



Praktikumsbericht

vorgelegt von
Daniel Braun

Praktikum bei der DFKI GmbH

20.07.09 – 14.08.09

betreut durch Christoph Endres



Inhaltsverzeichnis

1	Aufgabenstellung.....	1
2	Technologie	2
2.1	Hardware	2
2.2	Software	5
2.2.1	LifeOS.....	5
2.2.2	Pawn	5
2.2.3	MySkit.....	6
2.2.4	DinoMITE	7
3	Pleopatra API.....	8
3.1	Schnittstelle	8
3.1.1	Herstellen einer Verbindung unter Windows	8
3.2	Funktionen.....	8
3.2.1	Audio	8
3.2.2	Streaming Protokoll.....	9
3.2.3	Foto.....	9
3.2.4	Pleo spezifische Befehle	10
3.3	Aufbau der API.....	10
4	Pleopatra Tools.....	11
4.1	Die Oberfläche.....	12
4.1.1	Connection	12
4.1.2	Pleo Camera	13
4.1.3	Motions	13
4.1.4	Personality	13
4.1.5	Sound.....	14
4.1.6	Status.....	14
4.1.7	Identity	15
4.1.8	Audio Recorder.....	15
4.1.9	Pleo Twitter	16
5	Pleopatra twittert.....	17
5.1	Twitter API.....	17
5.2	Funktionsweise.....	17
5.3	Twitter Profil.....	17
6	Zusammenfassung & Ausblick.....	19
6.1	Zusammenfassung.....	19
6.2	Ausblick.....	19
6.2.1	Twitter	19
6.2.2	Hardware	19

6.2.3	API / Pleopatra Tools	20
7	Anhang.....	21
7.1	Schnittstellenprotokoll	21
7.1.1	Befehlsübersicht:.....	21
7.1.2	Übersicht Motoren	27
7.1.3	Übersicht Sensoren	27
7.1.4	Übersicht Log-Events	28
7.2	Pleopatra API	29
7.2.1	Befehlsübersicht:.....	29

1 Aufgabenstellung

Im ersten Schritt ging es darum, sich mit der Hardware und der Software von Pleo vertraut zu machen und bereits vorhandene Programme zu suchen und zu analysieren. Dies stellte sich als relativ kompliziert dar, da es aufgrund der Insolvenz des Herstellers zurzeit kaum Informationen oder Programme gibt und bisher noch keinerlei Dokumentationen veröffentlicht wurden.

In einem zweiten Schritt war das Ziel dann eine Möglichkeit zu finden auf die Hardware zuzugreifen und den Roboter zu programmieren. Aus oben genannten Problemen fiel auch das schwer, obwohl Pleo eigentlich als offene Plattform konzipiert wurde.

Nachdem eine entsprechende Möglichkeit über die serielle Schnittstelle gefunden wurde, sollte zuerst die Kommunikation ermöglicht werden um in einem weiteren Schritt dann eine API für diese Kommunikation zu entwerfen und zu implementieren.

Im Anschluss haben wir zwei auf der API basierende Applikationen entworfen, die die Möglichkeiten, die die API bietet, verdeutlichen.

Am Ende stand die Dokumentation und Präsentation der Ergebnisse.

2 Technologie

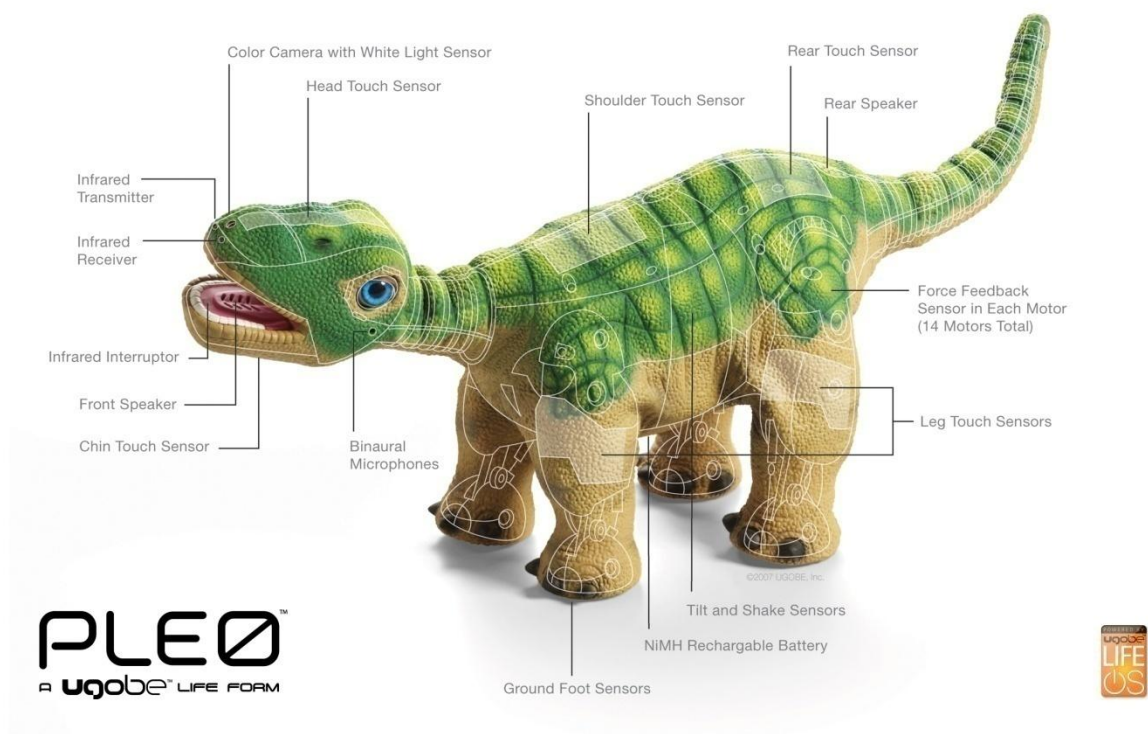
2.1 Hardware

Die Grundlage von Pleo bildet ein Atmel ARM7 32-Bit-Mikroprozessor (Hauptprozessor), sowie ein NXP ARM7 32-Bit-Mikroprozessor (Kamera, Audio) und vier Toshiba TMP86FH47AUG 8-Bit-Mikroprozessoren (Steuerung der Motoren).

Zur Bewegung seines Körpers stehen insgesamt 14 Motoren zur Verfügung, davon jeweils zwei in jedem Bein, einer im Kopf, jeweils zwei in Nacken und Schwanz (für vertikale und horizontale Bewegungen) und einer im Rücken. In jedem Motor befindet sich außerdem ein Feedback-Sensor, der die aktuelle Motor-Position überwacht.

Zusätzlich zu diesen Sensoren verfügt Pleo über:

- eine im Kopf eingebaute Kamera zur Helligkeitserkennung und zur Erkennung von Gegenständen
- zwei, ebenfalls im Kopf eingebaute, Mikrofone, zur Erkennung und Ortung von Lärm- bzw. Geräuschquellen
- acht Berührungssensoren (jeweils einer in jedem Bein, am Kopf, am Kinn, an den Schultern und am Rücken)
- jeweils einen Taster unter jedem Fuß
- einen Neigungs-Sensor
- einen Infrarotschranke im Mund
- Infrarot-Sender und Empfänger am Kopf



Quelle: Ugobe

Außerdem verfügt Pleo über zwei Lautsprecher; einer in seinem Mund und einer am Rücken unter seiner Haut und er verfügt über einen eingebauten Flash-Speicher, auf dem seine Software gespeichert ist.

Zum Übertragen von Persönlichkeitsprofilen, Sounds und Updates ist an der Unterseite ein SD-Karten-Slot integriert, der jedoch nicht alle SD-Karten problemlos unterstützt. Offiziell wird zwar nur der SDHC Standard nicht unterstützt, es sind aber ebenfalls Probleme mit Kingston SD-Karten bekannt. An der Unterseite befindet sich auch ein mini USB-Port, der auch zum Updaten der Software, oder zur Steuerung verwendet werden kann, der an einen USB zu Seriell Wandler angeschlossen ist.

Intern verfügt Pleo über zwei weitere serielle Schnittstellen, die allerdings nicht nach Außen geführt sind. Eine ist nur durch öffnen des gesamten Körper erreichbar, die andere ist auch von Außen, durch öffnen einer Abdeckung, relativ leicht erreichbar.

Den äußeren Anschluss erreicht man durch entfernen folgender Abdeckung:



Quelle: http://www.grip-online.com/de/pleo_hack/xbee

Nach dem Entfernen der Haut findet man den inneren Anschluss auf der Platine direkt unterhalb des Batteriefachs. Die Belegung ist wie folgt:



Pins of Pleos Serial Port:

- 1 - Ground (Black)
- 2 - Ground (Black)
- 3 - Pleo Xmit Data (Brown), ie hook to DataIn of transceiver
- 4 - Pleo Receive Data (Yellow), ie hook to DataOut of transceiver
- 5 - VCC (Orange), 3.3V
- 6 - ? Unknown (Red)
- 7 - ? Unknown (Red)

Pins of the XBee-Module:

- 1 - Power Supply (2.8 - 3.4 V)
- 2 - Data Out
- 3 - Data In
- ...
- 10 - Ground



Quelle: http://www.grip-online.com/de/pleo_hack/xbee

Diese beiden Ports können zum Beispiel dazu genutzt werden, ein Bluetooth- oder Funkmodul an einen Pleo anzuschließen und somit auch die drahtlose Kommunikation mit einem Computer zu ermöglichen.

Ein Nickel-Metallhydrid-Akku, mit einer Spannung von 7.2 V und einer Ladung von 2200 mAh versorgt den gesamten Roboter mit Strom. Der Akku ist auch einzeln als Zubehör erhältlich.



Quelle: Ugobe

Mit einer Akku-Ladung kann Pleo ca. 2 Stunden betrieben werden, die Aufladung des Akkus beansprucht allerdings bis zu 4 Stunden.

