

Datenbankbasierte Laufmaschine

Manuel-Leonhard Rixen

28. Juli 2016

Inhaltsverzeichnis

1	Messsystem	1
1.1	Einheit	2
1.1.1	Verwendete Komponenten	2
1.1.2	Testversion	4
1.1.3	Beta version	5
1.1.4	Release version	6
1.1.5	Programmaufbau	6
1.2	Hauptrechner	7
1.2.1	Verwendete Komponenten	7
1.2.2	Release Version	8
1.2.3	Programmaufbau	8
1.3	Anzug	8
1.3.1	Verwendete Komponenten	8
1.3.2	Testversion	9
1.3.3	Beta version	9
1.3.4	Release version	9
1.4	Erfassungsprogramm	9
1.4.1	Beschreibung HMI - Main	10
1.4.2	Beschreibung HMI - Visualisierung	11
1.4.3	Beschreibung HMI - Datenbank	12
1.4.4	Beschreibung Programmcode	13
2	Robotersystem	14

Zusammenfassung

Mithilfe eines speziellen Aufnahmesystems erfolgt das Erfassen definierter Bewegungsabläufe eines menschlichen Organismus. Aus diesen Werten werden parametrierbare Schrittfolgen extrahiert und in einer Datenbank abgespeichert. Ein bipedales Robotersystem greift über das Internet auf diese Daten zu und bewegt sich entsprechend der Schrittfolgen.

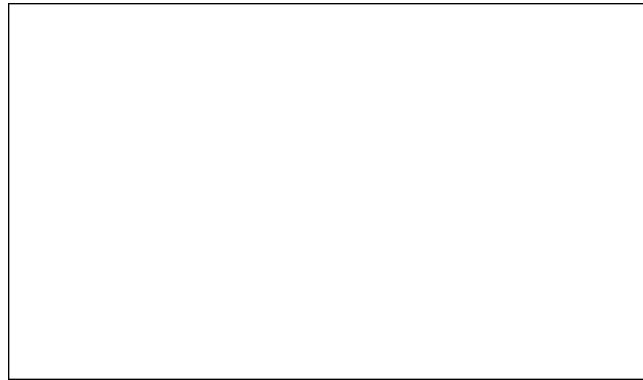


Abbildung 1: Bipedale Laufmaschine

Kapitel 1

Messsystem

Das Messsystem besteht aus mehreren Sensor-Einheiten, die an bestimmten Positionen eines menschlichen Körpers angebracht sind. Mit diesen Einheiten erfolgt das Erfassen von definierten Bewegungsabläufen, wie z.B. das Gehen geradeaus auf einer geraden Ebene, etc. Die Sensor-Einheiten sind über ein Bus-System (CAN-Bus) miteinander verbunden und kommunizieren mit einem Hauptrechner (Server), der die Sensordaten verarbeitet und über einen TCP-Socket versendet. Eine Client-Application verbindet sich mit dem Server und dient als HMI des Systems. Mit diesem Programm können die Messwerte grafisch dargestellt und in einer Datenbank abgespeichert werden.

Die Sensor-Einheiten und der Server sind in einen tragbaren Anzug eingearbeitet, während die Client-Application über einen herkömmlichen PC oder ein Smartphone bedient werden kann (Stoppen, Starten eines Messvorgangs, etc.).



Abbildung 1.1: Messsystem

1.1 Einheit

Die Mcp-Executor-Einheit definiert ein Gerät, welches Beschleunigungsdaten über SPI an ein Master-Gerät schickt, sobald es eingeschaltet wird.

Für diese Funktionalität werden die in Kapitel 1.1.1 beschriebenen Komponenten verwendet, welche wie in Abbildung 1.2 dargestellt entsprechend verknüpft sind.

