# Entwicklung einer Programmierschnittstelle zum Roboterarm „Katana“

Ingo Schalk-Schupp

Studienarbeit am Lehrstuhl für kognitive Systeme

9. Februar 2010

# Überblick

Themen des Vortrags:

- Aufgabenstellung
- Grundlagen
- Ergebnisse
- Ausblick

# Überblick

Themen des Vortrags:

- Aufgabenstellung
- Grundlagen
- Ergebnisse
- Ausblick

# Überblick

Themen des Vortrags:

- Aufgabenstellung
- Grundlagen
- Ergebnisse
- Ausblick

# Überblick

Themen des Vortrags:

- Aufgabenstellung
- Grundlagen
- Ergebnisse
- Ausblick

# Überblick

Themen des Vortrags:

- Aufgabenstellung
- Grundlagen
- Ergebnisse
- Ausblick

# Aufgabenstellung

Die wichtigsten Punkte der Aufgabenstellung:

- Kommunikation zwischen PC und Katana
- Routine zur Justierung
- Entwurf einer Programmierschnittstelle
- Inverse Kinematik

# Aufgabenstellung

Die wichtigsten Punkte der Aufgabenstellung:

- Kommunikation zwischen PC und Katana
- Routine zur Justierung
- Entwurf einer Programmierschnittstelle
- Inverse Kinematik

# Aufgabenstellung

Die wichtigsten Punkte der Aufgabenstellung:

- Kommunikation zwischen PC und Katana
- Routine zur Justierung
- Entwurf einer Programmierschnittstelle
- Inverse Kinematik

# Aufgabenstellung

Die wichtigsten Punkte der Aufgabenstellung:

- Kommunikation zwischen PC und Katana
- Routine zur Justierung
- Entwurf einer Programmierschnittstelle
- Inverse Kinematik

# Aufgabenstellung

Die wichtigsten Punkte der Aufgabenstellung:

- Kommunikation zwischen PC und Katana
- Routine zur Justierung
- Entwurf einer Programmierschnittstelle
- Inverse Kinematik

# Grundlagen

- Kommunikation
  - physikalisch
  - logisch
- Kinematik

# Serielle Schnittstelle

RS-232-Protokoll:

- Symbolträger Spannungspegel
- Aufteilung in Datenworte
- Zwei Datenleitungen
- frei wählbare Symbolrate (hier: 57600 baud)
- wählbarer Overhead (hier: 1 Stopbit, keine Parität)

| 1 | $x_0$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Start | Daten | | | | | | | | Stop |

# Serielle Schnittstelle

RS-232-Protokoll:

- Symbolträger Spannungspegel
- Aufteilung in Datenworte
- Zwei Datenleitungen
- frei wählbare Symbolrate (hier: 57600 baud)
- wählbarer Overhead (hier: 1 Stopbit, keine Parität)

| 1 | $x_0$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Start | Daten | | | | | | | | Stop |

# Serielle Schnittstelle

RS-232-Protokoll:

- Symbolträger Spannungspegel
- Aufteilung in Datenworte
- Zwei Datenleitungen
- frei wählbare Symbolrate (hier: 57600 baud)
- wählbarer Overhead (hier: 1 Stopbit, keine Parität)

| 1 | $x_0$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Start | Daten | | | | | | | | Stop |

# Serielle Schnittstelle

RS-232-Protokoll:

- Symbolträger Spannungspegel
- Aufteilung in Datenworte
- Zwei Datenleitungen
- frei wählbare Symbolrate (hier: 57600 baud)
- wählbarer Overhead (hier: 1 Stopbit, keine Parität)

| 1 | $x_0$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Start | Daten | | | | | | | | Stop |

# Serielle Schnittstelle

RS-232-Protokoll:

- Symbolträger Spannungspegel
- Aufteilung in Datenworte
- Zwei Datenleitungen
- frei wählbare Symbolrate (hier: 57600 baud)
- wählbarer Overhead (hier: 1 Stopbit, keine Parität)

| 1 | $x_0$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Start | Daten | | | | | | | | Stop |

# Serielle Schnittstelle

RS-232-Protokoll:

- Symbolträger Spannungspegel
- Aufteilung in Datenworte
- Zwei Datenleitungen
- frei wählbare Symbolrate (hier: 57600 baud)
- wählbarer Overhead (hier: 1 Stopbit, keine Parität)

| 1 | $x_0$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | 0 |
|---|---|---|---|---|---|---|---|---|---|
| Start | Daten | | | | | | | | Stop |

