S 8-1
Pandaboard, TurtleBot, Kinect und Co.

Low-Cost Hardware im Lehreinsatz für die mobile Robotik

Von Stephan Kallweit[1]

1 Abstract

Die mobile Robotik wird durch den Einsatz von Low-Cost Hardware einem breiten Publikum zugänglich. Bis vor kurzem basierte eine erschwingliche Hardware meist auf Mikrocontrollern mit den entsprechenden Leistungseinschränkungen z.B. im Bereich der Bildverarbeitung. Die Wahrnehmung einer 3D-Umgebung und somit die Möglichkeit zur autonomen Navigation wurde mit relativ kostenintensiver Hardware, z.B. Stereo-Vision-Systemen und Laserscannern gelöst. Die zur Auswertung der Sensorik notwendige Rechenleistung stand - entweder aufgrund des Stromverbrauchs oder der Performance meist für mobile Plattformen (lokal) - nicht zur Verfügung. Durch Einsatz von leistungsfähigen Prozessoren aus dem Bereich der Mobilgeräte (Smartphones, Tablets) und neuartigen Sensoren des Consumer-Bereichs, wie der Kinect, können mobile Roboter kostengünstig für den Einsatz in der Lehre aufgebaut werden.

2 Aufbau mobiler, autonomer Systeme

Die meisten mobilen, autonomen Systeme (Bild 1) bestehen aus den Komponenten:

- CPU,
- Sensorik,
- Aktuatorik,
- Schnittstellen,
- Betriebssystem und
- Stromversorgung.

Des Weiteren spielt der kinematische Aufbau des Roboters für die Steuerung der Aktuatorik und für die Navigation eine wesentliche Rolle, da hier z.B. die Randbedingungen für die Odometry einfließen. Nachfolgend werden verschiedene Alternativen für die einzelnen Komponenten beschrieben und die Vor- und Nachteile für die mobile Robotik erläutert und bewertet.

2.1 CPU

Die zentrale Rechen- und Steuereinheit des mobilen Systems muss die Sensordaten hinsichtlich Wahrnehmung und Lokalisierung auswerten [1]. Weiterhin sind Pfadplanung und Steuerung der Aktuatorik Aufgaben der CPU. Diese können zwar mit zusätzlicher Hardware, wie z.B. Servocontrollern gekoppelt werden, doch müssen in jedem Fall – auch bei dezentraler Architektur – die Daten zentral verwaltet und Steuerbefehle generiert werden.
Die Rechenleistung aktueller *Netbooks*, z.B. aus der Eee PC Reihe der Asus 1215N, ist für viele Algorithmen der mobilen Robotik, wie einfache Bildverarbeitung, SLAM [2] und Lokalisie-

[1] Prof. Dr.-Ing. Stephan Kallweit, FH Aachen/Lehrgebiet Mess- und Automatisierungstechnik, Goethestr. 1, 52064 Aachen

rungsverfahren (AMCL [5]) ausreichend. Weiterhin sind Netbooks auf einen stromsparenden, mobilen Betrieb optimiert und besitzen eine Vielzahl von Standard-Schnittstellen, wie USB, Ethernet, WLAN und Bluetooth. Als OS werden meist verschiedene Linux Distributionen eingesetzt, da die meisten „Betriebssysteme“ (Frameworks) für mobile Roboter unter Linux entwickelt werden (siehe 2.4).