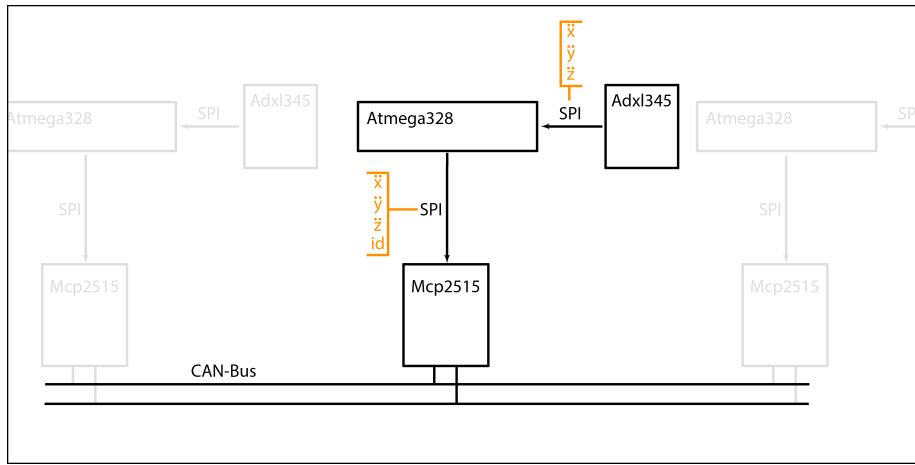


Abbildung 1.2: Zusammenspiel der Komponenten Mcp-Executor-Einheit

Der Microcontroller Atmega328 lässt von dem Beschleunigungssensor Adxl über die Schnittstelle SPI Sensordaten aus und gibt diese (ebenfalls über SPI) an den Mcp2515 weiter. Dieser schiebt die Daten auf den CAN-Bus.

1.1.1 Verwendete Komponenten

Für eine Einheit werden verschiedene Hardware-Komponenten verwendet, deren Wahl vor allem hinsichtlich des Kostenaufwandes erfolgte. Auch die Bauteilgröße ist ein entscheidender Faktor, da die Einheiten möglichst nicht störend sein sollen.

- **Mcp2515**

Der Mcp2515 beschreibt an sich den Chip, welcher auf der, in Abbildung 1.3 dargestellten, Platine verbaut ist. Dieser Name wird jedoch für den gesamten Chip verwendet.

Die Funktion des Mcp2515 besteht darin Daten über die Schnittstelle SPI zu empfangen und über einen CAN-Bus zu versendet. Die Steuerung des

Chips (z.B. Beschreiben eines Sende-Buffers) erfolgt dabei mit entsprechenden Kommandos über SPI. Das Konvertieren in L-, H-Pegel erfolgt komplett automatisch.

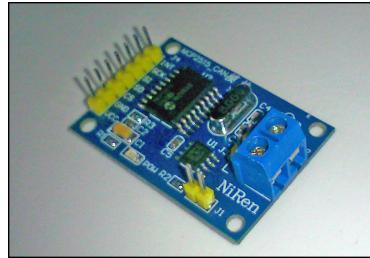


Abbildung 1.3: Mcp2515

- **Adxl345**

Bei dem Adxl345 handelt es sich um einen 3-Achsen Beschleunigungssensor (s. Abbildung 1.4), dessen Messwerte über SPI in verschiedenen Genauigkeiten ausgelesen werden können. Der Chip besitzt eine äußerst geringe Leistungsaufnahme und ist kompatibel in seiner Bauform.



Abbildung 1.4: Adxl345

- **Atmega328**

Der Microcontroller Atmega328 ist u.A. in dem Arduino-Board verbaut. Es ist ein kostenfünstiger Allrounder, welcher hauptsächlich wegen seiner großen Verbreitung gewählt wurde. Der Chip (s. Abbildung 1.5) besitzt eine SPI-Schnittstelle und diverse I/O's, weshalb die Projektanforderungen vollständig erfüllt werden.

Es existieren zwei verschiedene Versionen des MCU's, von denen die Atmega328-P Variante für eine geringe Leistungsaufnahme konzipiert ist.