# Katana-Protokoll

Serial-Zero-Protocol:

- setzt serielle Schnittstelle voraus
- teilweise dokumentiert
- Paket = Header + Befehl + Prüfsumme

| | Katana-Befehlspaket mit Prüfsumme | | | | | |
|---|---|---|---|---|---|---|
| Abschnitt | Header | | | Befehl | | Prüfsumme |
| Inhalt | 1 | Adresse | Länge | Opcode | Daten | CRC16 |
| Länge in Bytes | 1 | 1 | 1 | 1 | n(0..255) | 2 |
| *Beispiel* | *1* | *24* | *5* | *'C'* | *3, 24, 4, 0* | *76, 199* |

bewegt das Ellenbogengelenk zur Position 1024, entspricht

```
CKatana::setRawMotorPosition(3,1024)
```

# Katana-Protokoll

Serial-Zero-Protocol:

- setzt serielle Schnittstelle voraus
- teilweise dokumentiert
- Paket = Header + Befehl + Prüfsumme

| | Katana-Befehlspaket mit Prüfsumme | | | | | |
|---|---|---|---|---|---|---|
| Abschnitt | Header | | | Befehl | | Prüfsumme |
| Inhalt | 1 | Adresse | Länge | Opcode | Daten | CRC16 |
| Länge in Bytes | 1 | 1 | 1 | 1 | n(0..255) | 2 |
| *Beispiel* | *1* | *24* | *5* | *'C'* | *3, 24, 4, 0* | *76, 199* |

bewegt das Ellenbogengelenk zur Position 1024, entspricht

```
CKatana::setRawMotorPosition(3,1024)
```

# Katana-Protokoll

Serial-Zero-Protocol:

- setzt serielle Schnittstelle voraus
- teilweise dokumentiert
- Paket = Header + Befehl + Prüfsumme

| | Katana-Befehlspaket mit Prüfsumme | | | | | |
|---|---|---|---|---|---|---|
| Abschnitt | Header | | | Befehl | | Prüfsumme |
| Inhalt | 1 | Adresse | Länge | Opcode | Daten | CRC16 |
| Länge in Bytes | 1 | 1 | 1 | 1 | n(0..255) | 2 |
| *Beispiel* | *1* | *24* | *5* | *'C'* | *3, 24, 4, 0* | *76, 199* |

bewegt das Ellenbogengelenk zur Position 1024, entspricht

```
CKatana::setRawMotorPosition(3,1024)
```

# Katana-Protokoll

Serial-Zero-Protocol:

- setzt serielle Schnittstelle voraus
- teilweise dokumentiert
- Paket = Header + Befehl + Prüfsumme

| | Katana-Befehlspaket mit Prüfsumme | | | | | |
|---|---|---|---|---|---|---|
| Abschnitt | Header | | | Befehl | | Prüfsumme |
| Inhalt | 1 | Adresse | Länge | Opcode | Daten | CRC16 |
| Länge in Bytes | 1 | 1 | 1 | 1 | n(0..255) | 2 |
| *Beispiel* | *1* | *24* | *5* | *'C'* | *3, 24, 4, 0* | *76, 199* |

bewegt das Ellenbogengelenk zur Position 1024, entspricht

```
CKatana::setRawMotorPosition(3,1024)
```

# Frames

Frames sind:

- Koordinatensysteme,
- ortsfest zu einem Körper und
- können durch homogene Matrizen beschrieben werden.

# Frames

Frames sind:

- Koordinatensysteme,
- ortsfest zu einem Körper und
- können durch homogene Matrizen beschrieben werden.

# Frames

Frames sind:

- Koordinatensysteme,
- ortsfest zu einem Körper und
- können durch homogene Matrizen beschrieben werden.

