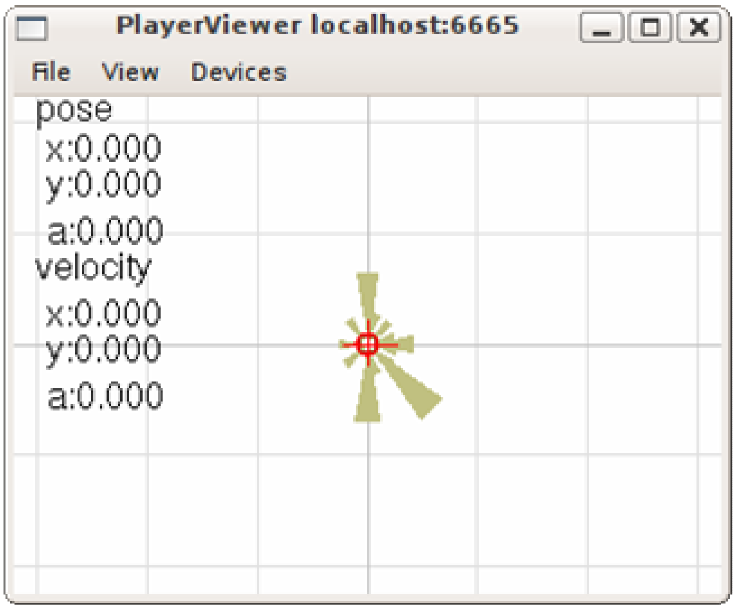
playerv -h IP_ADDRESS_OF_THE_ROBOT --ir:0 --position2d:0 --sonar:0 --power:0

3) You can drive the robot with the playerv interface (fig. 2.16):

Go to *"Devices/Position2d"* and select *"Command"*

=> A red cross appears on the robot.

Move this red cross to drive the robot.



```
pose
 x:0.000
 y:0.000
 a:0.000
velocity
 x:0.000
 y:0.000
 a:0.000
```

Figrue 2.16: PlayerViewer graphical interface

2.2.3.2 Modifying and rebuilding the driver from source

To modify and rebuild the Khepera 3 driver, you must have these components:

i. Required hardware:

- Khepera 3 with KorebotLE
- Computer and wireless (Wifi) network

ii. Required software:

- linux operating system version 2.6.x on the computer
- Player/Stage version 2.1.1 installed on the computer [7]
- Kernel 2.6.x on the KorebotLE
- Korebot 2.6 development toolchain installed on the computer (see [10] for installation procedure)
- from the source directory of the directory of the Khepera 3 Player/Stage driver [9]:
  - khepera3toolbox-2008-05-05.zip
  - khepera3toolbox.diff
  - libtool1.5.10_lib.tar.bz2
  - player2.1.1_include.tar.bz2
  - player2.1.1_lib.tar.bz2
  - player_driver_dev.tar.bz2
  - kh3-plugin-2009.03.16.tar.bz2

Build procedure

Here after the procedure for rebuilding the driver is described.

1) Extract the 3 following files in the base directory / (you must be root).

   These files are installed in the cross-compiler sub-directory.

   Command for extracting any compacted file into the base directory:

   tar -xjvf COMPACTED_FILE.tar.bz2 -C /

   files:

   - ***player2.1.1_include.tar.bz2***
   - ***player2.1.1_lib.tar.bz2***
   - ***libtool1.5.10_lib.tar.bz2***

2) Extract the ***player_driver_dev.tar.bz2*** file and enter into it. It will be your base directory:

   tar -xjvf player_driver_dev.tar.bz2

3) Modify ***KTEAM_HOME*** variable in the ***env.sh*** file to point this directory

4) Extract the khepera toolbox in the base directory:

   unzip khepera3toolbox-2008-05-05.zip

5) In the toobox directory, patch the toolbox :

   patch -p1 < khepera3toolbox.diff

6) Source the env.sh file to have access to the cross-compiler

   source env.sh

7) Compile the following modules of the toolbox:

   - ***i2cal***
   - ***khepera3***
   - ***odometry_goto***
   - ***odometry_track***

   commands for each module ***MODULE_NAME***:

   cd Modules/MODULE_NAME

   make

8) Extract ***kh3-plugin-2009.03.16.tar.bz2*** in the base directory.