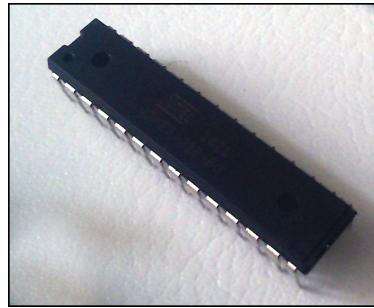


Abbildung 1.5: Atmega328

- **Oscillator und Kondensator**

Der Atmega328 benötigt einen externen Taktgeber, welcher mit einem 16Mhz Quarz und entsprechenden Kondensatoren gewählt wird (s. Abbildung 1.6).

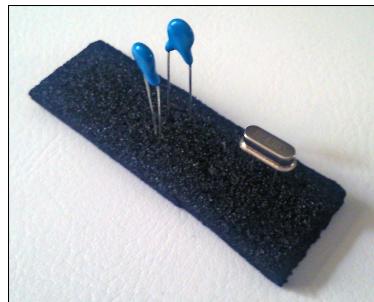


Abbildung 1.6: Oscillator

1.1.2 Testversion

Um die Funktionen der Chips zu verstehen und beherrschen zu können erfolgt der Aufbau einer Einheit auf einem Entwicklungsboard (Whiteboard). Dieser Aufbau beinhaltet alles, was für eine Einheit vorgesehen ist, d.h. einen Atmega328 (mit externem Taktgeber), einen Adxl345 und einen Mcp2515. Mit entsprechender Verdrahtung (s. Schaltbild ??) und der Entwicklung eines Programms, welches auf der MCU aktiv ist, lassen sich die Sensorwerte über einen CAN-Bus auslesen. Die Sensor-ID muss programmatisch auf der MCU definiert werden.

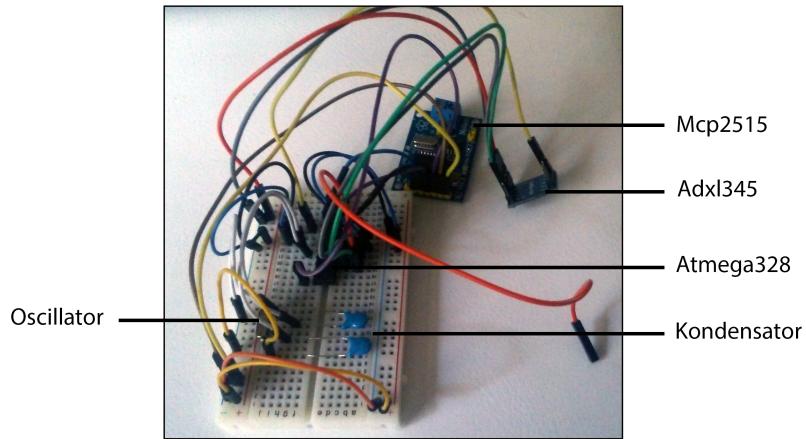


Abbildung 1.7: Testversion einer Mcp-Executor-Einheit

Mit zwei solcher Testaufbauten wurde das Aufnehmen von Sensorwerten validiert.

1.1.3 Beta version

Für die Betaversion der Mcp-Executor-Einheit werden alle Komponenten in ein Gehäuse integriert (s. Abbildung 1.8), welches gerade so groß ist, dass es beim Tragen am Körper möglichst nicht stört und einfach befestigt werden kann. Für das Integrieren werden zusätzliche Platinen angefertigt. Aufgrund des Feststellens verschiedener negativer Aspekte während der Entstehung der Einheiten existieren mehrere Beta-Versionen.