Im Inneren des Akku-Packs befindet sich ein Temperatur-Sensor, der während des Betriebs und während des Ladens ständig die Temperatur überwacht und den Ladevorgang gegebenenfalls unterbricht, beziehungsweise den Roboter abschaltet, sollte sich der Akku während des Betriebs überhitzen. Die Abschalttemperatur für den Betrieb kann auf Wunsch manuell geändert werden.

Eine der größten Schwachstellen von Pleo ist seine Haut; diese ist aus einem sehr elastischen Kunststoff gefertigt, der sehr empfindlich gegenüber Reibung ist und dadurch schnell seine Farbe verliert.

2.2 Software

2.2.1 LifeOS

Pleos Verhalten wird durch das sogenannte **LifeOS** gesteuert, das auf dem internen Flash-Speicher gespeichert ist. Pleos Verhalten und Entwicklung lässt sich in drei Phasen gliedern:

Geburt: In dieser ca. 10 bis 15 Minuten andauernden Phase ist Pleo besonders träge und liegt nur, da er noch nicht laufen kann. Diese Phase findet nach dem ersten Anschalten statt.

Schlüpfen: Diese Phase dauert 30 bis 45 Minuten. Während dieser Zeit macht Pleo seine ersten Schritte und beginnt seine Umgebung zu erkunden, ist aber immer noch vergleichsweise träge und schläft viel.

Erwachsenenalter: Pleo ist voll entwickelt, bewegt sich, interagiert und erkundet seine Umgebung.

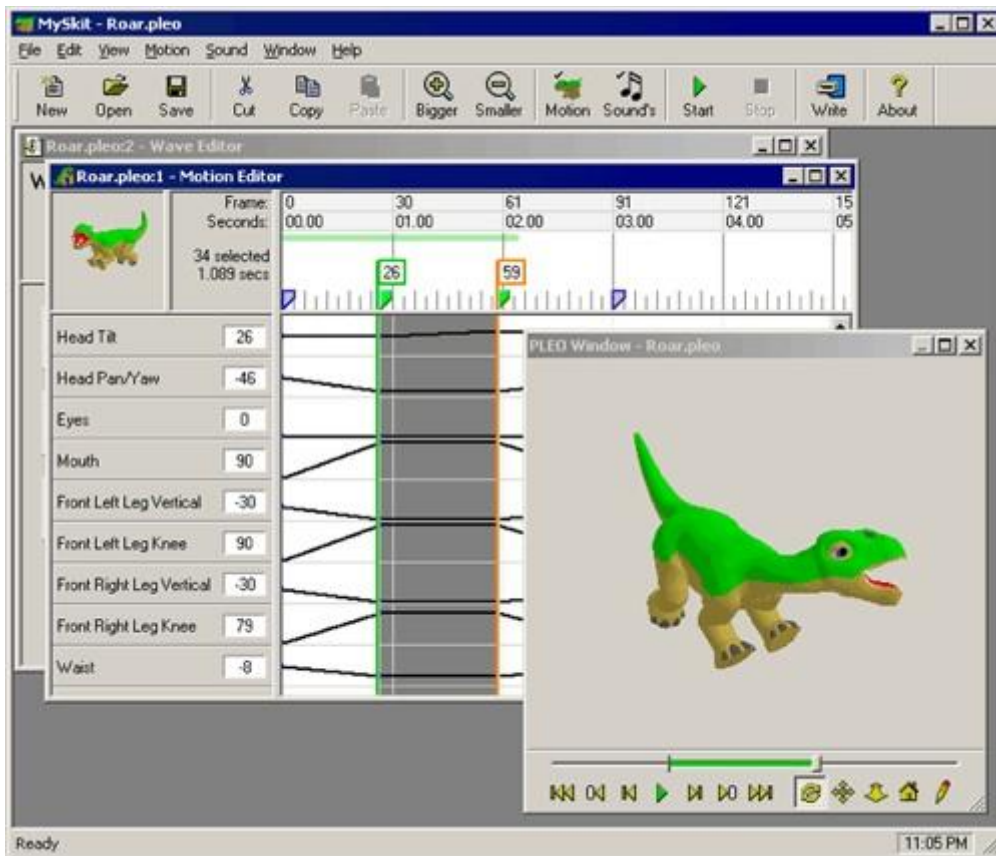
Laut Angaben des Herstellers beeinflusst die Art und Intensität der Interaktion, die Pleo während der ersten beiden Phasen erfährt, sein späteres Verhalten. Im Erwachsenenalter ist eine Beeinflussung des Verhaltens nicht mehr möglich. In unseren eigenen Tests und in Nutzerberichten fiel die Beeinflussung aber eher gering aus.

Ein reset ist nicht möglich, das heißt, die ersten beiden Phasen können nur einmal durchlaufen werden und eventuelle „Fehler“ die bei der „Erziehung“ gemacht wurden können nicht rückgängig gemacht werden.

2.2.2 Pawn

Pawn ist eine unter der Apache License 2.0 veröffentlichte, C-ähnliche Skriptsprache der Firma *CompuPhase*. Auf Pleo läuft eine VM, die es zukünftig ermöglichen soll eigene Pawn-Skripts zu schreiben und diese auszuführen, bisher fehlen dazu aber die passenden Bibliotheken, eine Dokumentation und ein benötigter Compiler. All das sollte eigentlich schon Ende 2008 im **Pleo Development Kit** (PDK) erscheinen, das sich aber Aufgrund wirtschaftlicher Probleme des Herstellers immer wieder verzögert hat. Aktuell ist kein Release-Termin bekannt.

2.2.3 MySkit



Quelle: <http://www.dogsbodynet.com/myskit/index.html>

MySkit¹ ist ein Programm zum Erstellen sogenannter Skits für Pleo. In Skits werden sequentielle Bewegungsabläufe gespeichert, die dann später auf dem Roboter ausgeführt werden können. Ursprünglich hieß das Programm **Skitter** und wurde für AIBO, den Roboterhund von Sony, entwickelt.

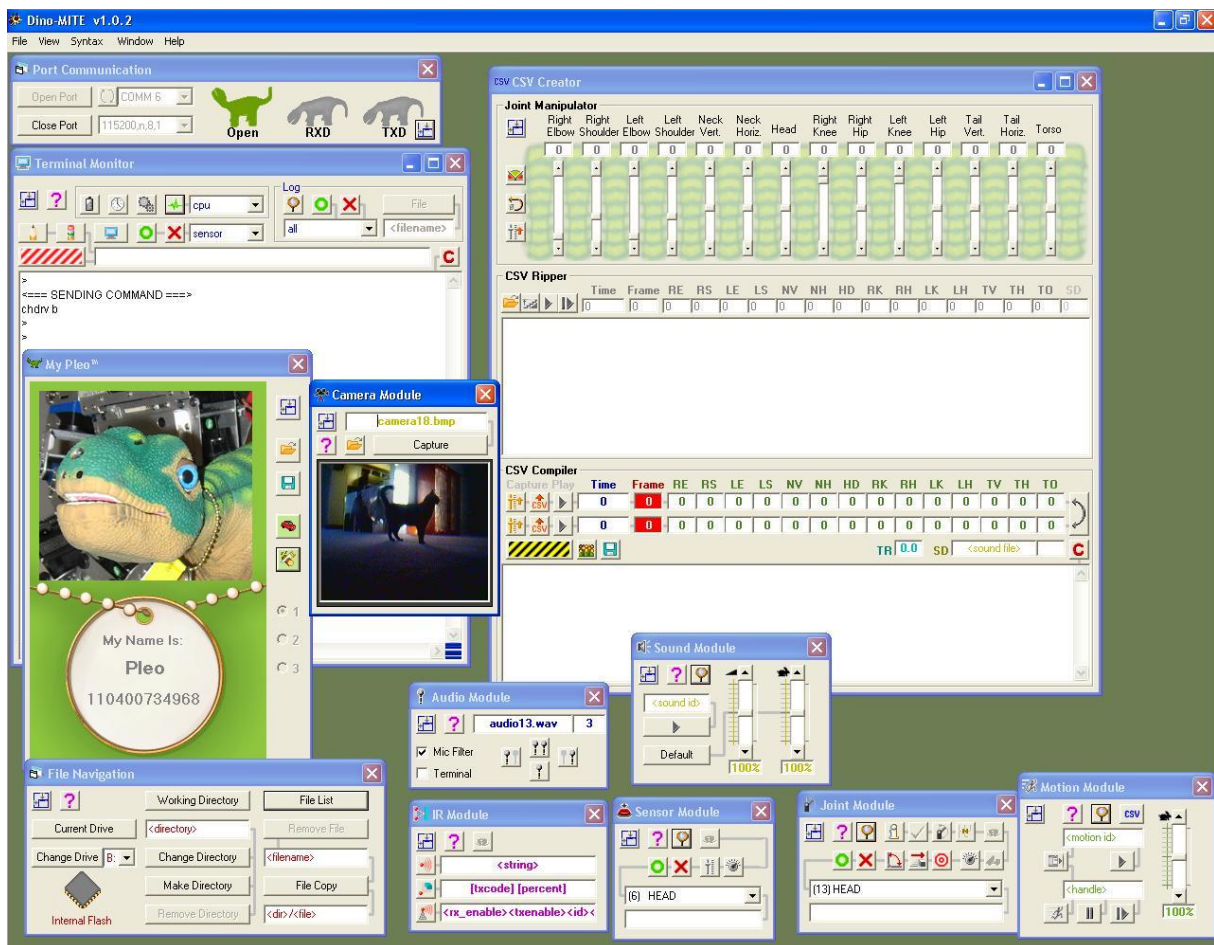
Außerdem ist es möglich mit **MySkit** ganze Persönlichkeitsdateien zu erstellen, die festlegen welche Bewegungsabläufe zum Beispiel bei welcher Berührung ab gespielt werden.

Einer der größten Nachteile von **MySkit** ist, dass nur sequentielle Bewegungsabläufe erstellt werden können, die auch nicht von Sensorwerten oder ähnlichem beeinflusst werden können. Dafür bietet das Programm einen praktischen 3D-Editor, mit dessen Hilfe die Bewegungsabläufe einfach anhand eines 3D-Modells des Roboters nach dem WYSIWYG-Prinzip erstellt werden können.

