Katana 450

Kurzanleitung

Programmieren mit KNI

Document No: 233490
Version 1.0.0

Neuronics AG, Technoparkstrasse 1, CH-8005 Zürich
Tel: +41 44 445 16 40, Fax: +41 44 445 16 44
www.neuronics.ch
info@neuronics.ch

Inhaltsverzeichnis

1 Überblick

Das Katana Native Interface KNI ist eine Open Source Software Bibliothek zur Ansteuerung des Katana Roboters. KNI ist in C++ geschrieben und so aufgebaut, dass eine Portierung in andere Sprachen und Frameworks einfach ist. Der Code ist plattformübergreifend gehalten und kann sowohl unter Windows mit dem MS Visual C++ Compiler als auch unter Linux mit der GNU Compiler Toolchain kompiliert werden.

Das KNI abstrahiert die unterliegenden Schichten und erlaubt es so, Applikationen für den Katana zu schreiben, ohne in die Details des Systems involviert zu werden. Verbindungsaufbau und Initialisierung des Roboters geschieht mit wenigen Funktionsaufrufen. Das Protokoll zur Steuerung des Roboters vom Computer aus ist komplett abstrahiert. Das KNI beinhaltet eine Implementation der Roboterkinematik und Bahnberechnungsroutinen, um alle Achsen synchron zu steuern und mit dem Endeffektor Bahnen im Raum zu fahren.

Die Offenheit der gesamten Sourcen macht das KNI auch zum idealen Tool für die Forschung und Ausbildung, da die gesamte Implementation nachvollzogen und nach Belieben modifiziert und angepasst werden kann.

2 Build Umgebung

2.1 Die KNI Sourcen

2.1.1 Bezugsquellen

Die aktuellsten Sourcen des KNI können auf der Webseite von Neuronics

```
http://www.neuronics.ch/
```

im Downloadbereich kostenlos bezogen werden. Bestehende Kunden werden jeweils automatisch informiert, wenn eine neue Version zur Verfügung steht.

2.1.2 Installation

2.1.2.1 Linux

In einer Shell in den Ordner wechseln, in den das KNI Archiv heruntergeladen wurde. Das Archiv mit

```
tar -jxvf KatanaNativeInterface-x.x.x.tar.bz2
```

entpacken und mit

```
cd KatanaNativeInterface-x.x.x/
```

in den KNI Ordner wechseln.

Nach Installation der Abhängigkeiten und Tools, beschrieben in Kapitel 2.2, können die Bibliotheken und Demos mit dem Befehl

```
make
```

kompiliert werden.

2.1.2.2 Windows

Den Installer ausführen und den Instruktionen folgen. Der Installationsordner ist

```
C:\Program Files\Neuronics AG\KatanaNativeInterface\
```

Der Installer erstellt eine Programm Gruppe 'Katana Native Interface' im Startmenü mit Links zur 'KNI Visual C++ Solution' und dem 'Demo Applications' Ordner.

Nach Installation der Abhängigkeiten und Tools, beschrieben in Kapitel 2.2, kann die '.sln' Datei im Installationsordner unter 'win32' doppelgeklickt werden, damit sich die Solution in Visual Studio öffnet. Alternativ kann auch 'KNI Visual C++ Solution' im Startmenu unter 'Programme' und 'Katana Native Interface' gestartet werden.

Im Visual Studio sind links die Projekte (Teile von KNI und Demos) aufgelistet. Um das Control Demo zu kompilieren auf 'Demo-Control' rechtsklicken und im erscheinenden Menu 'Erstellen' wählen. Das fertige Programm liegt im Installationsordner unter `demo\control\control.exe`