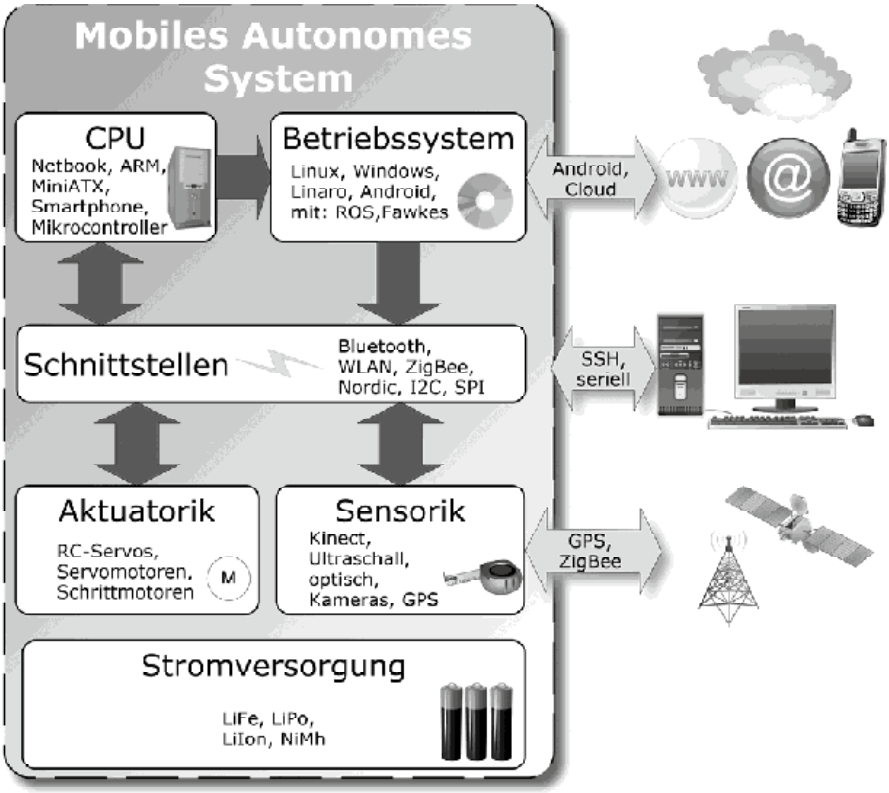

Bild 1: Komponenten mobiler, autonomer Systeme

Leider ist das Powermanagement von einigen Linux Distributionen wie z.B. Ubuntu nicht so effizient wie bei Windows7, so dass kürzere Akku-Laufzeiten des Netbooks die Folge sind. Ein Nachteil bei Netbooks ist das Fehlen von analogen und digitalen I/Os. Diese müssen durch geeignete externe Hardware wie z.B. analoge oder digitale I/O- oder Mikrocontrollerboards zur Verfügung gestellt werden, um die Sensorik einzubinden.

Eine kostengünstige Alternative zu Netbooks bieten *Single-Board-Computer* (SBC), die meist auf preisgünstiger Mobilgeräte-Hardware, wie z.B. ARM-Prozessoren basieren. Es existieren verschiedene „Open-Source“ Projekte, z.B. das Beagle- oder Pandaboard, die beide mit der Texas Instruments OMAP Plattform aufgebaut sind [3].
Das *Pandaboard* ist ein SBC, der alle wesentlichen Elemente eines Standard Netbooks, bis auf Display und Tastatur, besitzt. Es ist äußerst kompakt auf Basis des OMAP4430 bzw. 4460 - je nach Revision - aufgebaut, der zwei ARM-Prozessorkerne bereitstellt. Sämtliche Standardschnittstellen sind vorhanden und auch ein Expansion Stecker, der weitere I/O Leitungen beherbergt. Beim Pandaboard stehen GPIOs bzw. SPI Anbindungen für entsprechende Sensoren zur Verfügung, doch scheint zur Nutzung von SPI eine Neukompilierung des Linux Kernels nötig. Seit kurzer Zeit ist auch ein Expansion Board für das Pandaboard verfügbar, welches einen Kompass- und Beschleunigungssensor, ein Gyroskop sowie einen Touchscreen und ein I2C-Interface besitzt. Auf dem Pandaboard können momentan die OS Versionen von Android, Linux Minimal (Angstrom), Linaro und Ubuntu installiert werden. Eine Installation von Windows CE ist ebenfalls möglich. Empfehlenswert ist die Installation von Ubuntu zur größtmöglichen Kompatibilität mit ROS [5].

	Pandaboard	**Netbook**	**Arduino**	**Smartphone**	**Mini ATX PC**
Rechenleistung	+	++	-	o	++
Betriebssystem	++	++	-	+	++
Sensoranbindung	o	-	++	-	-
Installation	o	++	+	o	++
Schnittstellen	++	++	+	+	+
Erweiterbarkeit	+	+	o	-	++
Stromaufnahme	+	o	++	+	--
Kosten	++	o	++	o	--