Um die Bewegungsabläufe abspielen zu können, müssen sie zuerst auf eine SD-Karte kopiert werden, die dann vor dem Anschalten in Pleo eingesteckt werden muss.

¹ <http://www.dogsbodynet.com/myskit/index.html>

2.2.4 DinoMITE



Im Gegensatz zu **MySkitter** baut **DinoMITE** eine direkte Verbindung zum Roboter über den USB-Port auf. Im Das Programm ist ein grafischer Aufsatz für das serielle Protokoll (dazu später mehr). Das Programm hat Zugriff auf fast alle Funktionen des Roboters.

Das Programm bietet folgende Funktionen:

- Direkte Steuerung der einzelnen Motoren
- Auslesen und Setzen von Sensorwerten
- Audio-Aufnahmen
- Speicherung von Kamerabildern
- Abspielen von Emotion
- Wiedergabe von Sound-Dateien
- Senden und Empfangen von Infrarot-Signale
- Zugriff auf das Dateisystem
- Anlegen von personalisierten Profilen für verschiedene Pleos

Bestehende Defizite der Oberfläche von **DinoMITE** sind die nicht sehr übersichtlich und unkomfortablen Eingabemöglichkeiten. Fast alle Parameter müssen von Hand eingegeben werden und die zahlreichen kleinen Icons lassen meist kaum erkennen für welche Funktion sie zuständig sind. Während unseres Tests kam es auch zu einigen Problemen mit der Programmstabilität.

3 Pleopatra API

Auf der Grundlage, die Pleo zur Steuerung und Kommunikation über den USB-Port bietet, haben wir eine Java API entwickelt, die alle wichtigen Funktionen des Roboters einfach über den Computer erreichbar macht und als Grundlage für spätere Applikationen verwendet werden kann.

3.1 Schnittstelle

Die Kommunikation zwischen dem Roboter und der API findet ausschließlich über den seriellen Port statt. Pleo stellt über die serielle Schnittstelle einen umfangreichen Befehlssatz zur Verfügung, der die Steuerung aller wichtigen Funktionen erlaubt. Eine Übersicht über das Protokoll befindet sich im Anhang.

3.1.1 Herstellen einer Verbindung unter Windows

Nachdem der Pleo eingeschaltet wurde, muss er mit dem mitgelieferten USB-Kabel, über den USB-Port an seiner Unterseite, an einen Computer angeschlossen werden.

Über das Windows Standardprogramm **Hyper Terminal** lässt sich dann recht einfach eine Verbindung zu Pleo aufbauen:

Nach dem Start von **Hyper Terminal** (Start > Programme > Zubehör > Kommunikation) eine Neue Verbindung erstellen.

Bei „Verbindung herstellen über“ den COM-Port auswählen, an dem Pleo angeschlossen ist. Im nächsten Fenster folgende Konfiguration verwenden:



Jetzt kann über die im Anhang angegebenen Befehle mit Pleo kommuniziert werden (alternativ zeigt der Befehl „help“ eine Übersicht über die verfügbaren Befehle an).

3.2 Funktionen

3.2.1 Audio

Besonders viele Möglichkeiten bietet die API im Bereich Audio-Aufnahme und Verarbeitung. Normalerweise werden Audio-Aufnahmen auf der SD-Karte gespeichert und können dann per Kartenleser auf einen Computer übertragen werden. Über die serielle Schnittstelle bietet Pleo aber auch die Möglichkeit die Aufnahme direkt als binäre Daten zu streamen. Die **Pleoparta API** kann

diese Daten nicht nur empfangen sondern auch weiterverarbeiten und codieren, zum Beispiel auch direkt als Wav-Datei auf dem Computer speichern. Somit wird es möglich Aufnahmen vom Roboter direkt auf dem Computer zu speichern. Außerdem ist es problemlos möglich die API zu erweitern und somit einen direkten Audio-Stream mit Ausgabe in nahezu Echtzeit zu realisieren.

Die Qualität der Aufnahmen ist für die weitere Verarbeitung, zum Beispiel Spracherkennung, ausreichend, solange der Roboter während der Aufnahme stillsteht. Während der Bewegung sinkt die Qualität, durch die Geräusche der Motoren, merklich.

3.2.2 Streaming Protokoll

Der Audiostream wird in folgender Form geliefert:

Audio Packet Ch=1, buf=1, packet=1, len= 96

[START BINARY DATA]???(348chars)...[END BINARY DATA]

Audio Packet Ch=1, buf=2, packet=1, len= 96

[START BINARY DATA]???(348chars)...[END BINARY DATA]

Audio Packet Ch=1, buf=<n>, packet=1, len= 96

[START BINARY DATA]???(348chars)...[END BINARY DATA]

Je länger die Aufnahme, desto mehr dieser Paket werden empfangen.

Zur Speicherung im wav-Format muss der Header und die Start- und End-Tags entfernt werden und die Zeichen in Bytes umgewandelt werden.

3.2.3 Foto

Von besonderem Interesse für die API war auch die eingebaute Kamera, da diese eine noch stärkere Interaktion zwischen Pleo und seiner Umwelt versprach. In der Praxis hat sich die Kamer aber als nur bedingt tauglich erwiesen.

Die Kamera liefert Bilder im BMP- oder RAW-Format mit einer maximalen Auflösung von 176 x 144 Pixel. Direkt vor der Kamera befindet sich eine Plexiglasscheibe, die die, aufgrund der niedrigen Auflösung eher geringe Qualität, weiter verschlechtert. Die Kamera ist auch gegenüber Bewegungen sehr empfindlich, schon kleinere Bewegungen führen zu Schlierenbildung. Das Aufnehmen und Abspeichern eines Fotos auf der SD-Karte beansprucht ca. 10 bis 15 Sekunden.



Es besteht zwar keine Möglichkeit, aufgenommene Bilder direkt über die serielle Schnittstelle zu streamen, mit Hilfe der cat-Befehle (s. Anhang) können Bilder jedoch nachdem sie auf der SD-Karte gespeichert wurden, auf den Computer übertragen werden und dort abgespeichert oder verarbeitet werden. Gesichtserkennung, oder ähnliches, lässt sich aber, Aufgrund der nicht ausreichenden Bildqualität vermutlich nicht realisieren.

3.2.4 Pleo spezifische Befehle

Auf unterster Ebene bietet die **Pleopatra API** Zugriff auf alle relevanten Befehle, die über die serielle Schnittstelle erreichbar sind, zum Beispiel das Bewegen von Motoren. Der Befehl hierzu lautet:

```
public void moveMotor(int id, int angle)
```

und ruft intern einen nativen Befehl der seriellen Schnittstelle auf.

Darüber hinaus bietet sie weitere, höhere Befehle, die den Umgang mit Pleo erleichtern, indem sie Befehle sinnvoll miteinander kombiniert und so zum Beispiel auch eine Methode zum umbenennen von Dateien auf einer sich im Roboter befindenden SD-Karte ermöglicht, die das serielle Protokoll nicht mitbringt. Der Befehl hierzu lautet:

```
public void renameFile(String oldFilename, String newFilename)
```

und kombiniert den Aufruf von Kopier- und Löschbefehlen der seriellen Schnittstelle.

3.3 Aufbau der API

Im Wesentlichen besteht die API aus zwei Teilen, den Klasse PleoComm und PleoControl. Dabei stellt PleoComm die untere Layer da, sie übernimmt lediglich das Verbinden und das Senden und Empfangen von Strings über die parallele Schnittstelle. Programme, die die API verwenden benötigen keinen Zugriff auf diese Klasse.

Für den Zugriff auf die parallele Schnittstelle wird die **RXTXcomm**-Bibliothek² verwendet.

PleoControl beinhaltet die beiden oberen Layer. Zum einen die Methoden, die die nativen Funktionen des seriellen Protokolls ansprechen, aber auch die Methoden, die die nativen Funktionen kombinieren oder die erhaltenen Daten weiterverarbeiten (zum Beispiel Speicherung von Bildern als BMP, die als binärer Datenstream empfangen werden).

² <http://users.frii.com/jarvi/rxtx/>

4 *Pleopatra Tools*

Pleopatra Tools ist eine erweiterbare Applikation, die auf der Pleopatra API beruht und dem User grafisch ermöglicht auf alle Bestandteile der API zuzugreifen.

