0. Introduction

khep_core is a basic package for communication with and control of khepera robots through the serial ports. It was initially developed on an Intel Pentium II box running RedHat Linux 6.0 using gnu c++. It has been used successfully, without modification, under RedHat Linux 7.1.

The lower-level serial communications code was modified and adapted from the khep_serial package authored by M. von Holzen, L. Tettoni, J.-Y. Tigli, and O. Michel, 0.1 ter 21/09/95 (in C), and which can be acquired from http://artscl.wustl.edu/~philos/pnp/robotlab/robotlab.html

This document is intended to serve as a brief overview of the structure and function of classes contained in the khep_core package. Section 1 contains a description of the general organization of the khep_core classes. The interface class is summarized in section 2. Finally, a summary of commands implemented in the khepera class appears in section 3. For more detail, please consult the code directly. Section 4 may contain further information on changes since version 0.5, if any changes have been made.

Please be forewarned that much of this code has not been extensively tested, so there is a possibility of bugs. Please let the author know of any bugs, fixes, or suggestions.

The author thanks the Department of Philosophy at Washington University for funding (summer 1999) contributing to the completion of this project.

License Agreement

Khep_core is a freeware public domain source code written by Whit Schonbein with portions based on code written by M. von Holzen, L. Tettoni, J.-Y. Tigli, and O. Michel. The author(s) cannot be held responsible for any software or hardware damage caused by the use of this code. Permission is hereby granted to copy this package for free distribution. Commercial use is forbidden. If you publish any academic work based upon experiments utilizing the khep_core classes, please include

---

1 This documentation is not up-to-date. Several small bugs have been fixed in the code, and this should not effect the documentation for the most part. – Whit Schonbein, August 2003.
a reference (e.g., in a footnote, bibliography, or acknowledgements) mentioning the khep_core's World Wide Web address: 
http://artsci.wustl.edu/~philos/pnp/robotlab/robotlab.html. Finally, permission is hereby granted to modify this package for personal (i.e., non-commercial) use, provided the modified code is not distributed without documentation of its modification. Please let Whit Schonbein know (whit@twinearth.wustl.edu) of any changes/improvements made.

1. General Architecture

The khep_core package consists of several classes. At the highest level (loosely speaking) is the khepera class. The khepera class contains code to implement the various commands as described in the Khepera User Manual, version 4.06 (K-Team SA, 1995). Embedded in each instance of the khepera class is an instance of a lower-level serial port communication class called the interface class. The khepera class is responsible for building commands in the form of text strings and sending them to the interface class. The interface class is responsible for sending these commands to the physical robot. Furthermore, the interface class takes any responses (e.g., sensor readings) and makes them available to the khepera class, which stores these responses in various buffers, where they are available to whatever code is utilizing the khepera class instance. Figure 1 depicts the relationships between classes. The dashed line indicates which classes are contained in the khep_core package. Arrows indicate directions of interaction between classes. The khep_core code is concerned solely with the khepera class and the serial interface class; see the khep_vehicle package for one example of how controlling classes can use the khep_core classes to control Khepera robots.

A note on the design: Everything is public. No attempt has been made to modularize data access.

2. The Interface Class
The interface class consists of routines to open a serial port connection to a physical robot, routines to pass commands to the robot, and routines to receive data from the robot. The role of the interface class is to serve as an intermediary between the khepera class and the physical robot. It is also responsible for opening and initializing the serial port.

The khepera class communicates with the interface class via the talk (char *) function, an interface class member function. The value of the char * argument is placed in the send buffer (send_buf) within the interface class, and also sent to the physical robot. Any response from the robot is placed in the ‘receive’ buffer (recv_buf) within the interface class, as well as being returned by the talk (char *) function for possible use by the khepera class, or by a controller class.

A brief summary of the available functions is as follows:

### 2.1 interface class functions

**interface()** : Default constructor. Uses the default baud rate as specified in KHEP_DEFAULT_BAUD, and the default port ttyS0.