# Frames

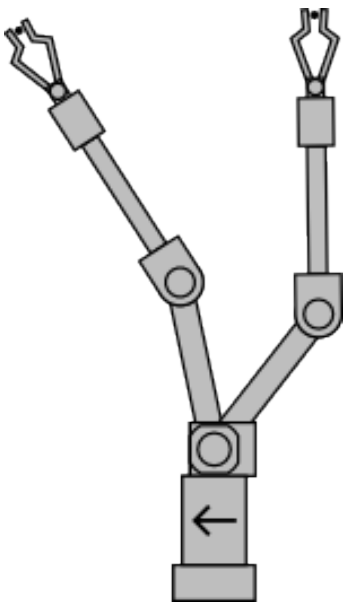
Frames sind:

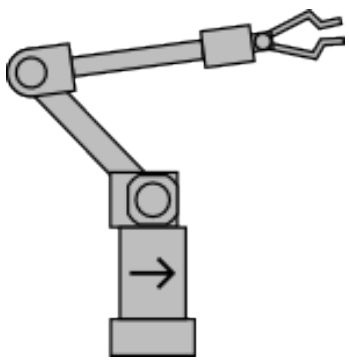
- Koordinatensysteme,
- ortsfest zu einem Körper und
- können durch homogene Matrizen beschrieben werden.

# Transformationen nach Denavit und Hartenberg

Denavit-Hartenberg-Transformationen (DH):

- Konvention zur effizienten Modellierung
- beschreiben einfache offene kinematische Ketten

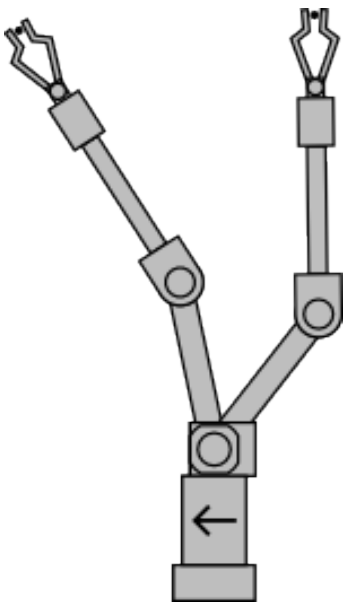



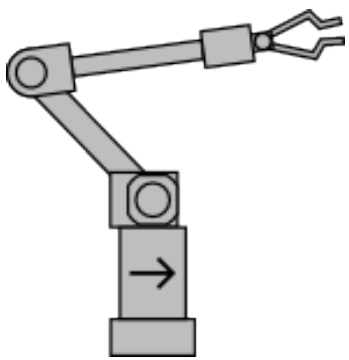


# Transformationen nach Denavit und Hartenberg

Denavit-Hartenberg-Transformationen (DH):

- Konvention zur effizienten Modellierung
- beschreiben einfache offene kinematische Ketten

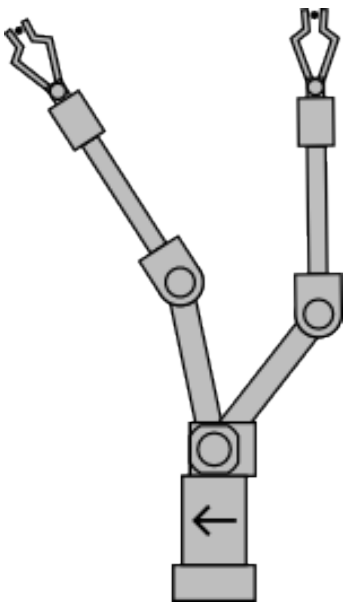



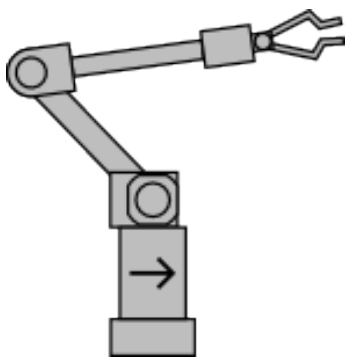


# Transformationen nach Denavit und Hartenberg

Denavit-Hartenberg-Transformationen (DH):

- Konvention zur effizienten Modellierung
- beschreiben einfache offene kinematische Ketten



# Vorwärtskinematik und inverse Kinematik

| | gegeben: | gesucht: | Eigenschaft: |
|---|---|---|---|
| Vorwärtskinematik: | Gelenkwinkel | Greiferframe | direkte Lösung |
| inverse Kinematik: | Greiferframe | Gelenkwinkel | mehrdeutig |

# Vorwärtskinematik und inverse Kinematik

| | gegeben: | gesucht: | Eigenschaft: |
|---|---|---|---|
| Vorwärtskinematik: | Gelenkwinkel | Greiferframe | direkte Lösung |
| inverse Kinematik: | Greiferframe | Gelenkwinkel | mehrdeutig |