   You can start here to modify the driver. The description is explained in chapter 2.2.1.

9) In kh3-plugin directory, compile by running the command : ***make***

   If there is any problem, verify the Makefile.

10) Then transfer the files KheperaIII.so and KheperaIII.cfg to the Korebot.

2.2.3.3 Rebuilding the player server for Khepera3

i. <u>Required hardware:</u>

To modify and rebuild the Khepera 3 driver, you must have these components:

- Khepera 3 with KorebotLE
- Computer and wireless (Wifi) network

ii. <u>Required software:</u>

- full development tools of korebot 2.6 [10]
- Kernel 2.6 on the Korebot
- from the source directory of the directory of the Khepera 3 Player/Stage driver [9]:
    - bitbake recipe ***player-oe.tar.bz2***

Build procedure

Here after the procedure for rebuilding the Player server for Khepera III is described.

1) Install the full development tools for korebot 2.6, if not already done.

2) Extract the recipe player-oe.tar.bz2 in /usr/local/korebot-oetools-1.0

   You can modify it to add a build in driver, by example ***urglaser***:

   Edit the variable ***EXTRA_OECONF*** in the file ***/usr/local/korebot-oetools-1.0/custom/packages/player/player_2.1.1.bb***:

   EXTRA_OECONF = "--disable-alldrivers   --enable-urglaser"

3) Compile player: in ***/usr/local/korebot-oetools-1.0/custom***, execute:

   start-build.sh build player

   =>The ouput packages will be in:

   /usr/local/korebot-oetools-1.0/build/tmp/deploy/glibc/ipk/armv5te/

    - ***libstdc++6_4.1.1-r16_armv5te.ipk***
    - ***libltdl3_1.5.10-r3_armv5te.ipk***
    - ***armv5te/player_2.1.1-r0_armv5te.ipk***

4) Copy and install the packages, as described in chapter 2.2.3.1.

3. References

[1] K-Team, Switzerland, Khepera III homepage, http://www.k-team.com/kteam/index.php?site=1&rub=22&page=197&version=EN

[2] Julien Tharin, K-Team, Switzerland, D4.4 *"Software package: Khepera III drivers for Player/Stage"*

[3] Rafa López, Robotnik Automation S.L.L, Spain, D4.1/D4.2: *"Preliminary System Architectural Design and Software Architecture"*

[4] Angström distribution of OpenEmbedded Linux: *http://www.angstrom-distribution.org/building-%C3%A5ngstr%C3%B6m*

[5] Leo Nondedeu, UJI Spain, D2.1.3-4: *"The navigation capabilities of the robot platforms for the selected scenario and common API specifications for the robot platforms"*

[6] SHU University,UK, Guardians Wik, Khepera III page: *http://vision.eng.shu.ac.uk/guardians/index.php/Khepera_III*

[7] Player/Stage website : *http://playerstage.sourceforge.net/*

[8] K-Team, Switzerland, Khepera III user's manual: *http://ftp.k-team.com/KheperaIII/Kh3.Robot.UserManual.2.2.pdf*

[9] K-Team, Switzerland, Khepera III Player/Stage software driver: *http://ftp.k-team.com/KheperaIII/player_stage*

[10] K-Team, Switzerland, Korebot software development toolchain 2.6: *http://ftp.k-team.com/korebot/toolchain-2.6-betaV0.1*

4. Annexes

In this part, the detailed descriptions of several tools and helpful commands are explained.

4.1 Using a Wireless compact flash card

Two wireless compact flash models are supported. The card name and its driver are listed below:

A) Ambicom WL1100C-CF with pxa2xx_cs driver module

B) Ambicom WL5400G-CF with libertas_cs driver module

Remark:

The following instructions are for the wireless compact flash A) **Ambicom WL1100C-CF**. With the model B) **WL5400G-CF**, you have to update your kernel and the driver as described at:

http://ftp.k-team.com/korebot/kernel/kernel2.6.25.7-kb1/Kernel2.6.25-7-kb1_WifiG-support.txt

Then you may replace the wireless port name ***wlan0*** by ***eth0*** in the following instructions.