**interface(char * port)** : Constructor. **port** is the serial port to be used (e.g., “/dev/ttyS0”). The default baud rate (B38400) is specified by the KHEP_DEFAULT_BAUD constant in khep_core5.cc.

**interface(char * port, long baud)** : Constructor. **port** is the serial port to be used. **baud** is the baud rate to be used.

**int serial_configure()** : Sets the attributes of the serial port. Returns **TRUE** if configuration is successful, **FALSE** otherwise.

**int serial_readline()** : Reads a line of characters from the serial port to a maximum of KHEP_MSG_SIZE bytes, until an error occurs, or until a newline is encountered (Khepera robots terminate their output strings with newline). The string is stored in the ‘receive’ buffer (recv_buf). The function returns either the number of bytes read (if the read is successful), or the error code returned by the failed read attempt (if the read is unsuccessful).

**int serial_drain(int verbose)** : Empties the input buffer and discards the contents. Doesn’t return anything at all (?). If **verbose** is **TRUE**, the contents of the buffer are displayed during draining.

**char * talk(char * send)** : Stores a copy of **send** in the send buffer (send_buf), transmits the string to the Khepera via the serial port, and places the response generated by the robot in the receive buffer (recv_buf). The function returns the Khepera’s response. Finally, the function verifies that the command was processed correctly by the Khepera by comparing the contents of the send buffer with the contents of the receive buffer, which should match if the communication was successful.
void initialize() : Initializes some of the data structures in the class.

void clear_recv_buf(char c) : Sets each element of the recv_buf array to character c.

void clear_send_buf(char c) : Sets each element of the send_buf array to character c.

char * get_recv_buf() : Returns a pointer to the receive buffer.

char * get_send_buf() : Returns a pointer to the send buffer.

3. The Khepera Class

The khepera class encapsulates most of the commands available for interaction with the robot. It contains arrays to hold the most recent set of sensor values (infrared sensors, ambient sensors, vision turret pixels, position, and speed sensors), as well as an instance of the interface class, through which communication with the robot is achieved. Note that the khepera class can be easily modified to work with a simulation by replacing the interface class instance with an interface to a simulated robot.

Most of the commands listed in the Khepera User Manual, version 4.06, are implemented in the khepera class, although several are not. Furthermore, many of the commands listed in the K213 Vision Turret User Manual, version 1.0, are also implemented, but again, several are not. Finally, the khepera class includes several additional functions that encapsulate various procedures, e.g. stop() (which stops the robot) and turn_degrees(int) (which rotates the robot the specified number of degrees).

3.1 khepera class functions

Commands currently implemented (v. 0.5) are of the following sorts (grouped into loosely associated classes):

CONSTRUCTORS:

khepera(char * robot_name, char * port) : Constructor. Creates an instance of the khepera class with the name robot_name, and containing an instance of the interface class configured to the default baud rate (see above) and to the serial port specified by the port argument.

khepera(char * robot_name, char * port, long baud) : Constructor. Same as above, except the baud rate is configured to the value specified by the baud argument.

LED LIGHTS:
void led_front_on(), led_front_off(), led_front_toggle(), led_side_on(),
led_side_off(), led_side_toggle() : Turns the indicated LED on or off, or toggles the light.
This command has not been tested.

MISCELLANEOUS COMMANDS:

char * get_recv_buf() : Returns pointer to the receive buffer of the interface class.

char * get_send_buf() : Returns pointer to the send buffer of the interface class.

char * get_name() : Returns pointer to the name of the robot.

void print_sensor_values() : Displays the values of the sensor_values array. The
sensor_values array stores the values of the proximity (infrared) sensors most recently
obtained from the robot during a query of the active infrared or ambient sensors. Used primarily as
a diagnostic during development.

void print_ambient_sensor_values() : Displays the values of the ambient_sensor_values
array.

void clear_sensors() : Clears the values of the sensor_values array.

void verbose(int) : If argument is TRUE (i.e., >0), then the information about the functioning of
the khepera class is displayed during runtime. Default is FALSE.