# Vorwärtskinematik und inverse Kinematik

| | gegeben: | gesucht: | Eigenschaft: |
|---|---|---|---|
| Vorwärtskinematik: | Gelenkwinkel | Greiferframe | direkte Lösung |
| inverse Kinematik: | Greiferframe | Gelenkwinkel | mehrdeutig |

# Vorwärtskinematik und inverse Kinematik

| | gegeben: | gesucht: | Eigenschaft: |
|---|---|---|---|
| Vorwärtskinematik: | Gelenkwinkel | Greiferframe | direkte Lösung |
| inverse Kinematik: | Greiferframe | Gelenkwinkel | mehrdeutig |

# Vorwärtskinematik und inverse Kinematik

| | gegeben: | gesucht: | Eigenschaft: |
|---|---|---|---|
| Vorwärtskinematik: | Gelenkwinkel | Greiferframe | direkte Lösung |
| inverse Kinematik: | Greiferframe | Gelenkwinkel | mehrdeutig |

# Vorwärtskinematik und inverse Kinematik

| | gegeben: | gesucht: | Eigenschaft: |
|---|---|---|---|
| Vorwärtskinematik: | Gelenkwinkel | Greiferframe | direkte Lösung |
| inverse Kinematik: | Greiferframe | Gelenkwinkel | mehrdeutig |

# Vorwärtskinematik und inverse Kinematik

| | gegeben: | gesucht: | Eigenschaft: |
|---|---|---|---|
| Vorwärtskinematik: | Gelenkwinkel | Greiferframe | direkte Lösung |
| inverse Kinematik: | Greiferframe | Gelenkwinkel | mehrdeutig |

Die Klasse `CKatana`:

- verwaltet die Kommunikation,
- bildet Katana-Befehle ab
  `CKatana::setRawMotorPosition(3,1024)`,
- stellt Datentypen zur Verfügung
  `CKatana::TPosition` und
- justiert den Katana
  `CKatana::Adjust()`.

Die Klasse `CKatana`:

- verwaltet die Kommunikation,
- bildet Katana-Befehle ab
  `CKatana::setRawMotorPosition(3,1024)`,
- stellt Datentypen zur Verfügung
  `CKatana::TPosition` und
- justiert den Katana
  `CKatana::Adjust()`.

Die Klasse `CKatana`:

- verwaltet die Kommunikation,
- bildet Katana-Befehle ab
  `CKatana::setRawMotorPosition(3,1024)`,
- stellt Datentypen zur Verfügung
  `CKatana::TPosition` und
- justiert den Katana
  `CKatana::Adjust()`.

Die Klasse `CKatana`:

- verwaltet die Kommunikation,
- bildet Katana-Befehle ab
  `CKatana::setRawMotorPosition(3,1024)`,
- stellt Datentypen zur Verfügung
  `CKatana::TPosition` und
- justiert den Katana
  `CKatana::Adjust()`.

Die Klasse `CKatana`:

- verwaltet die Kommunikation,
- bildet Katana-Befehle ab `CKatana::setRawMotorPosition(3,1024)`,
- stellt Datentypen zur Verfügung `CKatana::TPosition` und
- justiert den Katana `CKatana::Adjust()`.

Die Klasse `CKatanaKinematic`:

- erweitert die Klasse `CKatana`,
- spiegelt die Physik des Katanas wider `CKatanaKinematik::CK_HeightShoulder` und
- implementiert die Kinematik `CKatanaKinematic::gotoXYZ(Matrix)`.

Die Klasse `CKatanaKinematic`:

- erweitert die Klasse `CKatana`,
- spiegelt die Physik des Katanas wider `CKatanaKinematik::CK_HeightShoulder` und
- implementiert die Kinematik `CKatanaKinematic::gotoXYZ(Matrix)`.

Die Klasse `CKatanaKinematic`:

- erweitert die Klasse `CKatana`,
- spiegelt die Physik des Katanas wider `CKatanaKinematik::CK_HeightShoulder` und
- implementiert die Kinematik `CKatanaKinematic::gotoXYZ(Matrix)`.

Die Klasse `CKatanaKinematic`:

- erweitert die Klasse `CKatana`,
- spiegelt die Physik des Katanas wider `CKatanaKinematik::CK_HeightShoulder` und
- implementiert die Kinematik `CKatanaKinematic::gotoXYZ(Matrix)`.