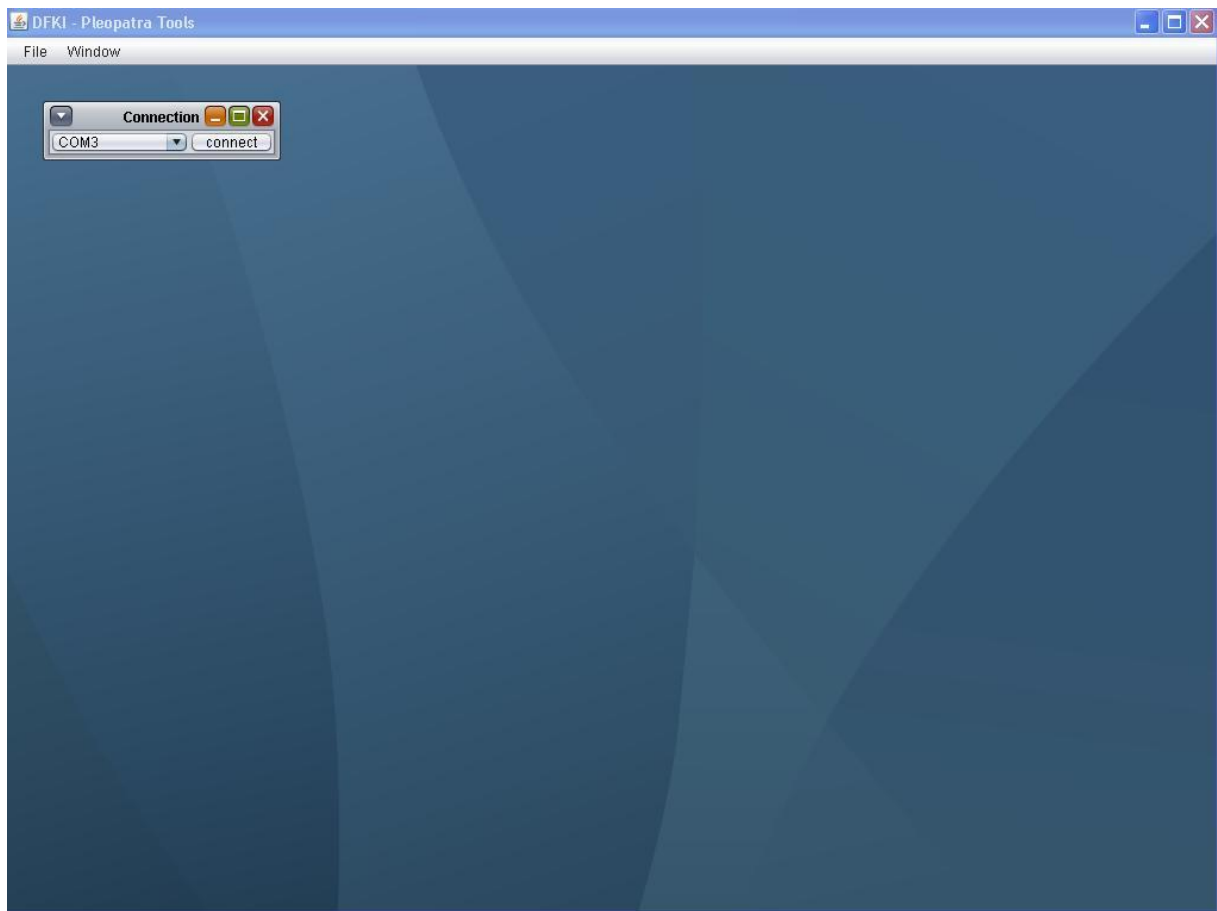
Die Oberfläche setzt sich modular aus verschiedenen Fenstern zusammen, die die einfache Überwachung und Steuerung eines angeschlossenen Pleos erlauben. Gleichzeitig ist das Programm auch ein Prototyp für die Nutzung der API zur Erstellung von Applikationen.

Pleopatra Tools bietet unter anderem folgende Funktionen:

- Anzeiger aller belegten COM-Ports
- Verbindung mit Pleo herstellen, Verbindung trennen
- Speicherung von personalisierten Informationen zum angeschlossenen Pleo
- Anzeigen von Sensorwerten
- Aufnahme von Audiodateien
- Anzeigen und Abspielen von:
 - Sound-Dateien
 - Bewegungs-Dateien
 - Persönlichkeits-Dateien
- Setzen von Sensorwerten
- Aufnahme von Fotos
- Twittern des aktuellen Status, entweder automatisch oder manuell
- usw

Verglichen mit **DinoMITE** liegen die Vorteile von **Pleopatra Tools** vor allem in der einfacheren Bedienung und dem größeren Funktionsumfang, sowie der Stabilität. Darüberhinaus handelt es sich bei den **Pleopatra Tools** um eine quelloffene und leicht erweiterbare Applikation, was eine zukünftige Weiterentwicklung ermöglicht.

4.1 Die Oberfläche

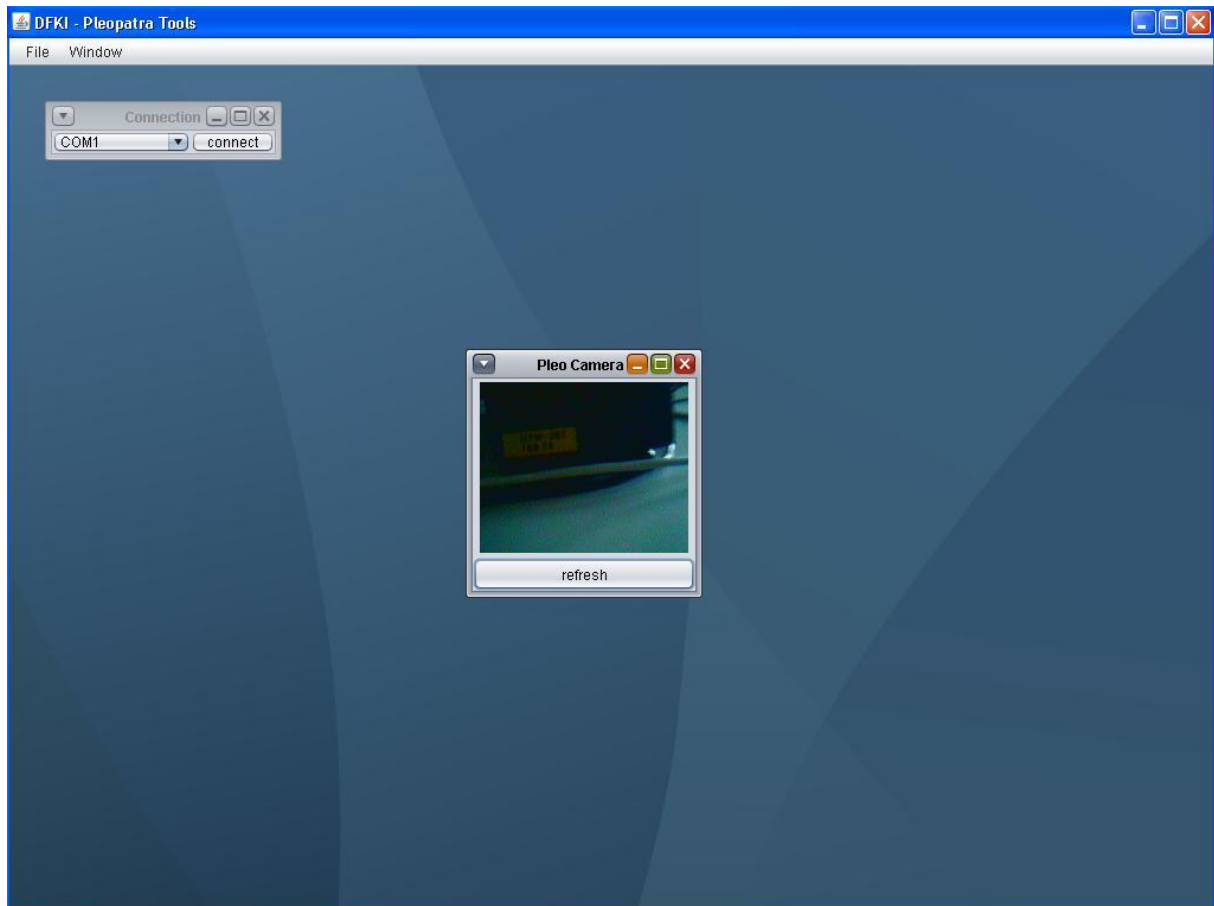


Die Oberfläche der **Pleopatra Tools** besteht aus einer Arbeitsfläche, auf der, je nach Bedarf, die Fenster mit den benötigten Funktionen angezeigt werden können. Die einzelnen Fenster können unter dem Menüpunkt *Window* geöffnet beziehungsweise geschlossen werden.

4.1.1 Connection

Beim Programmstart öffnet sich automatisch das *Connection*-Fenster. In einer Dropdown-Liste werden alle aktiven COM-Ports des Computers angezeigt. Nachdem der Port ausgewählt wurde, an dem der Pleo angeschlossen ist, kann per Klick auf den Button *connect* eine Verbindung hergestellt werden. Sobald eine Verbindung besteht wechselt der Button seine Beschriftung zu *disconnect* und kann ab sofort dazu verwendet werden, die Verbindung zu beenden.

4.1.2 Pleo Camera



Im Fenster *Pleo Camera* können aktuelle Aufnahmen von der Kamera in Pleo angefordert werden. Um die Bildqualität zu verbessern wird Pleo dazu vorübergehend angehalten und läuft erst weiter, nachdem das Bild gemacht wurde. Da keine direkte Übertragung möglich ist, wird das Foto zuerst auf der SD-Karte in Pleo gespeichert und dann auf den PC übertragen. Deshalb kann es ca. 10 bis 15 Sekunden dauern, bis ein Bild angezeigt wird.