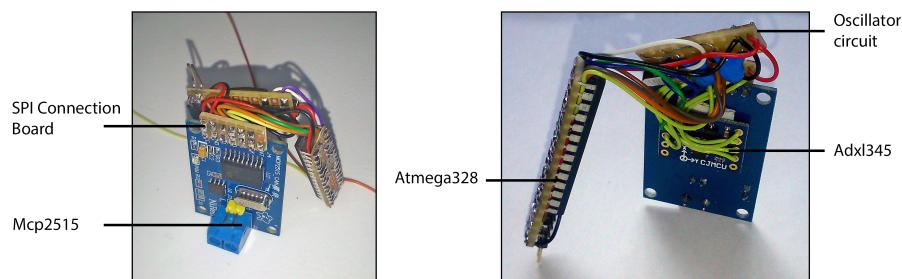


Abbildung 1.8: Betaversion einer Mcp-Executor-Einheit

Die Befestigung der Einheit am Körper erfolgt anhand eines Positioniergurtes. Dieser besteht in der ersten Beta-Version aus zwei und in der zweiten Version aus einem Klettverschluss, die an dem Abdeckblech der Einheit befestigt sind. Durch ein entsprechendes Gegenstück des Klettverschlusses auf der Rückseite des Abdeckblechs ist der notwendig Halt gegeben.

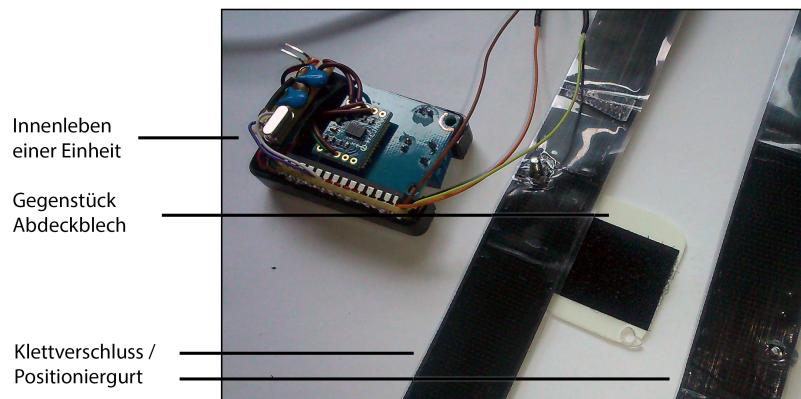


Abbildung 1.9: Betaversion einer Mcp-Executor-Einheit - Klettverschluss

An dem Anzug sind ebenfalls entsprechende Gegenstücke des Klettverschlusses angebracht, sodass die Einheiten während der Bewegung nicht verrutschen.

1.1.4 Release version

Bei der Realease-Version der Mcp-Executor-Einheit wird die Beta-Version zunächst bewertet und alle negativen Aspekte aufgezeigt und möglcihe Lösungen erarbeitet. Die zu verbessernden Punkte sind in Tabelle ?? aufgelistet:

1.1.5 Programmaufbau

Das Programm, welches auf der MCU einer Einheit aktiv ist besteht aus einer Setup- und einer Loop-Routine. Innerhalb des Setups erfolgt die Definition verschiedener Variablen und die Konfiguration der SPI-Schnisttstelle, sowie der Komponenten Adxl345 und Mcp2515 (s. Listing ??). In Zeile XXX

In der Loop-Routine (s. Listing ??) werden zwei Routinen aufgerufen, von denen die Routine `getAdxlData()` die Sensordaten ließt und zusammen mit einem Zeitstempel in ein Array speichert. Die Routine `mcp2515_load_txbuffer0()` schreibt die Daten in den Sende-Buffer 0.

Innerhalb der Routine `getAdxlData()` erfolgt in Zeile XXX... das

Innerhalb der Routine `mcp2515_load_txbuffer0()` erfolgt in Zeile XXX... das

1.2 Hauptrechner

Der Hauptrechner definiert ein Gerät, welches als Master die Sensordaten aller Mcp-Executor-Einheiten einfordert und verarbeitet.

1.2.1 Verwendete Komponenten

Die Bestandteile des Hauptrechners sind ein Raspberry-Pi 2 mit Wlan-Adapter und dem Betriebssystem Windows 10 IoT, sowie einer Energiequelle (Akku von Anker) mit XXX mAh.

- **Raspberry-Pi 2**

Der Raspberry Pi 2 (s. Abbildung ??) ist in seiner zweiten Version ein stabiles System, welches in Verbindung mit Windows IoT mit Csharp programmiert werden kann. Die Programme entsprechen den Anwendungen wie man sie von Smartphones kennt.