Tabelle 1: Vergleich verschiedener CPU

Um eine einfache Anbindung an die Sensorik zu realisieren, können *Mikrocontrollerboards* benutzt werden. Diverse Evaluationboards der Hersteller von Mikrocontrollern stehen kostengünstig zur Verfügung und können aufgrund ihrer diversen Schnittstellen eingesetzt werden. Leider werden die Boards i.A. nur kurzzeitig angeboten, da sie meist mit der nächsten Generation von Controllern verschwinden.
Ein längerfristiger Standard ist mit den *Arduino-Boards* entstanden [4], die es in den verschiedensten Ausführungen gibt. Großer Vorteil der Arduino Systematik ist die einfache Programmierbarkeit im System, so dass keine externen Programmiertools oder weitere Hardware, wie z.B. JTAG-Debugger benötigt werden. Zusätzlich werden die Arduino-Boards bereits vom *Robot Operating System* (ROS) [5] mit einem eigenen Stack (rosserial_arduino) unterstützt, der die gesamte serielle Kommunikation in die Systematik von ROS einbindet.
Nachdem ROS seit kurzer Zeit auch *Smartphones* mit *Android* unterstützt, steht eine weitere kostengünstige Hardware für die mobile Robotik zur Verfügung, die bereits mit den essentiellen Sensoren ausgerüstet ist. Das Problem der Ansteuerung der Aktuatorik bzw. der Anbindung an weitere externe Sensorik kann mittels Googles „Open Accessory API“ [6] gelöst werden: Hier wird ein Arduino-Board mit einem Android Smartphone z.B. über die USB Schnittstelle erweitert. Da ein Smartphone bereits viele Daten auslagert, um lokal Speicherplatz zu

sparen, ist auch eine Art Cloud-Robotik möglich: Bildverarbeitungen, die für den lokalen Prozessor zu rechenintensiv sind, werden ausgelagert und nur die Ergebnisse zurückgesendet. Große Datenbanken (z.B. Gebäudegrundrisse) könnten zur autonomen Navigation durchsucht werden und die Lokalisierung des Roboters erleichtern.

2.2 Sensorik

Seit der Einführung des *Kinect-Sensors* [7] steht eine preisgünstige, ausreichend präzise und universell einsetzbare Tiefenkamera zur Verfügung, die zusätzlich über die Möglichkeit der Gestenerkennung verfügt. Verschiedene Projekte der mobilen Robotik nutzen bereits die Kinect zur Lokalisierung und Kartenerstellung, zum Tracking von Objekten und zur autonomen Navigation, wie z.B. das Projekt TurtleBot [8].
Die Kinect basiert auf einer Entwicklung von PrimeSense, die für Microsoft diesen Sensor für die Xbox360 Spielekonsole entwickelt hat. Die Kinect erschien Ende 2010 und vereinigt eine RGB Farbkamera, eine Infrarot-Projektionseinheit und einen Infrarot-Sensor (IR-CMOS-Kamera) [9], sowie ein Mikrofonarray bestehend aus vier Mikrofonen. Die Kinect erzeugt eine 3D-Punktewolke in QVGA Auflösung (320x240) mit maximal 30 FPS im Bereich von ~0,7 bis 6m mittels strukturierter Beleuchtung. Die optimale Performance wird in einer Entfernung von ~2,3m erreicht - da wo sich im Konsolenbetrieb die Spieler befinden. Zusätzlich befindet sich ein Motor zum vertikalen Ausrichten (±28°) des Kinect-Sensors im Standfuß.
Für die mobile Robotik wird meist der Tiefensensor benutzt. Die generierten 3D-Punktewolken können mittels der Point-Cloud-Library (PCL) [10] vielfältig verarbeitet werden. So können Punktewolken z.B. nach der z-Koordinate gefiltert werden, um Daten jenseits einer bestimmten Entfernung vom Sensor auszublenden (Filtering). Punktewolken aus verschiedenen Aufnahmepositionen können miteinander kombiniert werden, um eine möglichst vollständige 3D Ansicht der Umgebung zu erhalten (Registration). Auch ein Template-Fit ist möglich, um aus der Punktewolke bekannte, parametrisierte 3D-Geometrien herauszufiltern um z.B. Greifpunkte zu extrahieren.
Zusätzlich bietet das *Open-NI* Framework (Open Natural Interaction) [11] eine Spieler- sowie Gestenerkennung, die es ermöglicht die Koordinaten und Winkel der einzelnen Gelenke eines „Spielers" zu erfassen. Damit sind komplexe Gestensteuerungen möglich, z.B. das Drücken eines imaginären Knopfes im 3D Raum, das Bewegen eines „Sliders" und das Drehen eines Knopfes. Das „Winken" mit einer Hand dient dabei zur Initialisierung der Gestensteuerung und erlaubt ein einfaches 3D-Tracking der erfassten Hand mit Ausgabe der 3D-Koordinaten, die für eine Vielzahl von Kontrollaufgaben bereits verwendet werden kann.
Da die Kinect jedoch einen Mindestabstand zur sicheren Detektion von Objekten benötigt, sollte man zusätzlich Standardsensoren, wie die optischen *Abstandssensoren* von Sharp z.B. der GP2Y0A21 und optische *Näherungsschalter* (Cliffsensoren) verwenden. Diese können mit den analogen bzw. digitalen Eingängen eines Arduino verbunden werden. Auch bewährte *Ultraschallsensoren* z.B. der SRF08 können in das Konzept integriert werden, um zusätzlich zu den Daten der Kinect eine sichere Erkennung der Umgebung zu gewährleisten.