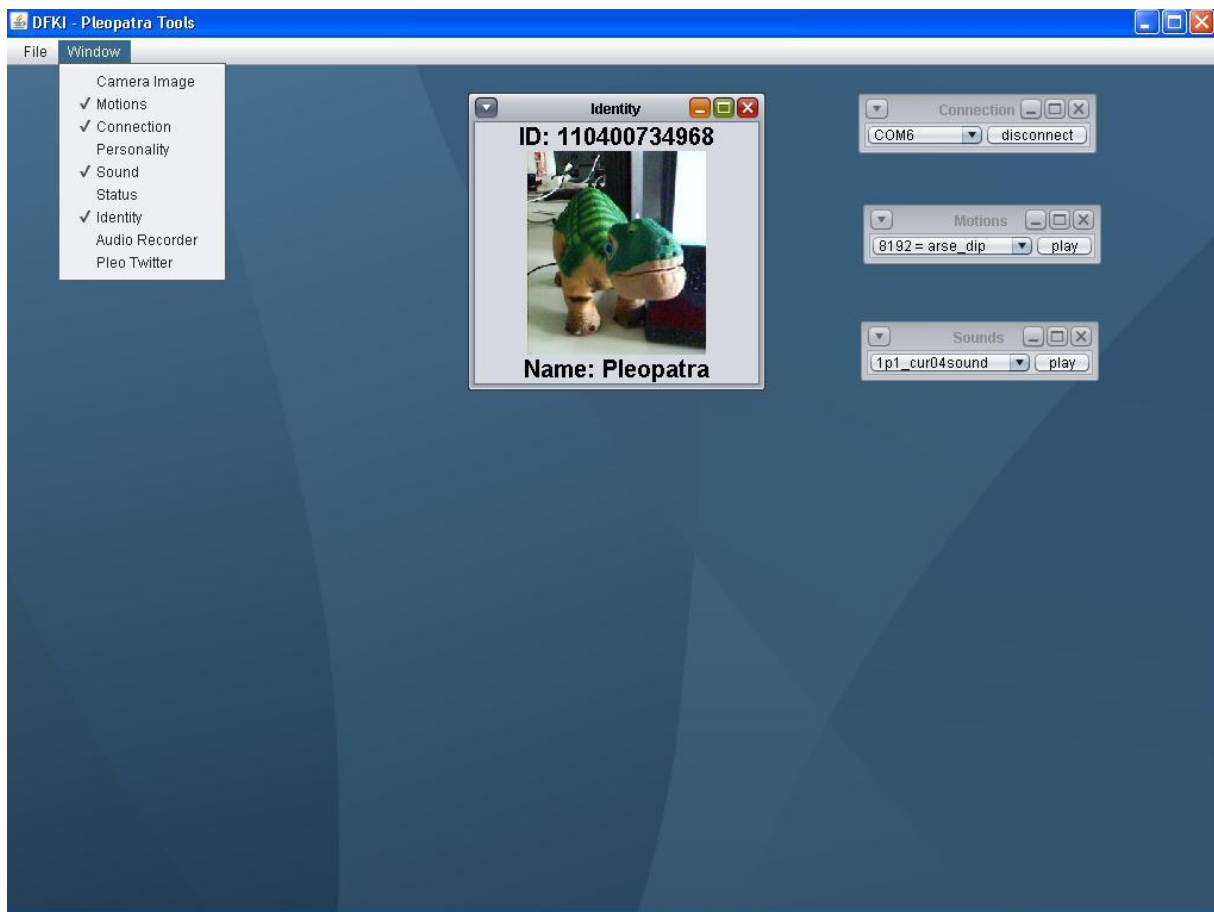
4.1.3 Motions

Motions zeigt eine Dropdown-Liste aller Motion-Files der aktuell laufenden Persönlichkeit an (wie sie zum Beispiel mit **MySkit** erstellt werden können). Durch einen Klick auf *play* können diese auch sofort abgespielt werden.

4.1.4 Personality

Das Fenster *Personality* zeigt eine Übersicht über alle auf der SD-Karte gespeicherten Persönlichkeiten und kann diese per Knopfdruck aktivieren, dazu wird die aktuell laufende Persönlichkeit zuerst gestoppt und dann überschrieben. Außerdem wird zusätzlich immer die Standard- Persönlichkeitsdatei angezeigt.

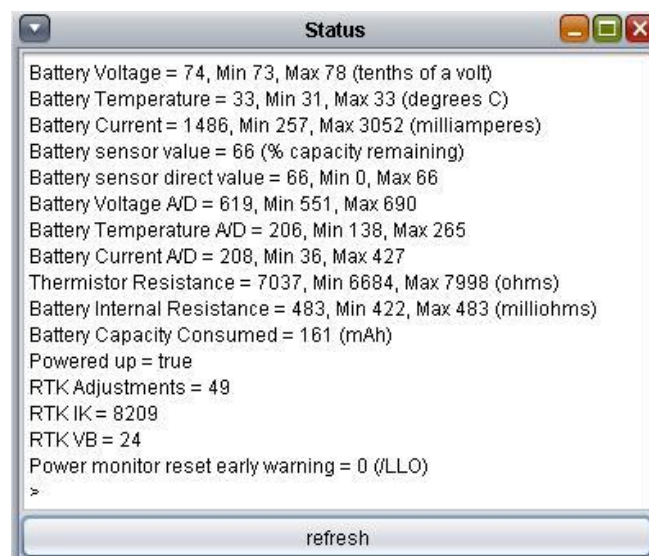
4.1.5 Sound



Ähnlich wie das Fenster *Motions*, besteht auch das Fenster *Sound* aus einer Dropdown-Liste und einem *play*-Button. In diesem Fenster werden alle Sounds der aktuell geladenen Persönlichkeit angezeigt und können per Klick sofort abgespielt werden.

4.1.6 Status

Im *Status*-Fenster werden aktuelle Angaben zum Betriebsstatus des angeschlossenen Pleos angezeigt, wie zum Beispiel Akku-Ladung, Temperatur, Spannung und weitere Informationen.

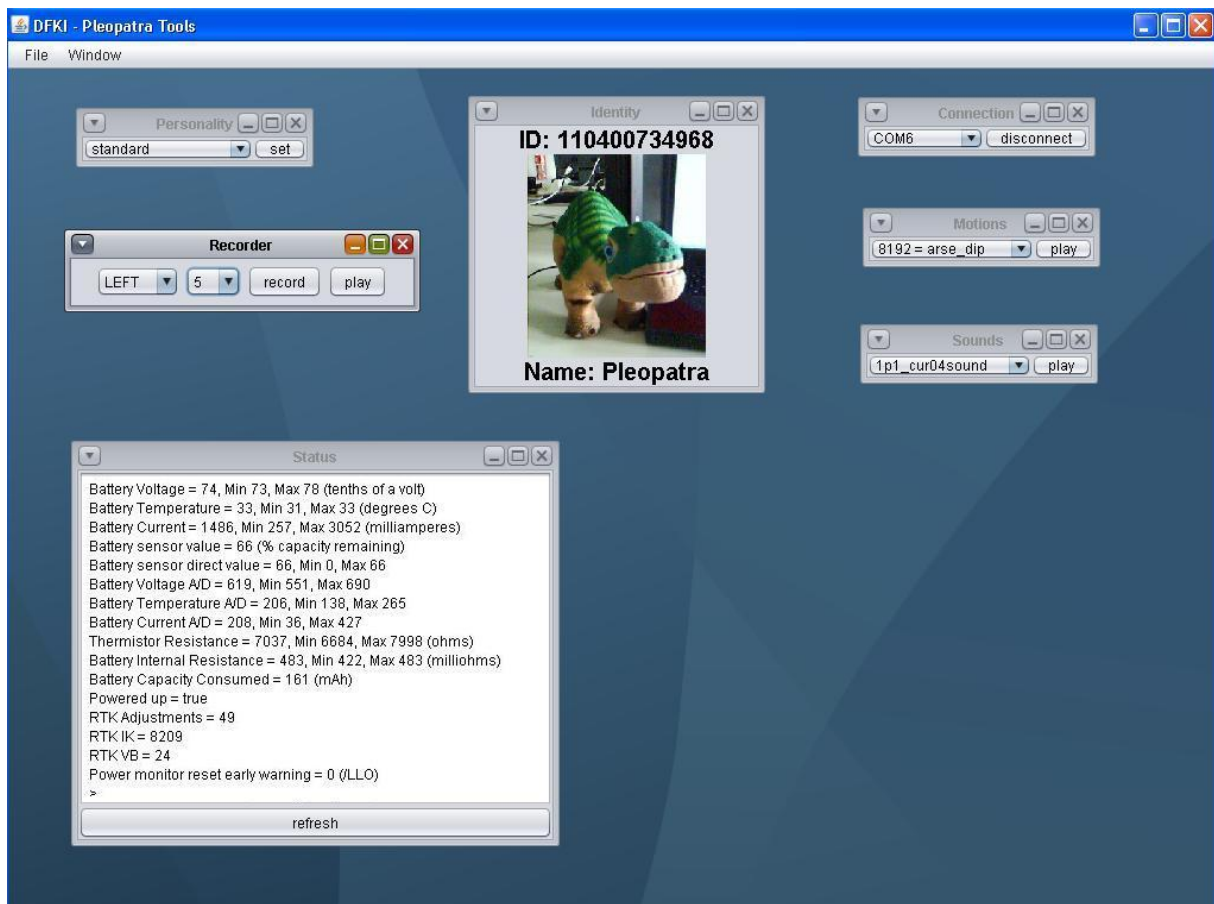


4.1.7 Identity

Das *Identity*-Fenster liest die Seriennummer des angeschlossenen Pleos aus und kann dieser anhand auf dem Computer abgelegter Pleo-Personality Dateien (pleop) einen Namen und ein Bild zuweisen. Ist für den angeschlossenen Pleo keine Personality Datei vorhanden wird nur die Seriennummer und ein Standardbild angezeigt. Ist kein Pleo angeschlossen weißt das Fenster entsprechend darauf hin.