Die zweite Version beinhaltet (im Vergleich zur Version 3) keinen Wlan-Adapter, weshalb dieser zusätzlich erworben werden muss.

- **Wlan-Adapter**

Bei der Wahl eines Wlan-Adapters muss die Kompatibilität zu Windows-IoT berücksichtigt werden, da nicht für jedes Gerät entsprechende Treiber zur Verfügung stehen. Aus diesem Grunde liegt die Wahl bei dem preisgünstigen Wlan-Adapter XXXX (s. Abbildung 1.4).

- **Akku von Anker**

Für die Energieversorgung (s. Abbildung ??) wird ein Akku benötigt, welcher eine hohe Kapazität aufweist und genügend Strom liefert, damit das System ordnungsgemäß funktioniert. Bspw. benötigt der Raspberry Pi 2 2A, um einen fehlerfreien Betrieb zu gewährleisten (mit zusätzlich angeschlossener Hardware, wie z.B. der Wlan-Adapter).

Der Akku XXX von Anker ist für diese Anforderungen wie geschaffen, da dieser XXXX mAh aufweist und 2 USB-Ports mit jeweils 2A zur Verfügung stellt.

1.2.2 Release Version

1.2.3 Programmaufbau

1.3 Anzug

1.3.1 Verwendete Komponenten

1.3.2 Testversion

1.3.3 Beta version

1.3.4 Release version

1.4 Erfassungsprogramm

Das Lesen, bzw. Speichern der Sensordaten erfolgt mit einem Programm, welches sich als Client mit dem Server verbindet und über WLAN die Daten aller Sensoren empfängt. Die Sensorwerte werden nicht automatisch gespeichert, jedoch empfangen und können grafisch dargestellt werden (x-, y-, z-Werte für jeden Sensor in einer Übersicht). Ist das Speichern der Daten gewünscht, so muss eine Anzahl an Samples angegeben werden. Das Speichern der Daten erfolgt dann in eine Datenbank.

1.4.1 Beschreibung HMI - Main

Die HMI gliedert sich in mehrere Bereiche, in denen verschiedene Funktionalitäten bereitgestellt werden. Diese finden im folgenden Abschnitt Erläuterung.

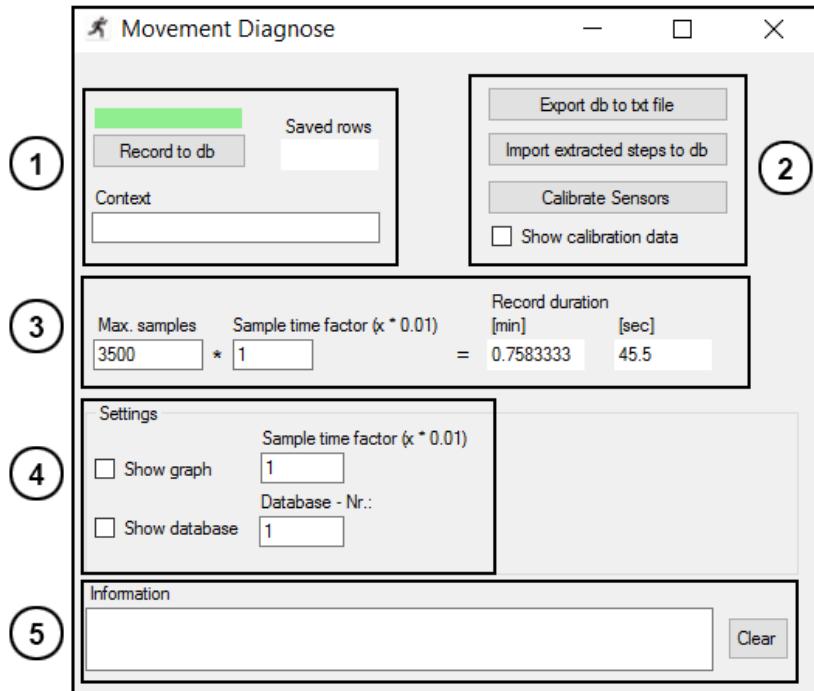


Abbildung 1.10: Client Graph HMI

1. Aufnahme starten

Der grüne Balken dient als Indikator, ob Messwerte empfangen werden (grün: Messung kann gestartet werden, rot: Es werden keine Daten empfangen).