2.1.3 Beschreibung des Source Trees

Das KNI präsentiert sich nach der Installation als einfacher Source Tree:

```
-rw-r--r-- 1 user user      359 2008-09-05 11:24 AUTHORS.txt
-rw-r--r-- 1 user user     3780 2007-11-21 12:30 changelog.txt
drwxr-xr-x 3 user user     4096 2008-06-25 14:48 configfiles400
drwxr-xr-x 3 user user     4096 2008-07-29 06:08 configfiles450
drwxr-xr-x 8 user user     4096 2008-09-09 08:41 demo
drwxr-xr-x 6 user user     4096 2008-06-06 11:12 doc
-rw-r--r-- 1 user user     9937 2008-06-06 11:10 Doxyfile
drwxr-xr-x 3 user user     4096 2007-07-24 08:48 drivers
drwxr-xr-x 8 user user     4096 2008-09-05 11:32 include
-rw-r--r-- 1 user user      426 2007-05-22 12:11 INSTALL.txt
drwxr-xr-x 3 user user     4096 2007-09-27 10:38 KNI.net
drwxr-xr-x 5 user user     4096 2007-05-22 12:11 lib
-rw-r--r-- 1 user user    15149 2007-05-22 12:11 LICENSE.txt
-rw-r--r-- 1 user user     1420 2008-09-05 11:37 Makefile
drwxr-xr-x 4 user user     4096 2008-09-05 11:28 py
-rw-r--r-- 1 user user     1553 2008-06-06 10:54 readme.txt
drwxr-xr-x 8 user user     4096 2008-09-04 14:51 src
```

Die Ordner des Wurzelverzeichnisses haben folgende Inhalte:

Name	Beschreibung
configfiles400	Konfigurations Dateien für den Katana400
configfiles450	Konfigurations Dateien für den Katana450
demo	KNI Demoprogramme und Beispiele
doc	KNI Dokumentation
drivers	Treiber für die USB Verbindung zum Katana
include	Die Include Dateien des KNI
KNI.net	ein KNI wrapper für .NET
lib	Die statischen und dynamischen Bibliotheken für Linux und Windows
py	Python bindings für das KNI
src	Die KNI Sourcen in C++

Die Dateien des Wurzelverzeichnisses haben folgende Funktionen:

Name	Beschreibung
AUTHORS.txt	Autoren der KNI Software
changelog.txt	Änderungen bei den Versionen
Doxyfile	Deskriptor für die Generierung der Dokumentation
INSTALL.txt	Installations-Anleitung
KatanaNativeInterface.*	VisualStudio Dateien (Windows)
LICENSE.txt	Lizenz GNU GPL Version 2
Makefile	Deskriptor, um mit make unter Linux zu kompilieren
readme.txt	Kurzinfo
WindowsInstaller.iss	Deskriptor, um den Windows Installer zu generieren

2.1.4 KNI Software Architektur

Die Abbildung 2.1 zeigt die grundlegende Software Architektur des KNI und der verschiedenen Möglichkeiten diese in eigene Applikationen und Frameworks einzubinden.