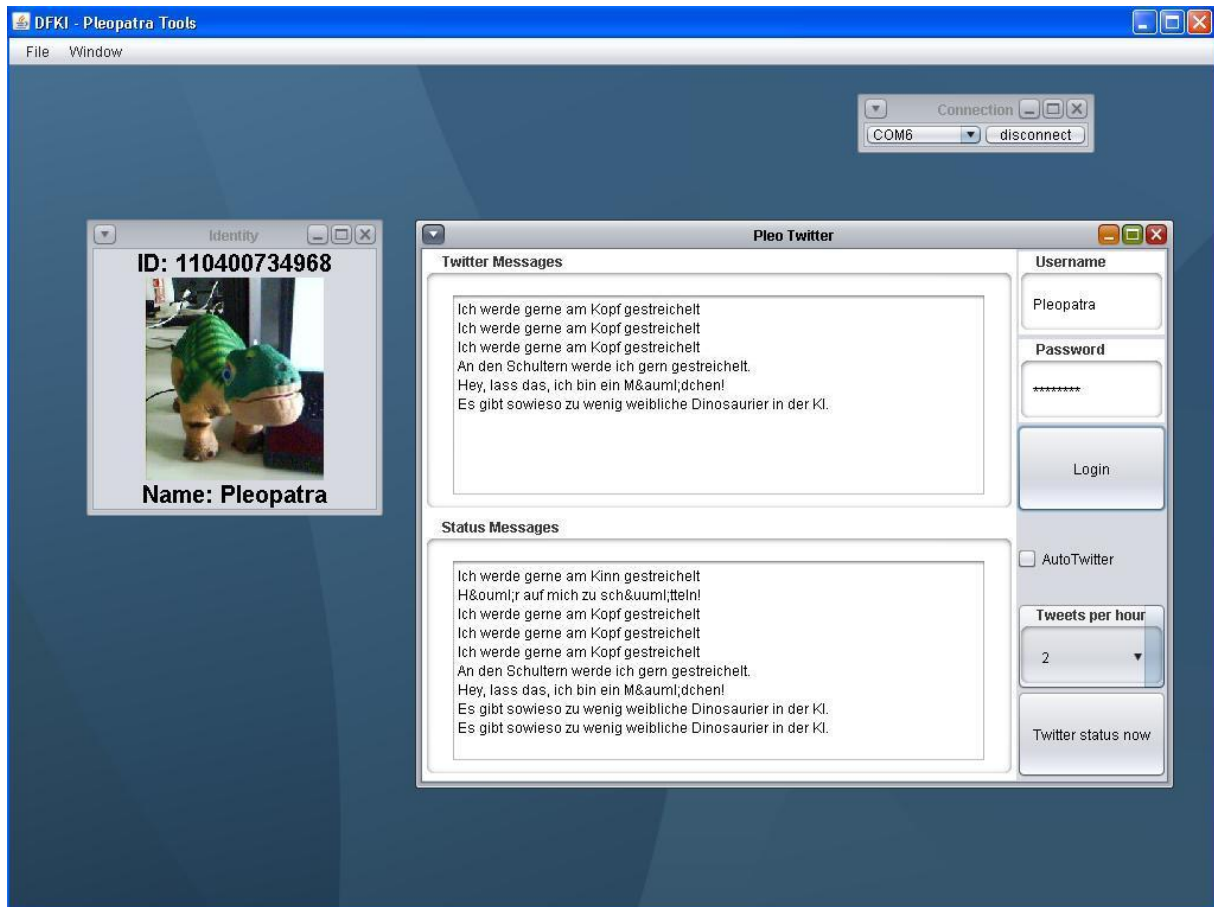
4.1.8 Audio Recorder



Mit Hilfe des Audiorecorders können Aufnahmen von Pleos Mikrofonen gemacht und an den PC übertragen werden. Die Audiodaten sind nahezu sofort nach Ablauf der Aufnahmezeit verfügbar und

können über den Button *play*, sofort im selben Fenster angehört werden. Für die Aufnahme können sowohl die Mikrofone zur Aufnahme, als auch die Aufnahmezeit eingestellt werden.

4.1.9 Pleo Twitter



Der letzte Eintrag im *Window*-Menü öffnet das Fenster mit dem Namen *Pleo Twitter*, das die Schnittstelle zwischen Pleo und Twitter bildet. Dafür muss man sich zuerst bei Twitter anmelden. Dabei wird als *Username* standardmäßig der Name aus der pleop-Datei verwendet. Nach der Eingabe des Passworts wird durch den Button *Login* eine Verbindung zu Twitter aufgebaut. Ab sofort werden im unteren Teil des Fensters aktuelle Statusmeldungen des angeschlossenen Pleos angezeigt. Durch einen Klick auf *Twitter status now*, wird die letzte erhaltene Nachricht bei Twitter veröffentlicht.

Alternativ kann über die Checkbox *AutoTwitter* auch ein Zeitintervall angegeben werden, nach dem der aktuelle Status automatisch getwittert werden soll (maximal alle 3 Minuten ~ 20 Tweets pro Stunde; minimal alle halbe Stunde ~ 2 Tweets pro Stunde).

Genauere Informationen zum Absenden und Erstellen der Twitter-Nachrichten folgen im nächsten Kapitel.

5 Pleopatra twittert

Als erste (und ursprünglich eigenständige) Applikation wurde ein Programm implementiert, das es Pleo erlaubt eigenständig, aufgrund der vorliegenden Sensordaten, zu „twittern“. Hierzu greift das Programm auf eine Reihe vorgefertigter Nachrichten (sog. „Tweets“) zurück und wählt daraus die zu einem aktuellen Ereignis passenden aus, ergänzt sie gegebenenfalls durch Sensorwerte und veröffentlicht sie dann auf der Plattform Twitter.

Da bei liegt der Vorteil von Twitter hauptsächlich in der aktuell hohen Popularität und der damit verbundenen Aufmerksamkeit. Außerdem ist die Anwendung ein gutes Beispiel, für die Möglichkeiten der API.

5.1 Twitter API

Um die Nachrichten später veröffentlichen zu können wurden erst verschiedene Twitter APIs für Java geprüft. Von den drei zurzeit angebotenen Java APIs viel die Wahl dann auf **jtwtiter**³, da diese API die einfachste Implementierung bei ausreichendem Funktionsumfang bot.

5.2 Funktionsweise

Da die Tweets auf den aktuellen Sensorwerten basieren verfügt das Programm über eine Klasse, das in festgelegten, regelmäßigen, Abständen alle Sensordaten aktualisiert. Die Daten werden dann nach „Auffälligkeiten“, also bestimmten Werten, durchsucht. Dabei gibt es eine Reihenfolge, die die Wichtigkeit von Ereignissen festlegt und somit entscheidet, welches Ereignis getwittert wird, wenn mehrere Ereignisse gleichzeitig stattfinden.

Wird kein besonderes Ereignis gefunden, gibt das Programm eine Idle-Message aus, die allerdings nicht getwittert wird. Dabei ist der Abstand, in dem neue Nachrichten verschickt werden sollen einstellbar.

Wird ein auffälliger Wert gefunden wird aus einer Liste von Tweets die zu dem Ereignis passenden Tweets geladen. Per Zufall wird aus den vorhandenen Tweets eine Auswahl getroffen. Ein Parser durchsucht den Tweet dann nach einem Platzhalter und ersetzt diesen, falls vorhanden, durch den aktuellen Sensorwert.

Danach wird noch einmal geprüft, ob das Ereignis tatsächlich neu ist, oder ob bereits ein Tweet mit gleichem Inhalt verschickt wurde. Ist das nicht der Fall, wird die Nachricht an **jtwtiter** übergeben und veröffentlicht.

5.3 Twitter Profil

Das Twitter-Profil ist unter <http://twitter.com/pleopatra> erreichbar. Dort stieß das Projekt innerhalb kurzer Zeit auf große Begeisterung.

Pleopatra wurde von ihren Followern zur „Tweeter Wall“⁴



³ <http://www.winterwell.com/software/jtwtiter.php>

in der Kategorie „Top Animal“ nominiert und konnte dort, trotz verspätetem Start in die Abstimmung, über 200 Stimmen auf sich vereinen.

⁴ <http://tweeterwall.mallplace.com>

6 Zusammenfassung & Ausblick

6.1 Zusammenfassung

Zu Beginn stellte es sich schnell als großes Problem heraus, dass es kaum Dokumentation rund um Pleo gab und man somit auf eigene, und die Erfahrungen anderer Nutzer angewiesen war. Da in naher Zukunft nicht damit zu rechnen ist, dass sich die Möglichkeit ergibt, Pleo direkt zu programmieren, wurde schnell klar, dass große Teile auf den Computer ausgelagert werden müssen. So fiel schnell die Entscheidung sich auf die Kommunikation über die serielle Schnittstelle zu konzentrieren, da das die vielversprechendste Möglichkeit bot Pleo zu steuern, auch wenn dies einige Nachteile mit sich bringt, zum Beispiel, dass Pleo ständig mit dem Computer verbunden sein muss.

Trotzdem konnte letztendlich gezeigt werden, dass Pleo noch zahlreiche ungenutzte Ressourcen hat, die eine noch stärkere Interaktion mit seiner Umgebung, besonders mit Menschen, ermöglichen.

6.2 Ausblick

In dieser Richtung besteht auch weiterhin großes Potential, so kann durch die Möglichkeit des Audio-Streamings relativ problemlos eine bereits bestehende Lösung zur Spracherkennung genutzt werden, um Pleo auch auf Sprachkommandos reagieren zu lassen.

6.2.1 Twitter

Da Pleo auch Audiodateien wiedergeben kann wäre es auch möglich, das Twitter-Konzept durch eine Sprachausgabe zu erweitern, die aus einer abgelegten Datenbank aus Sound-Dateien die passenden Ausschnitte aussucht und wiedergibt.

Vorstellbar wäre auch die Tweets individuell und zur aktuellen Situation passend zu generieren, anstatt auf eine Datenbank vorgefertigter Tweets zurückzugreifen um ständige Wiederholungen zu vermeiden.

In diesem Zusammenhang wäre auch die Integration eines Diskursgedächtnis interessant, um aktuelle Ereignisse besser bewerten und einordnen zu können.

Durch die API des Dienstes TwitPic⁵ könnten auch aktuelle Bilder der Kamera, passend zum jeweiligen Sensorstatus, in die Tweets integrierten werden.

6.2.2 Hardware

Einer Gruppe der Uni Bamberg ist es auch gelungen die Kamera von Pleo zu ersetzen, durch die neue Kamera kann das Bild in wesentlich besserer Qualität sofort an den Computer gestreamt werden.⁶ Kombiniert mit einer Bilderkennung und der **Pleopatras API** würden sich so weitere Möglichkeiten, wie zum Beispiel Gesichtserkennung ergeben.

⁵ <http://www.twitpic.com>

⁶ http://www.grip-online.com/de/pleo_hack/spycam

Besonders für den praktischen Einsatz ist auch die Möglichkeit interessant, an einen der internen seriellen Anschlüsse eine Bluetooth- oder Funkmodul anzuschließen, damit der Roboter nicht ständig per Kabel mit dem PC verbunden sein muss und somit die Reichweite erhöht werden kann.⁷

6.2.3 API / Pleopatra Tools

Auch eine Erweiterung der Pleopatra Tools bietet sich an, da noch einige Funktionen der API nicht durch die Anwendung genutzt werden, wie zum Beispiel Zugriffe auf das Dateisystem.

Durch die Übernahme von Pleo durch die Firma Jetta⁸, bleibt auch die Hoffnung darauf, dass in absehbarer Zukunft doch noch das PDK erscheint und es somit auch möglich wird Programme direkt auf dem Roboter auszuführen, ohne das eine Verbindung bestehen muss, in Verbindung mit der **Pleopatra API** würden sich somit weitere Möglichkeit ergeben, da man dann nur noch rechenintensive Vorgänge (wie zum Beispiel Spracherkennung) auf den Computer auslagern müsste.