Durch das Betätigen des Buttons *Record to db* wird eine Messung gestartet bis diese abgebrochen oder die max. Anzahl an Samples erreicht wird. Das Label *Saved rows* zeigt die aktuelle Anzahl aufgenommener Messwerte an.

Im Textfeld *Context* kann eine Bezeichnung der Messwerte eingegeben werden (die exportierten Textdateien tragen diesen Namen).

2. Import / Export und Kalibrierung

Mit dem Betätigen des Buttons *Export...* erfolgt das Exportieren der er-

fassten Sensordaten in eine CSV-Datei. Jeder Sensor generiert eine eigene Datei mit dem unter *COntext* angegebenen Namen.

Durch das Betätigen des Buttons *Import...* kann eine Textdatei (CSV-Format) eingelesen und in die Datenbank geschrieben werden.

Durch das Betätigen des Buttons *Calibrate Sensors* erfolgt das Kalibrieren aller Sensoren (s. Beschreibung in 1.4). Durch das Setzen des Häkchens *Show ...* erfolgt das Darstellen der kalibrierten Sensorwerte.

3. Aufnahmezeit

In das Textfeld *Max. samples* wird die aufzunehmende Anzahl an Messwerten eingetragen. Danach stoppt die Messung automatisch. Standardmäßig sind 3500 Samples eingetragen, da bei einer zu großen Anzahl eine *Out of memory Exception* geworfen wird.

4. Settings

Durch das Setzen des Häkchens bei *Show graph* wird ein grafische Visualisierung der Sensorwerte dargestellt.

Durch das Setzen des Häkchens bei *Show database* werden die Einträge der Datenbank dargestellt.

5. Information

In dem Textfeld erfolgt das Darstellen wichtiger Systemereignisse (z.B. Fehler), die durch das Betätigen des Buttons *Clear* gelöscht werden können.

1.4.2 Beschreibung HMI - Visualisierung

DURch das Betätigen des Buttons *Show graph* innerhalb der Hauptansicht des Programms, erfolgt das Darstellen der Visualisierung.

Das Visualisieren der Sensorwerte erfolgt durch vier Diagramme pro Sensor (s. Abbildung 1.11), in denen die Beschleunigungswerte in jeder Achse einzeln und einmal alle zusammen dargestellt werden. Die Auswahl des Sensors erfolgt durch einen Up-Down-Selector (s. Abbildung - Pos. 2) und das Erstellen eines Abbilds durch das Betätigen des Buttons *Snapshot*. Dieses wird auf dem Desktop abgelegt.