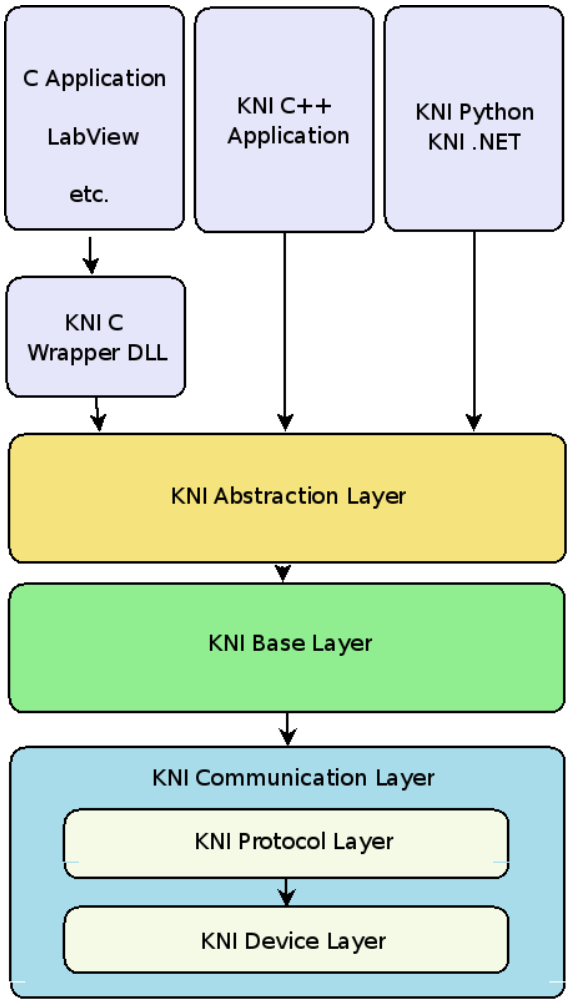

Abbildung 2.1: KNI Software Architecture

2.2 Build Abhängigkeiten und benötigte Tools

2.2.1 Linux und Mac

Um KNI zu kompilieren, benötigt man den GNU C und C++ Compiler mit den Standardbibliotheken und das GNU Tool 'make'.

2.2.2 Windows

Um die KNI-Bibliotheken und C++ Programme, die KNI benutzen, zu kompilieren, wird der MS Visual C++ Compiler verwendet. Informationen zu Microsoft VisualStudio findet man unter

```
http://www.microsoft.com/
```

2.3 Die Katana Konfigurations Dateien

Die Konfigurations-Dateien befinden sich in den Ordnern 'configfiles400' für den Katana400 bzw. 'configfiles450' für den Katana450. Für jeden Robotertyp gibt es eine eigene Datei, die entsprechend benannt ist.

Die Dateien enthalten gute Konfigurations-Werte und sollten nur verändert werden, wenn man genau weiss, was man tut. Bei Fehlmanipulationen in der Konfiguration kann der Roboter im Betrieb Schaden nehmen!

3 Programmierung in C++

Die folgenden Code-Ausschnitte zeigen die Verwendung von KNI: Verbinden, initializieren und bewegen.

3.1 Verbindung

Listing 3.1 zeigt einen Code-Ausschnitt, um die Verbindung zum Katana aufzubauen. Zuerst wird der Netzwerk-Socket geöffnet, das Protokoll instanziert und mit dem Socket initialisiert.

Listing 3.1: Verbindung

```
int port = 5566;
char* ip = "192.168.168.232";
std::auto_ptr<CCdlSocket> device;
std::auto_ptr<CCplSerialCRC> protocol;
try {
        // set device
        device.reset(new CCdlSocket(ip, port));
        // set protocol
        protocol.reset(new CCplSerialCRC());
        // initialize protocol
        protocol->init(device.get()); //fails if no response from Katana
} catch(Exception &e) {
        // handle exception
}
```

3.2 Initialisierung

Listing 3.2 zeigt einen Code-Ausschnitt, um den Katana zu initialisieren. Nach dem Instanzieren der Klasse CLMBase, die den Katana mit Linear Movement implementiert, wird diese mit der Konfigurations-Datei und der Protokoll-Instanz initialisiert.

Listing 3.2: Initialisierung

```
char* configfile = "configfiles450/katana6M90T.cfg";
std::auto_ptr<CLMBase> katana;
try {
        // create and initialize katana object
        katana.reset(new CLMBase());
        katana->create(configfile, protocol.get());
} catch(Exception &e) {
        // handle exception
}
```

3.3 Bewegungen

Listing 3.3 zeigt Beispiele für Punkt-zu-Punkt- und Linear-Bewegungen. Als erstes muss der Katana immer kalibriert werden. Danach wird die aktuelle Pose, d.h. die Position und Orientierung im Raum, ausgelesen und

eine zweite Pose daraus konstruiert. Die erste Bewegung mit 'moveRobotTo()' fährt ohne spezielle Beachtung der Bahn von der aktuellen zur konstruierten Pose. Die zweite Bewegung mit 'moveRobotLinearTo()' fährt auf einer linearen Bahn im Raum zurück zur ursprünglichen Pose.