1) insert a Wireless compact flash card in the Korebot

2) load the module by typing: ***modprobe pxa2xx_cs***

You may load the Wifi module automatically by adding pxa2xx_cs in the file ***/etc/modules***

You can use the following command echo to add the module name to the file: ***echo pxa2xx_cs>>/etc/modules***

3)

- i) WEP support

a) for configuring the wifi connection, type:

iwconfig wlan0 essid YOUR_SSID_OF_NETWORK

b) if the network is secured, enter the key by typing :

iwconfig wlan0 key YOUR_KEY

c) then set an ip address to the korebot:

ifconfig wlan0 YOUR_IP_ADDRESS

d) configure the gateway by entering the gateway ip:

```
route add default gw YOUR_GATEWAY_IP wlan0
```

e) insert the local domain name in /etc/resolv.conf

```
echo search YOUR_LOCAL_DOMAIN_NAME>>etc/resolv.conf
```

f) and the dns server

```
echo nameserver YOUR_DNS_SERVER_IP_ADDRESS>>
/etc/resolv.conf
```

You can also create a file in */etc/network/if-pre-up.d* named *wireless* to have these settings saved.

Put the following into it:

```
#!/bin/sh
ifconfig wlan0 up
iwconfig wlan0 essid YOUR_SSID_OF_NETWORK
iwconfig wlan0 key s:YOUR_KEY
ifconfig wlan0 YOUR_IP_ADDRESS
route add default gw YOUR_GATEWAY_IP wlan0
```

And the following in a file named */etc/resolv.conf*:

```
search YOUR_LOCAL_DOMAIN_NAME
nameserver YOUR_DNS_SERVER_IP_ADDRESS
```

ii) WEP, WPA and other encryptions:

a) create a file named */etc/wpa_supplicant/wpa_supplicant.conf* and insert your selected wireless encryption:

WEP:

```
#Shared WEP key connection (no WPA):
network={
ssid="YOUR_SSID"
key_mgmt=NONE
wep_key0="YOUR_WEP_KEY"
auth_alg=SHARED
wep_tx_keyidx=0
priority=5
}
```

WPA-TKIP:

- see instructions at:
http://ftp.k-team.com/korebot/kernel/modules/wpa-tkip/wpa-tkip_support.txt

```
#/etc/wpa_supplicant/wpa_supplicant.conf

with WPA-PSK TKIT:

network={

ssid="YOUR_SSID"

psk="YOUR_PASS_KEY"

key_mgmt=WPA-PSK

group=TKIP

pairwise=TKIP

proto=WPA

priority=5

}
```

You can check the following link for other encryptions:
http://hostap.epitest.fi/wpa_supplicant/

b) run the daemon controlling the wireless connection with the following command:

```
wpa_supplicant -c /etc/wpa_supplicant/wpa_supplicant.conf -i wlan0 -Dwext -B
```

You can also add the command above to a file in */etc/network/if-pre-up.d* named *wireless*:

```
#!/bin/sh
ifconfig wlan0 up
ifconfig wlan0 YOUR_IP_ADDRESS
route add default gw YOUR_GATEWAY wlan0
wpa_supplicant -c /etc/wpa_supplicant/wpa_supplicant.conf -i wlan0 -Dwext -B
```

4.2 Connecting to the Korebot with network over usb cable

a) launch the usb module on the Korebot:

```
modprobe g_ether
```

b) connect the computer to the Korebot USB slave port with an USB cable

c) configure the usb port on the Korebot:

```
ifconfig usb0 10.0.0.2/24
```

d) on the computer, configure also the usb port (you must be root, or use sudo):

```
ifconfig usb0 10.0.0.1/24
```

4.3 Transferring files using scp (ssh)

1) Establish a network connection between the computer and the korebot (using Wifi see chapter "Using a Wireless compact flash card", or using Ethernet over usb see chapter "Connecting to the Korebot with network over usb cable"

2) Execute the following command, where FILE, is the file to transfer, KOREBOT_IP the Korebot ip address.

```
scp FILE root@KOREBOT_IP:/home/root
```