## Die Klasse `Matrix`:

- stellt 2D-Matrizen beliebiger Größe dar
  `Matrix::Matrix(int xSize=4, int ySize=4)` und
- implementiert viele Matrix-Operatoren
  `Matrix::flipUD()`
  `Matrix::transpose()`
  `Matrix::operator *=(const double);`
  `Matrix::operator *=(const Matrix&);`

Die Klasse `Matrix`:

- stellt 2D-Matrizen beliebiger Größe dar
  `Matrix::Matrix(int xSize=4, int ySize=4)` und
- implementiert viele Matrix-Operatoren
  `Matrix::flipUD()`
  `Matrix::transpose()`
  `Matrix::operator *=(const double);`
  `Matrix::operator *=(const Matrix&);`

Die Klasse `Matrix`:

- stellt 2D-Matrizen beliebiger Größe dar
  `Matrix::Matrix(int xSize=4, int ySize=4)` und
- implementiert viele Matrix-Operatoren
  `Matrix::flipUD()`
  `Matrix::transpose()`
  `Matrix::operator *=(const double);`
  `Matrix::operator *=(const Matrix&);`

Die Bibliothek `ISSMatrixManipulations`:

- erstellt Transformationsmatrizen zur
  - Translation,
  - Rotation und für
  - Denavit-Hartenberg-Transformationen sowie
  - deren jeweilige Inversen
- und bietet DH-Matrizen mit variablem Parameter an.

Die Bibliothek `ISSMatrixManipulations`:

- erstellt Transformationsmatrizen zur
  - Translation,
  - Rotation und für
  - Denavit-Hartenberg-Transformationen sowie
  - deren jeweilige Inversen
- und bietet DH-Matrizen mit variablem Parameter an.

Die Bibliothek `ISSMatrixManipulations`:

- erstellt Transformationsmatrizen zur
  - Translation,
  - Rotation und für
  - Denavit-Hartenberg-Transformationen sowie
  - deren jeweilige Inversen
- und bietet DH-Matrizen mit variablem Parameter an.

Die Bibliothek `ISSMatrixManipulations`:

- erstellt Transformationsmatrizen zur
  - Translation,
  - Rotation und für
  - Denavit-Hartenberg-Transformationen sowie
  - deren jeweilige Inversen
- und bietet DH-Matrizen mit variablem Parameter an.

Die Bibliothek `ISSMatrixManipulations`:

- erstellt Transformationsmatrizen zur
  - Translation,
  - Rotation und für
  - Denavit-Hartenberg-Transformationen sowie
  - deren jeweilige Inversen
- und bietet DH-Matrizen mit variablem Parameter an.

Die Bibliothek `ISSMatrixManipulations`:

- erstellt Transformationsmatrizen zur
  - Translation,
  - Rotation und für
  - Denavit-Hartenberg-Transformationen sowie
  - deren jeweilige Inversen
- und bietet DH-Matrizen mit variablem Parameter an.

Die Bibliothek `ISSMatrixManipulations`:

- erstellt Transformationsmatrizen zur
  - Translation,
  - Rotation und für
  - Denavit-Hartenberg-Transformationen sowie
  - deren jeweilige Inversen
- und bietet DH-Matrizen mit variablem Parameter an.

Weitere Möglichkeiten und Anwendungen:

- Genauigkeitsbetrachtungen
- Teaching
- Kollisionsvermeidung
- Kollisionserkennung
- Sensoren

Weitere Möglichkeiten und Anwendungen:

- Genauigkeitsbetrachtungen
- Teaching
- Kollisionsvermeidung
- Kollisionserkennung
- Sensoren

Weitere Möglichkeiten und Anwendungen:

- Genauigkeitsbetrachtungen
- Teaching
- Kollisionsvermeidung
- Kollisionserkennung
- Sensoren

Weitere Möglichkeiten und Anwendungen:

- Genauigkeitsbetrachtungen
- Teaching
- Kollisionsvermeidung
- Kollisionserkennung
- Sensoren

Weitere Möglichkeiten und Anwendungen:

- Genauigkeitsbetrachtungen
- Teaching
- Kollisionsvermeidung
- Kollisionserkennung
- Sensoren

Weitere Möglichkeiten und Anwendungen:

- Genauigkeitsbetrachtungen
- Teaching
- Kollisionsvermeidung
- Kollisionserkennung
- Sensoren

# Vorwärtskinematik

Programm `ISSKinTest`.

# inverse Kinematik

Programm `ISSKatanaPanel`.