Listing 3.3: Bewegungen

```
// calibrate katana!
katana->calibrate();
// get actual position
double x1, y1, z1, phi1, theta1, psi1;
katana->getCoordinates(x1, y1, z1, phi1, theta1, psi1);
// create second position
double x2 = x1 + 100.0;
double y2 = y1 - 50.0;
double z2 = z1 + 80.0;
double phi2 = phi1;
double theta2 = theta1;
double psi2 = psi1;
// move to position 2
katana->moveRobotTo(x2, y2, z2, phi2, theta2, psi2);
// move linear back to position 1
katana->moveRobotLinearTo(x1, y1, z1, phi1, theta1, psi1);
```

3.4 Demo Programme

Die KNI Sourcen beinhalten auch einige Demo Programme, welche die Nutzung der KNI Bibliothek veranschaulichen. Diese befinden sich im Verzeichnis demo und werden beim Aufruf von 'make' im Wurzelverzeichnis automatisch mit kompiliert. In den folgenden Abschnitten werden die verschiedenen Beispielprogramme kurz beschrieben.

3.4.1 control

Dieses Beispiel ist der von Neuronics empfohlene Einstieg in die Programmierung mit KNI. Nach dem Start mit dem folgenden Befehl kann direkt mit dem Katana über verschiedene Eingaben kommuniziert, Daten abgefragt und Programme geteacht werden:

Listing 3.4: Beispielprogramm control

```
./control ../../configfiles450/(katanatype) IP-address
success: katana initialized
-----------------------------------------
?: Display this help
c: Calibrate the Katana
e: Read the current encoder values
o: Switch motors off/on (Default: On)
r: Switch angle format: Radian/Degree (Default: Rad)
x: Read the current position
v: Set the velocity limits for all motors seperately
V: Set the velocity limits for all motors (or for the TCP if in linear movement mode)
a: Set the acceleration limits for all motors seperately
A: Set the acceleration limits for all motors
w: Read the velocity limits of all motors
W: Read the acceleration limits of all motors
q: Read the Sensors
y: Set a new position using IK
l: Switch on/off linear movements
<: Add a point to the point list
```

```
>: Move to a specific point
 : (space) Move to the next point in the point list
=: write pointlist to file
f: read pointlist from file
g: Open Gripper
h: Close Gripper
n: Set the speed collision limit for all motors seperately
N: Set the speed collision limit for all motors
s: Set the position collision limit for all motors seperately
S: Set the position collision limit for all motors
t: Switch collision limit on
T: Switch collision limit off
u: Unblock motors after crash
d: Move motor to degrees
z: Set TCP offset

1: Move motor1 left
2: Move motor1 right
3: Move motor2 left
4: Move motor2 right
5: Move motor3 left
6: Move motor3 right
7: Move motor4 left
8: Move motor4 right
9: Move motor5 left
0: Move motor5 right
/: Move motor6 left
*: Move motor6 right
.: Toggle Step mode
+: Increase step size
-: Decrease step size

$: Start/Stop Program
p: Start/Stop movement through points list
```

3.4.2 commands

Dieses Beispiel zeigt die Kommunikation mit dem Katana basierend auf den einzelnen Kommandos des Katana Firmware Protokolls. Nach dem Start mit

Listing 3.5: Beispielprogramm commands

```
./commands ../../configfiles450/(katanatype) IP-address
```

können mittels den aufgelisteten Eingabeoptionen die einzelnen Kommandos konfiguriert und gesendet werden.

3.4.3 csharp

Dies ist ein kleines Beispiel für die Einbindung des KNI.NET wrappers mit csharp für Microsoft Visual Studio.

3.4.4 kni_wrapper

Dies ist ein Beispiel für die Nutzung der kni_wrapper Bibliothek, welche Aufrufe aus nicht objektorientierten Programmiersprachen (C) und Entwicklungsumgebungen (LabView, Matlab) erlaubt.

4 Kinematik

In der Katana-Konfigurations-Datei können unter [KATANA] [GENERAL] kinematics zwei verschiedene Kinematiken ausgewählt werden: Eine integrierte analytische Kinematik und eine externe numerische Kinematik, die auch in der Steuerungssoftware Katana4D verwendet wird.

4.1 Integrierte Kinematik Bibliothek

Setzt man in der Konfiguration 'kinematics = Analytical', wird die integrierte analytische Kinematik verwendet. Diese Implementierung berechnet die inverse Kinematik sehr schnell, hat aber konzeptionell Probleme in der numerischen Berechnung und findet deshalb unter Umständen falsche Lösungen.