⁷ http://www.grip-online.com/de/pleo_hack/xbee

⁸ <http://www.jetta.com.hk>

7 Anhang

7.1 Schnittstellenprotokoll

Die folgenden Befehle ermöglichen die Steuerung eines per USB an einen Computer angeschlossenen Pleos. Mit einem entsprechenden Programm (z.B. **Hypter Terminal** oder **Putty**) kann man sich mit folgenden Parametern zu Pleo verbinden:

- Bits pro Sekunde: 115200
- Datenbits: 8
- Parität: Keine
- Stoppbits: 1

7.1.1 Befehlsübersicht:

7.1.1.1 Persönlichkeitsdateien

app autorun 0/1

Legt fest, ob eingelegte Persönlichkeitsdateien automatisch gestartet werden sollen.

app load

Lädt die Persönlichkeitsdatei mit dem Namen „Pleo.urf“ von einer eingelegten SD-Karte.

app unload

Schließt geladene Persönlichkeitsdateien (Pleo steht still).

7.1.1.2 Aufnahme

audio average filename seconds

Nimmt für die angegebene Zeit das Mittel von beiden Mikrofonen auf und speichert die Aufnahme unter dem angegebenen Dateinamen.

audio both filename seconds

Nimmt für die angegebene Zeit auf beiden Mikrofonen auf und speichert die Aufnahme unter dem angegebenen Dateinamen.

audio left filename seconds

Nimmt für die angegebene Zeit vom linken Mikrofonen auf und speichert die Aufnahme unter dem angegebenen Dateinamen.

audio right filename seconds

Nimmt für die angegebene Zeit vom rechten Mikrofonen auf und speichert die Aufnahme unter dem angegebenen Dateinamen.

audio terminal on/off

Schaltet das Streaming von aufgenommenen Daten in die Konsole an bzw. ab.

micfilter on / off

Schaltet Mikrofon-Aufnahmen während sich Pleo bewegt komplett an bzw. aus.

7.1.1.3 Kamera

camera average *ulx uly lrx lry*

Festlegen der Frame-Größe.

camera capture *filename raw/bmp new/last*

Macht ein Foto (new) bzw. nimm das letzte Foto (last) und speichert es unter dem angegebenen Dateinamen als BMP- oder RAW-Datei.

camera color *mono / yuv / rgb*

Setzt den Farbmodus der Kamera.

camera getobjects

Zeigt die aktuell verfolgten Objekte an.

camera terminal *on / off*

Ohne Funktion.

camera window *x y*

Legt die Größen der Kameraaufnahme fest.

7.1.1.4 Dateisystem

cat *filename*

Gibt den Inhalt der angegebenen Datei auf der Konsole aus.

cd *directory*

Öffnet das angegebene Verzeichnis.

chdrv *A / B*

Wechselt auf das angegebene Laufwerk (a = SD-Karte; b = Flash).

copy *source destination*

Kopiert die angegebene Quelldatei in die angegebene Zieldatei.

curdrv

Gibt das aktuell geöffnete Laufwerk aus (a = SD-Karte; b = Flash).

ls

Zeigt den Inhalt des geöffneten Ordners an.

pwd

Zeigt das aktuelle Verzeichnis an.

rm *filename*

Löscht die angegebene Datei.

rmdir *directory*

Löscht den angegebenen Ordner.

sdcard *on / off*

Schaltet die SD-Karte ein bzw. aus.

receive filename byte chunksize

Im Test konnte keine Funktion festgestellt werden.

upgrade filename

Installiert das angegebene Firmware-Update.

xcopy source destination

7.1.1.5 Infrarot

ir beacon rx_enable tx_enable id data1 data2 data3

Infrarot-Leuchtfener anschalten.

ir object txcode percent

Infrarot-Objekt aktivieren.

ir send string

Sendet den angegebenen String über den Infrarot-Sender.

ir watch

Wartet auf eine Infrarot-Übertragung.

7.1.1.6 Motoren

joint angle motor angle timeout allozged_time

Setzt den Motor innerhalb der vorgegebenen Zeit auf den angegebenen Winkel.

joint check motor

Überprüft Kalibrierung des angegebenen Motors auf Gültigkeit.

joint deadband motor deadband

Setzt den maximalen Bewegungsradius des angegebenen Motors.

joint disable motor

Schaltet den angegebenen Motor ab.

joint enable motor

Schaltet den angegebenen Motor ein.

joint dump filename

Lädt die angegebene Kalibrierungseinstellung.

joint info motor

Zeigt den aktuellen Status des angegebenen Motors an.

joint move motor angle

Bewegt den angegebenen Motor um den angegebenen Winkel.

joint neutral

Bewegt alle Motoren in die Ausgangsstellung.

joint show motor

Zeigt den aktuellen Status des angegebenen Motors an.

joint target motor target time_ms

Bewegt den angegebenen Motor in der vorgegebenen Zeit zum angegebenen Ziel.

joint tune motor low_target high_target

Bewegt den angegebenen Motor zwischen den beiden Zeilen hin und her.

7.1.1.7 Log

log color on / off

Schaltet Farbe beim Logging an bzw. ab.

log device null / usb / serial

Festlegung des Log-Device.

log disable event

Schaltet das Logging für das angegebene Event ab.

log enable event

Schaltet das Logging für das angegebene Event an.

log eol dos / unix

Schaltet das Logging Dateieinde des Logfiles auf Dos bzw. Unix.

log file filename / off

Speichert das Logfile in der angegebenen Datei bzw. erstellt kein Logfile.

log status

Zeigt eine Liste der Events, die geloggt werden.

log usb_enable

Gibt den Log auch über die USB-Verbindung aus.

log help

Zeigt die Hilfe an.

7.1.1.8 Monitoring

monitor disable sensor / motor / sound / motion / all

Deaktiviert das Monitoring für das angegebene Modul.

monitor enable sensor / motor / sound / motion / all

Aktiviert das Monitoring für das angegebene Modul.

monitor mode on / off

Schaltet das Monitoring an bzw. aus.

7.1.1.9 Motion-Files

motion csv

Startet / Stoppt die Speicherung des Bewegungsablaufs in einer CSV-Datei.

motion dump

Zeigt Informationen über die aktuelle geladenen Bewegungsdateien.

motion load filename

Lädt die angegebene Bewegungsdatei.

motion pause motion_handle

Stoppt die angegebene Bewegung.

motion play motion_name

Spielt die angegebene Bewegung.

motion run motion_handle loop

Startet die angegebene Bewegung, ggf. mit Wiederholungen.

motion show

Zeigt alle Bewegungsdateien.

motion speed percent

Setzt die Abspielgeschwindigkeit für die Bewegungsdateien.

motion step motion_handle

Setzt Motion-Handler auf angegebenen Wert.

motion stop motion_handle

Stoppt die angegebene Bewegung.

7.1.1.10 Package***pkt baud***

Baudrate setzen.

pkt burn filename

Datei „brennen“.

pkt reset

Header zurücksetzen.

pkt send

Packet senden.

pkt stats

Packet-Status anzeigen.

pkt test

Packet test.

7.1.1.11 Power***power down / up***

Schaltet Pleo an bzw. aus.

pthresh 0 - 16

Legt fest ab welchem Ladestatus sich Pleo automatisch abschalten soll.

tthresh temperature

Legt fest, ab welcher Temperatur Pleo sich automatisch abgeschalten soll.

7.1.1.12 Sensoren

sensor disable sensor

sensor enable sensor

sensor setlevel sensor value

Setzt den Wert des Sensor auf die angegebene Zahl.

sensor show sensor

Zeigt den aktuellen Sensorwert und weitere Informationen an.

sensor watch sensor

Zeigt den Sensorwert und weitere Informationen so lange an, bis eine Taste gedrückt wird.

7.1.1.13 Sound-Dateien

sound play filename

Spielt das Sound-File mit dem angegebenen Dateinamen.

sound show

Zeigt alle verfügbaren Sound-Files.

sound speed percent

Gibt an, mit wie viel Prozent der Originalgeschwindigkeit die Sound-Files abgespielt werden sollen.

sound volume percent

Gibt an, mit wie viel Prozent der Originallautstärke die Sound-Files abgespielt werden sollen.

7.1.1.14 Sonstiges

stats all/cpu/fs/hist/mem/power/sd/sys/toshiba/vm

Zeigt die Statistik für das angegebene Modul (bzw. für alle) an.

time

Gibt die Uptime an.

clear

Leert die Ausgabe der Konsole.

config

Zeigt die Konfiguration an.

passthru port0baud port1baud

Seriellensignal durchschleifen.

reset

Startet alle Systeme (Motoren, Kamera, Audio, Persönlichkeit etc.) neu.

7.1.2 Übersicht Motoren⁹

Nr	Motor
0	Bein VL oben
1	Bein VL unten
2	Bein VR oben
3	Bein VR unten
4	Bein HR oben
5	Bein HR unten
6	Bein HL oben
7	Bein HL unten
8	Hüfte
9	Schwanz horizontal
10	Schwanz vertikal
11	Hals horizontal
12	Hals vertikal
13	Augen + Mund

7.1.3 Übersicht Sensoren

Nr	Sensoren
0	
1	
2	Batterie
3	
4	
5	
6	Kopf
7	Kinn
8	Schulter
9	Bein HR
10	Bein HL
11	Bein VR
12	Bein VL
13	Rücken
14	Fuß VR
15	Fuß VL
16	Fuß HR

⁹ VL = vorne links; VR = vorne rechts; HL = hinten links; HR = hinten rechts

17	Fuß HL
18	SD-Karte
19	Schreibschutz SD-Karte
20	
21	Helligkeit
22	
23	
24	Mund
25	
26	
27	
28	
29	
30	
31	
32	USB-Kabel
33	Aufwachen
34	Batterie Temperatur
35	
36	Lagesensor
37	
38	
39	

7.1.4 Übersicht Log-Events

FATAL_ERROR
ERROR
WARN
INFO
DEBUG
low
high
vm
monitor
mi
mu
sensor
joint
motion
sound
timing
io

7.2 Pleopatra API

7.2.1 Befehlsübersicht:

Die Pleopatra API befindet sich im Package: *de.dfki.pleopatra.api*

7.2.1.1 Sensoren

public void setSensor(int id, int value)

Setzt den Wert des Sensors mit der angegebenen ID auf den angegebenen Wert.

public void setSensorEx(int id, int triggerlevel, int auxtriggerlevel, int debounce)

Setzt den Sensorwert und zusätzliche Werte.

public int sensorValue(int id)

Gibt den Wert des angegebenen Sensor zurück.

public void enableSensor(int id)

Schaltet den angegebenen Sensor ein.

public void disableSensor (int id)

Schaltet den angegebenen Sensor aus.

public boolean isEating()

Gibt true zurück, wenn Pleo isst, ansonsten false.