2.3 Aktuatorik

Meist werden Modellbauservos (RC-Servos) oder Servomotoren für die Aktuatorik eingesetzt. RC-Servos zeichnen sich durch Ihre einfache Ansteuerung mittels PWM Signalen aus, doch besitzen sie oft kein Positionsfeedback und sind nicht so präzise zu steuern wie Servomotoren.

Als hochwertige Servos im Bereich der mobilen Robotik hat sich die *Dynamixel* Serie etabliert, da sie robust, kostengünstig und komfortabel über Standardschnittstellen (USB, RS485) anzusteuern sind. Die Servos werden über einen Controller im Netzwerk gesteuert, von integrierten PID Reglern geregelt und liefern Positionsmeldungen an den Host zurück. Die Servos der RX-28 und RX-64 Serie werden innerhalb von ROS bereits von einem eigenen „package“ (robotis) unterstützt.

Bei den Servomotoren überzeugen professionelle Lösungen z.B. von *Maxon Motors* durch eine sehr präzise Positions- und Geschwindigkeitsregelung mit definierten Beschleunigungs- und Bremsrampen. Allerdings benötigt jeder Motor ein individuelles Kontrollmodul (EPOS), die untereinander per CAN verbunden werden. Diese Module unterstützen auch bürstenlose, elektrisch kommutierte Elektromotoren mit hohen Encoderauflösungen für anspruchsvolle Aufgaben wie z.B. beim Antrieb von Manipulatoren.

Es existieren zahlreiche Anbieter von kostengünstigen Servo-Motortreibern. Eine sehr kostengünstige Lösung stellt das *MD25 Board* [12] dar. Es wird über eine RS232-TTL-Level oder I2C Schnittstelle betrieben und treibt 12V Gleichstrommotoren mit einem Strom von maximal 2A. Mit geeigneten Getriebemotoren kann ein differentiell angetriebener mobiler Roboter sehr kostengünstig aufgebaut werden.

Als Alternative zum Bau einer differentiellen oder holonomen Plattform können existierende Roboter modifiziert werden, wie z.B. der Staubsauger-Roboter *iRobot Roomba*. Die Roomba 500er Serie kann mittels serieller Schnittstelle (RS232 TTL-Level über Mini-DIN Stecker) mit den Befehlen des iRobot Open Interface (OI) [13] gesteuert werden, wobei zwischen drei Betriebsarten unterschieden wird. Im „Safe Mode“ führt z.B. ein Auslösen der Cliffsensoren zum sofortigen Stop des Roboters – im „Full Mode“ muss der Benutzer selbst über diese Sensoren wachen. Eine Entfernung der Bürsten und des Staubbehälters führt zu einer Gewichtsersparnis und einer längeren Akkulaufzeit. Als Alternative bietet iRobot den *iCreate* an, der bereits als Experimentierplattform konstruiert wurde, d.h. hier fehlt die staubsaugerspezifische Hardware völlig, dafür sind über einen D-Sub-Stecker weitere I/Os ansteuerbar. Der iCreate bildet die Plattform für den TurtleBot und ist für eigene Projekte sehr gut geeignet - jedoch leider in Deutschland nicht lieferbar.

2.4 Betriebssystem

Es existieren eine Vielzahl von Frameworks für mobile Roboter, z.B. das *Robot Operating System (ROS)* [5], *Fawkes* [14], *RoboFrame* [15], *c't-Bot-Framework* [16] und viele weitere.