4.2 RobAnaGuess Kinematik Bibliothek

Setzt man in der Konfiguration 'kinematics = RobAnaGuess', wird die externe numerische Kinematik verwendet. Diese braucht für die Berechnung etwas mehr Zeit, ist dafür aber viel robuster. Der Schwachpunkt der numerischen Methode, nämlich dass zur Berechnung der inversen Kinematik ein 'guter' Startwert in der Nähe der Lösung angegeben werden muss, wird mit Hilfe der instabileren analytischen Implementierung und einigen Tricks entschärft. Durch diese Kombination werden massiv bessere Resultate erzielt als dies mit der rein analytischen Implementierung der Fall ist.

5 Integration in andere Sprachen und Frameworks

5.1 .NET

Im Verzeichnis KNI.net befindet sich ein .NET wrapper, welcher die Einbindung des KNI unter .NET erlaubt. Dieser Wrapper kann mit der .NET Compileroption 'cli' kompiliert werden. Eine Beispiel Visual Studio Projektdatei zeigt dabei die nötigen Compiler Einstellungen.

5.2 C basierte Schnittstellen

Die KNI Bibliothek kni_wrapper beinhaltet ein nicht objektorientiertes Interface zu KNI, welches Aufrufe von 'C' basierten Umgebungen erlaubt. Damit lässt sich das KNI in Mess- und Steuerungsumgebungen wie zum Beispiel LabView oder Matlab integrieren. Für diesen Wrapper gibt es zur Illustration ein Beispielprogramm unter `demo/kni_wrapper`.

5.3 Python

Das KNI kann über den kni_wrapper und einen automatisch mittels SWIG erstellten Python Wrapper in der Skriptsprache Python verwendet werden. Die entsprechenden Sourcen und Dateien befinden sich im Verzeichnis `py`. Mit dem Aufruf von 'make' in diesem Verzeichnis werden die Dateien KNI.py und _KNI.so generiert. Diese beinhalten die gesamten KNI Bibliotheken und können so direkt in einer Python Shell oder in einem Python Skript importiert und genutzt werden. Dadurch kann das KNI auch skriptbasiert eingebunden werden.

Das Interface ist im Allgemeinen gleich wie beim kni_wrapper. Nur wenige Funktionen unterscheiden sich in der Signatur wegen konzeptioneller Unterschiede zwischen C und Python. Genauere Informationen sind in der `readme.txt` Datei im `py` Verzeichnis zu finden.

Listing 5.1 zeigt die Verwendung des KNI aus einer Python Shell.

Listing 5.1: Benutzung von KNI aus Python

```
>>> import KNI
>>> KNI.initKatana("../configfiles450/katana6M90T.cfg", "192.168.1.1")
1
>>> KNI.calibrate(0)
Katana4xx calibration started
...done with calibration.
1
>>> KNI.getEncoder(1)
30500
>>> KNI.moveMot(1, 20000, 50, 2)
1
>>> from KNI import TMovement
>>> home = TMovement()
>>> KNI.getPosition(home.pos)
1
```

```
>>> home.transition = KNI.PTP
>>> home.velocity = 50
>>> home.acceleration = 2
>>> KNI.allMotorsOff()
1
>>> """move robot by hand to an other position"""
'move robot by hand to an other position'
>>> KNI.allMotorsOn()
1
>>> KNI.executeMovement(home)
1
>>> KNI.allMotorsOff()
1
>>>
```

5.3.1 LabView

Das KNI kann über den kni_wrapper in Labview integriert werden. Unter dem Verzeichnis demo/kni_labview gibt es das Beispielprogramm `kni_labview.vi` für die Einbindung des KNI in LabView.

5.3.2 Matlab

Das KNI kann über den kni_wrapper in Matlab integriert werden. Unter dem Verzeichnis demo/kni_matlab gibt es das Beispielprogramm `kni_matlab.m` für die Einbindung des KNI in Matlab.