INFRARED SENSOR COMMANDS:

void get_proximity_sensor_values() : this sends the N command (as described in the
Khepera User Manual) to the interface class instance, which directs it to the serial port. The
results returned from the interface class are parsed and stored in the sensor_values[] array. If
verbose is TRUE, several lines are directed to cout indicating the contents of the send_buf and
the recv_buf in the instance of the interface class contained in the khepera class.

void get_ambient_sensor_values() : this is the same as get_proximity_sensor_values(),
except does not utilize active sensing. Values are stored in the ambient_sensor_values[] array.
It issues command O.

int get_proximity_sensor_value(int) : returns value of a particular sensor (0-7). Sensors are
numbered in accordance with figure 6 in the Khepera user manual, version 4.06, p. 6. To avoid
possible overhead, this function does not query the sensors directly (i.e., it does not issue a
get_proximity_sensor_values() command). Rather, it simply returns the appropriate value
from the sensor_values[] array. In order to get an up-to-date sensor reading, one must first call
get_proximity_sensor_values(). (Alternatively, uncomment the line in the source to enable
the function to update sensor_values[] itself (see khep_core.cc). If verbose is TRUE, the
value of the sensor queried is displayed.

int get_ambient_sensor_value(int) : Same as above, only for the ambient sensors.

void print_sensor_values() : prints the sensor_values and ambient_sensor_values arrays
to cout.

void clear_sensors() : sets all sensor values (int sensor_values[]) to 0.

void clear_ambient_sensors() : sets all sensor values (in ambient_sensor_values[]) to 0.

MOVEMENT COMMANDS:

void get_speed() : reads current speed of left and right motors and stores the values in
speed[left, right]. This is the E command.

int get_speed_left() and int get_speed_right() : returns the appropriate value from
speed[]. Does not call get_speed() itself.

void set_speed(int left, int right) : sets the speeds of the left and right motors (khepera
command D). Stores the requested speeds in speed_goal[] so that the actual speed of the robot
might be compared with the desired speed (using get_speed()). If verbose is TRUE, the function
displays some information concerning the send and receive buffers from the interface class.

void clear_position() : sets the position counters to zero (khepera command G called with
arguments of 0 and 0).

void set_position(int left, int right) : sets the desired values of the position counters to
the specified arguments. Verbose mode displays some data concerning the 'receive' and 'send'
buffers in the interface. Note that there is no range checking implemented yet.

void go_to_position(int left, int right) : tells the khepera to move its wheels until their
respective counters match the values specified in the arguments (khepera command C). There is no
range checking implemented. Furthermore, any other command sent to the khepera while the
khepera is in motion as the result of a C command will stop the execution of the C command. So
some sort of sleep loop may be desired. One possible modification (not currently implemented) is
to implement this in the interface class so that the interface works as a proxy. Commands sent to
the interface will then either be queued, or, if an 'urgent' flag is set, the so marked command will be
allowed through to the physical robot.

void turn_degrees(int deg) : If everything is working correctly, a positive value should rotate
the robot the specified number of degrees clockwise, and a negative value should do the same but
clockwise. This function was adapted directly from the older khep_serial code (see
introduction).
void stop() : stops the robot.

void print_speed() : Displays the values of the speed_read array (i.e., the last speeds sent to the motors).

VISION TURRET COMMANDS

void clear_vision() : The most recent values of the vision turret pixels are stored in the vision_values[] array. This command sets them all to zero.

void vision_read_image() : Get values of each of the 64 pixels and put them in vision_values[]. Pixels are indexed in accordance with figure 2 in the vision turret manual.

int vision_get_pixel_value(int pixel) : Returns the value of the requested pixel. This does not actually query the vision turret; rather, it just returns the value in the appropriate cell of the vision_values[] array. If you wish this function to query the vision turret, uncomment the line included in the khep_core code.

int vision_read_max_int_pixel() : returns the index of the pixel with the maximal light intensity.

int vision_read_light_intensity() : returns the light intensity value of the field of view of the vision module (p. 12 in vision turret manual).