7.2.1.2 Motoren

public void moveMotor(int id, int angle)

Bewegt den angegebenen Motor um den angegebenen Winkel.

public void enableMotor(int id)

Schaltet den Motor mit der angegeben ID ein.

public void disableMotor(int id)

Schaltet den Motor mit der angegeben ID aus.

public void setDeadband(int id, int angle)

Setzt den toten Bereich für den angegebenen Motor.

public String motorInfo(int id)

Zeigt Informationen zum angegebenen Motor.

public String showMotor(int id)

Gibt den aktuellen Status des angegebenen Motors zurück.

public void motorPatrol(int id, int start, int end)

Bewegt den angegebenen Motor zwischen den beiden angegebenen Punkten hin und her.

public String checkMotor(int id)

Prüft die Kalibrierung des angegebenen Motors auf Gültigkeit.

public void watchMotor(int id)

Zeigt den Status des angegebenen Motors an.

public void moveMotorTime(int id, int angle, int timeout)

Bewegt den Motor in der angegebenen Zeit um den angegebenen Winkel.

public void moveMotorTarget(int id, int target, int timeout)

Bewegt den Motor in der angegebenen Zeit zum angegebenen Ziel.

public void loadCalibration(String filename)

Lädt die angegebene Motor-Kalibrierung.

7.2.1.3 Audio

public void playSound(int id)

Spielt die Sound-Datei mit der angegebenen ID ab.

public void micFilter(boolean status)

Schaltet den Mikrofonfilter an bzw. aus.

public void streamAudio(boolean status)

Schaltet das Audio-Streaming an bzw. aus.

public void recordAudio(String mic, String filename, int seconds)

Nimmt von den angegebenen Mikrofonen (left, right, both, average) für die angegebene Zeit in die angegebene Datei auf.

public void setPlaybackVolume(int volume)

Gibt an, mit wie viel Prozent der Originallautstärke Audiodateien abgespielt werden sollen.

public void setPlaybackSpeed(int speed)

Gibt an, mit wie viel Prozent der Originalgeschwindigkeit Audiodateien abgespielt werden sollen.

public String getSoundFiles()

Zeigt alle vorhandenen Audiodateien an.

public byte[] getAudio(int seconds, String mic)

Nimmt einen Audio-Stream der angegebenen Länge auf und wandelt ihn in ein Array von Bytes um.

public void saveAudioToPC(int seconds, String mic, String filename)

Speichert eine Aufnahme der angegebene Länge mit den angegebenen Mikrofonen (left, right, both, average) unter dem angegebenen Pfad auf dem Computer.

7.2.1.4 Bilder

public void capturePicture(String filename, String format)

Nimmt ein Foto auf und speichert es unter dem angegebenen Dateinamen und im angegebenen Format (raw oder bmp).

public byte[] getPicture(String filename)

Gibt das angegebene Foto als Array von Byte aus.

public void savePictureToPC(String filename, String saveName)

Speichert das angegebene (auf der SD-Karte gespeicherte) Foto unter dem angegebenen Namen auf dem Computer.

public void pictureSetSize(int x, int y)

Legt die Größe des Kameraausschnitts fest.

public String trackedObjects()

Gibt die verfolgten Objekte zurück.

public void setColormode(String mode)

Setzt den Farbmodus der Kamera (*mono / yuv / rgb*).

7.2.1.5 Bewegungsdateien

public void playMotion(int id)

Spielt die Bewegungsdatei mit der angegebenen ID ab.

public void setMotionSpeed(int speed)

Gibt an mit wie viel Prozent der Originalgeschwindigkeit die Bewegungsdateien abgespielt werden sollen.

public void runMotion(int motion_handle)

Startet die angegebene Bewegungsdatei.

public void stopMotion(int motion_handle)

Stopt die angegebene Bewegungsdatei.

public void pauseMotion(int motion_handle)

Pausiert die angegebene Bewegungsdatei.

public void stepMotion(int motion_handle)

Überspringt die angegebene Bewegungsdatei.

public String createCSV()

Erstellt eine CSV Datei aus den Bewegungen des Roboters.

public String motionInfo()

Zeigt alle Bewegungsdateien an.

7.2.1.6 Sonstiges

public void powerUp()

Schaltet alle Systeme von Pleo an.

public void powerDown()

Schaltet alle Systeme von Pleo ab.

public String info()

Zeigt die aktuelle Konfiguration an.

public String help()

Gibt den Inhalt des Hilfebildschirms als String zurück.

public void clear()

Löscht die Konsolenausgabe.

public String stats(String selection)

Zeigt die Statistik der angegebenen Module an (mögliche Module: *all/cpu/fs/hist/mem/power/sd/sys/toshiba/vm*)

public String uptime()

Gibt die Uptime zurück.

public void setTemperatureThreshold(int temperature)

Setzt die Abschalttemperatur auf den angegebenen Wert.

public void setPowerThreshold(int power)

Setzt den Grenzwert für die Batterieladung, ab der Pleo automatisch abgeschaltet wird.

public void reset()

Startet alle Systeme (Motoren, Kamera, Audio, Persönlichkeit etc.) neu.

serialPassthru(int baud0, int baud1)

Schleift das Signal von einem, zum anderen Seriellen Port durch.

public void sdcardOn()

Schaltet die SD-Karte ein.

public void sdcardOff()

Schaltet die SD-Karte ab.

public String getSerialNumber()

Gibt die Seriennummer des aktuell angeschlossenen Pleos zurück.

7.2.1.7 Dateisystem

public void openDirectory(String dirname)

Öffnet den angegebenen Ordner.

public void setDefaultDrive(char drivename)

Setze Standardlaufwerk (A: SD-Karte, B: Flash).

public void copyFile(String source, String destination)

Kopiert die angegebene Datei zum angegebenen Ziel.

public void xCopy(String source, String destination)

Kopiert das angegebene Verzeichnis mit allen Unterverzeichnissen.

public String getCurrentDrive()

Zeigt das aktuell aktive Laufwerk an.

public void renameFile(String oldFilename, String newFilename)

Benennt die angegebene Datei um.

public String ls()

Gibt den Inhalt des geöffneten Ordners zurück.

public void deleteDir(String dirname)

Löscht das angegebene Verzeichnis.

public void delete(String filename)

Löscht die angegebene Datei.

public void upgrade(String filename)

Installiert das angegebene Firmware-Update.

public void cat(String filename)

public String getWorkingDirectory()

Gibt das aktuell geöffnete Verzeichnis zurück.

7.2.1.8 Log

public String getLogStatus()

Gibt den aktuellen Log-Status zurück.

public void enableLog(String event)

Aktiviert Logging für die angegebenen Events.

public void disableLog(String event)

Deaktiviert Logging für die angegebenen Events.

public void colorLogging(boolean status)

Aktiviert bzw. deaktiviert Color-Logging.

public void setLogfile(String filename)

Setzt den Pfad und den Dateinamen für das Log-File.

public void enableUSBLog()

Schaltet Logging über den USB-Anschluss ein.

public void setLogEOLFormat(String format)

Setzt das Zeilenende des Logs (dos oder unix).

public String showLogEOLFormat()

Gibt das aktuelle Zeilenende der Log-Datei zurück.

public void setLogDevice(String device)

Gibt an, welche Schnittstellen geloggt werden sollen.

7.2.1.9 Persönlichkeiten

public void autorunPresonality(boolean status)

Legt fest, ob eingelegte Persönlichkeiten automatisch gestartet werden sollen.

public void loadPersonality()

Lädt eine eingelegte Persönlichkeitsdatei.

public void unloadPersonality()

Stoppt die aktuell laufende Persönlichkeitsdatei.

public void changePersonality(String filenameNewP, String filenameOldP)

Tauscht die aktuell laufende Persönlichkeit (filenameOldP) gegen eine neue aus (filenameNewP).

public String[] getPersonalities()

Gibt alle, auf der SD-Karte gespeicherten, Persönlichkeitsdateien aus.

public void setPersonality(String personality)

Startet die angegebene Persönlichkeit.

7.2.1.10 Infrarot

public void irSend(String message)

Sendet die angegebene Nachricht über den Infrarot-Sender.

public void irToMonitor()

Infrarot-Empfang überwachen.

public void setIRObjectmonitor(int txcode, int percent)

Zeigt verfügbare Infrarot-Objekte an.

public void setIRBeacon(int rx_enable, int txenable, int id, int data1, int data2, int data3)

Startet Infrarot-„Leuchtfeuer“

7.2.1.11 Monitoring

public void monitorOn()

Monitor-Modus anschalten.

public void monitorOff()

Monitor-Modus ausschalten.

public void monitorEnable(String selection)

Aktiviert das Monitoring für das angegebene Modul (*sensor / motor / sound / motion / all*).

public void disableEnable(String selection)

Deaktiviert das Monitoring für das angegebene Modul (*sensor / motor / sound / motion / all*).

7.2.1.12 Packets

public void setBaud(int baud)

Setzt die Baudrate.

public void burnFile(String filename)

„Brennt“ die angegebene Datei.

public void resetPKTHead()

Setzt den Packet-Header zurück.

public void pktTest()

Packet-Test.

public String pktStat()

Gibt den Packet-Status zurück.

public void sendPKT(String myPackage)
Versendet das angegebene Packet.