Für ROS sind die verschiedenen Ubuntu Versionen empfehlenswert, da ROS direkt als Debian „package“ installierbar ist.

Leider müssen einige ROS Stacks bei der Verwendung von ARM-Prozessoren, wie z.B. auf dem Pandaboard, neu kompiliert werden, da keine Binär-Installationen, wie für die Standardplattformen (Intel, AMD), zur Verfügung stehen.

Bei Verwendung eines Mikrocontrollersystems als CPU kann meist kein Standard OS eingesetzt werden, so dass die Programmierung auf dem Host erfolgt und per Download auf die Hardware zugegriffen wird. Ein Nachteil dieser Lösung ist meist die schlechte Portierbarkeit und somit die kurze Lebensdauer der entwickelten Software.
Der Einsatz eines Standard-OS bei der mobilen Robotik hat den großen Vorteil, dass sämtliche Dienste und Werkzeuge, wie z.B. SSH, Python, Java und C/C++ Compiler auf dem Roboter zur Verfügung stehen und sich der Benutzer in einer vertrauten Umgebung befindet. Die Einarbeitungszeit ist daher minimal und es kann direkt auf der Plattform entwickelt werden – ein Cross-Compiler entfällt.

2.5 Stromversorgung

Fortschritte der elektrischen Speichertechnologie sind durch Verwendung von *Lithium-Polymer-* oder *Lithium-Eisen-Phosphat-*Akkus (LiFe) – aus dem RC-Flugbereich – nutzbar, die eine deutlich höhere spezifische Energiedichte besitzen, als die noch vor einigen Jahren üblichen NiCd- oder NiMH-Zellen. Es muss jedoch auf einen Tiefentladeschutz der LiPo-, bzw. LiFe-Akkus geachtet werden. Durch Monitoring der einzelnen Zellen des Akkus unter Benutzung des integrierten Balancerausgangs mittels eines Mikrocontrollers kann ein Tiefentladeschutz kostengünstig gelöst werden. Kommt es nicht auf die letzten Gramm Gewichtsersparnis an – wie z.B. bei fliegenden Plattformen – sollten eher LiFe-Zellen verwendet werden, da sie sowohl bei der Ladung als auch bei der Entladung robuster und einfacher in der Handhabung sind. Daher haben sie zumeist eine längere Lebensdauer.

2.6 Schnittstellen

Eine Unterscheidung zwischen *internen* und *externen Schnittstellen* macht beim Aufbau eines mobilen Systems Sinn. So werden interne Schnittstellen wie bspw. I2C, SPI oder USB für die Kommunikation mit der Sensorik innerhalb des mobilen Systems verwendet. Externe Schnittstellen wie WLAN, Bluetooth und ZigBee stellen die Verbindung zum Host bzw. ROS-Master her. Die Übergänge sind allerdings fließend, da z.B. ein GPS Modul für die Outdoor-Navigation eine externe unidirektionale Schnittstelle zum Satelliten besitzt, aber als Sensor im mobilen System benutzt wird.
Schnittstellen, wie z.B. das *nRF24L01-Modul* von Nordic [17], welches eine drahtlose Verbindung zwischen Mikrocontrollern erlaubt, sind schwer in die Systematik einzuordnen, können aber beim Design von mobilen Systemen sehr hilfreich sein. Die Stromaufnahme beträgt beim nRF24L01 im Sendebetrieb unter 12mA und im Standby-Betrieb ~20µA, womit eine sehr stromsparende drahtlose Kommunikation zwischen intelligenten Sensoren möglich ist.
Bei der Verwendung eines Smartphone als CPU ist oft eine relativ einfache Anbindung über *Bluetooth* möglich, da keine Gerätetreiber installiert werden müssen und USB Schnittstellen meist knapp sind.