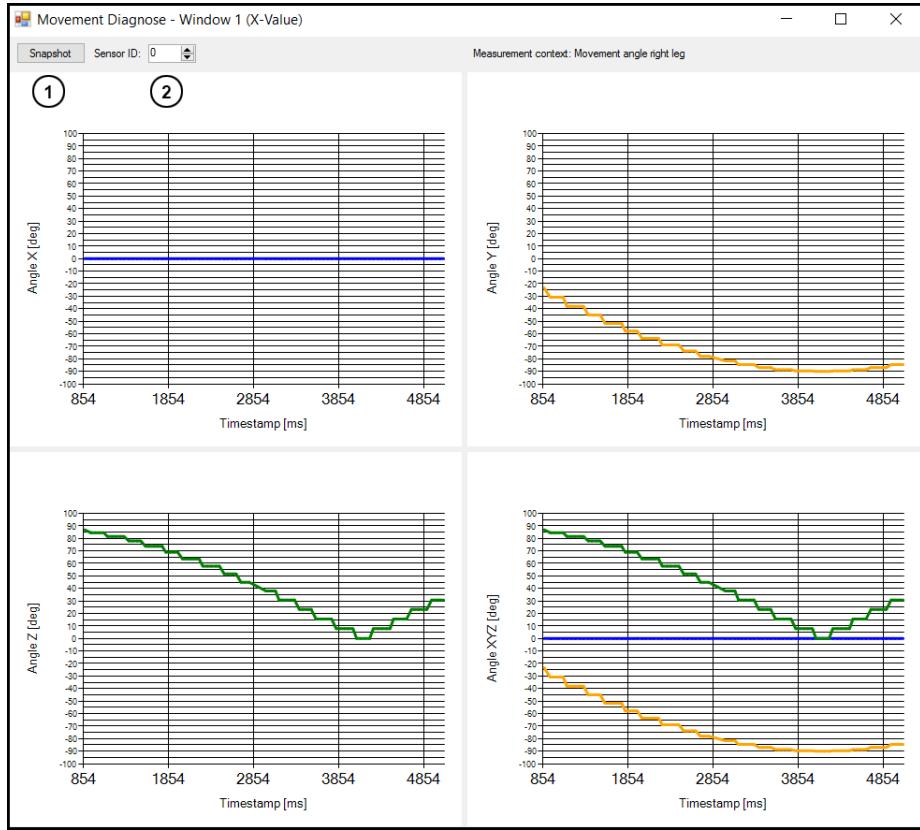


Abbildung 1.11: Diagramm der Sensorwerte

1.4.3 Beschreibung HMI - Datenbank

Während des Aufzeichnens von Sensorwerten erfolgt das Abspeichern in eine Datenbank. Diese beinhaltet die Daten für jeden Sensor einzeln mit den Werten Beschleunigungswerten in x, y und z, sowie einen Zeitstempel. Die Datenbank kann durch das Betätigen des Buttons *Show Database*, in der Hauptansicht des Programms, angezeigt werden (s. Abbildung 1.12). Durch das Betätigen des Up-Down-Selectors erfolgt die Auswahl des entsprechenden Sensors.

x	y	z	time
0	-86.933...	23.29371	4816
0	-86.933...	23.29371	4830
0	-86.933...	23.29371	4844
0	-86.933...	23.29371	4858
0	-86.933...	23.29371	4872
0	-86.933...	23.29371	4886
0	-86.933...	23.29371	4900
0	-84.572...	30.781...	4914
0	-84.572...	30.781...	4928
0	-84.572...	30.781...	4942
0	-84.572...	30.781...	4956
0	-84.572...	30.781...	4970
0	-84.572...	30.781...	4984
0	-84.572...	30.781...	4998
0	-84.572...	30.781...	5012
0	-84.572...	30.781...	5026
0	-84.572...	30.781...	5040
0	-84.572...	30.781...	5054
0	-84.572...	30.781...	5068
0	-84.572...	30.781...	5082
0	-84.572...	30.781...	5096
0	-84.572...	30.781...	5110
0	-84.572...	30.781...	5124
0	-81.567...	38.035...	5138
0	-81.567...	38.035...	5152
0	-81.567...	38.035...	5166
0	-81.567...	38.035...	5180

Abbildung 1.12: Datenbank mit Sensorwerten

1.4.4 Beschreibung Programmaufbau

In diesem Kapitel wird zu jeder Funktion (z.B. das Starten einer Messung) die dahinter liegenden Programm-Elemente aufgezeigt und erläutert. Triviale Funktionalität wie z.B. das Löschen eines Textfeldes werden dabei außer Acht gelassen.

Die Elemente werden dabei mit der in der HMI angegebenen Bezeichnung beschrieben.

1. Record to db

Pseudotext

2. Export db to txt file

Pseudotext

3. Import extracted steps to db
Pseudotext

4. Calibrate Sensors
Pseudotext

5. Show graph
Pseudotext

6. Show database
Pseudotext

Kapitel 2

Robotersystem