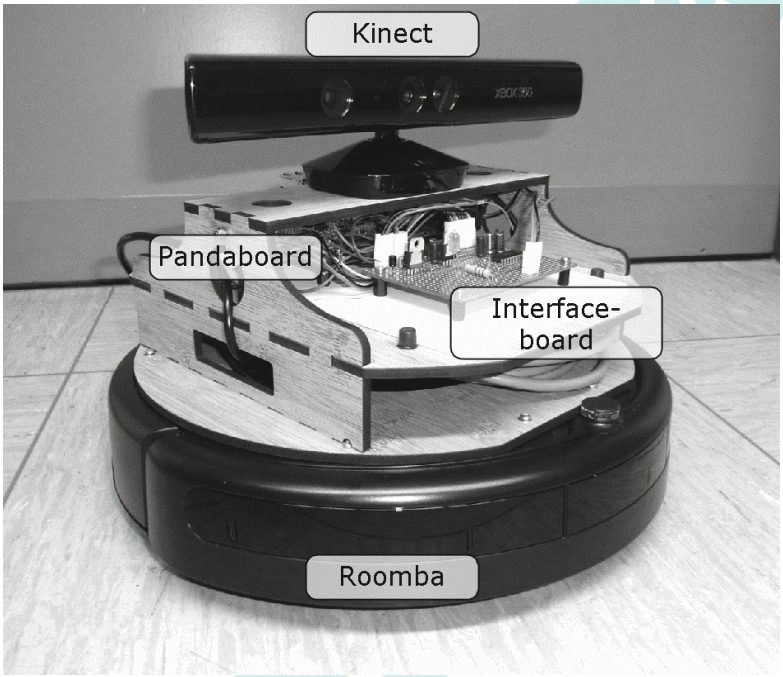

Bild 2: Roomba mit Kinect, Pandaboard und Interfaceboard, Kosten ~700,- Euro

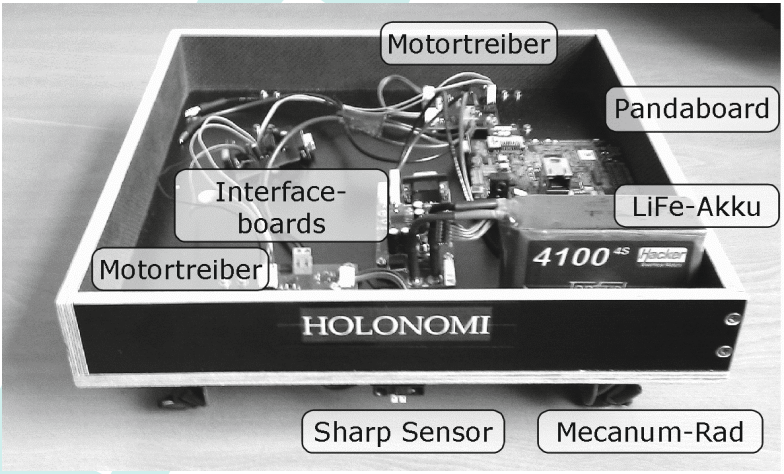

Bild 3: Holonome Plattform mit Pandaboard und Motortreibern, Kosten ~700,- Euro

3 Projekte

Mit den erwähnten Komponenten lassen sich zahlreiche interessante Projekte im Bereich der mobilen Robotik aufbauen. Die Kombination eines *Roomba 555* mit einem *Pandaboard*, einer *Kinect* und einem entsprechendem *Interface* (Bild 2), welches die RS232 Pegel auf TTL Level umsetzt und die Spannungsversorgungen generiert, ist eine einfache Variante des TurtleBot und kann als universelle Plattform für viele Zwecke eingesetzt werden. Falls die Installation von ROS und OpenNI-Treibern auf dem Pandaboard als Quellcode zu kompliziert ist, kann auch das Asus 1215N eingesetzt werden, welches unter Ubuntu 11.04 problemlos mit ROS Electric und den Kinect-Treibern funktioniert.
Als Alternative zu den differentiellen Plattformen, kann man mittels *Pandaboard*, zwei *MD25 Motortreibern*, vier Servomotoren, vier Mecanum-Rädern, einigen Sharp Abstandssensoren und einem LiFe-Akku eine holonome Plattform aufbauen, die ~10kg Zuladung erlaubt (Bild 3). Hier können Kamerasysteme, Manipulatoren oder auch Laserscanner montiert werden. Durch die relativ große Akkukapazität (13,2V/4100mAh) steht eine ausreichende Stromversorgung für weitere Hardware zur Verfügung.
Eine Erweiterung des *TurtleBot* mit einem *Manipulator* zeigt Bild 4. Hier dienen vier Servomotoren als Antrieb für einen, nach dem SCARA Prinzip aufgebauten, Roboterarm, der ca. 500g bewegen kann. Die Einzelteile basieren größtenteils auf Normteilen. Die zu fertigenden mechanischen Baugruppen für den Ober- und Unterarm werden lasergeschnitten. Der Greifer wird aus drei FinGripper Fingern von Festo aufgebaut [18].

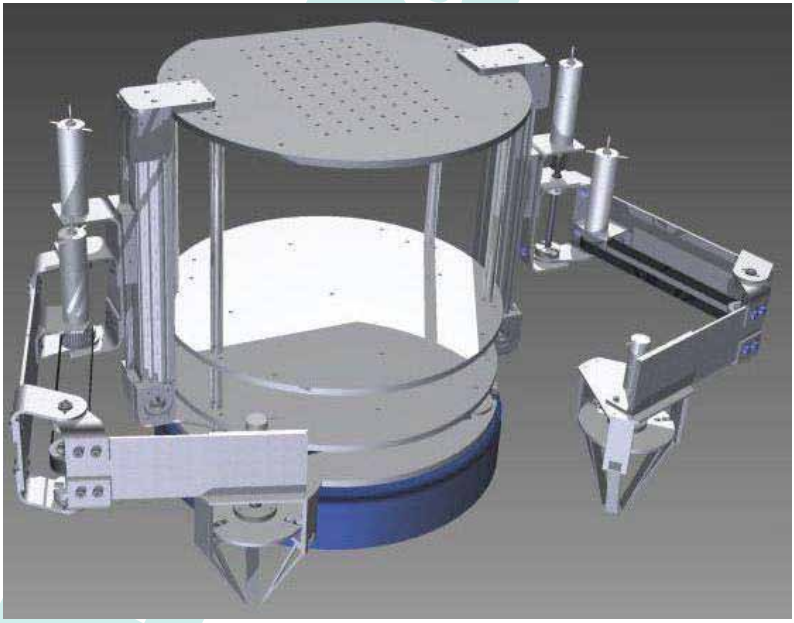

Bild 4:TurtleBot mit Manipulator

4 Fazit

Die Auswahl eines geeigneten Projektes der mobilen Robotik zu Lehrzwecken kann unter didaktischen Aspekten festgelegt werden: Sollen Prinzipien aus den Bereichen der Sensorik und Aktuatorik vermittelt werden, so sind einfache, differentielle Roboterplattformen mit Mikrocontrolleranbindung (Arduino), Pandaboard/Netbook und einfache Motorcontroller eine ideale Lernumgebung für *mechatronische* Grundlagen. Zusätzlich wird die hardwarenahe Programmierung von „*embedded systems*“ an einem praxisrelevanten Beispiel erlernt.
Liegt der Schwerpunkt auf der Verarbeitung der Sensordaten, der kognitiven Erfassung der Umgebung bzw. im Bereich der *Künstlichen Intelligenz*, so ist eine vorgefertigte Plattform, wie der TurtleBot, geeignet. Als „plug-and-play“ Lösung kann der TurtleBot für die Grundlagen der mobilen Robotik aber auch für eine Vielzahl anspruchsvoller Aufgaben eingesetzt werden. Die Hardware ist dabei selbst für Hochschulen in größeren Stückzahlen erschwinglich und die notwendige Software zum Betrieb des Roboters (ROS) hat sich als Standard etabliert. Im industriellen Umfeld existieren ähnliche Vorhaben, so dass ROS auch dort bereits eingesetzt wird.
Eine Erweiterung des TurtleBot mit einem einfachen Manipulator eröffnet einen breiteren Anwendungsbereich und kann mittels der vorgestellten Hardware realisiert werden. Der wesentliche Teil der notwendigen Software existiert ebenfalls in Form der Point-Cloud-Library (PCL) und wird stetig weiterentwickelt.
So kann das gesamte Spektrum der mobilen Robotik von der Wahrnehmung, über die Navigation, bis hin zur Manipulation von Objekten mit kostengünstiger Hardware und Open-Source-Software in der Lehre präsentiert werden.

5 Literatur

[1] Siegwart, R.; Nourbakhsh, I.: Introduction to Autonomous Mobile Robots, MIT Press, 2004
[2] Engelhard, N.; Endres, F.; Hess, J.; Sturm, J.; Burgard, W.: Real-time 3D visual SLAM with a hand-held RGB-D camera, Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Vasteras, Sweden, 2011
[3] pandaboard.org: Board References, 2012
[4] www.arduino.cc: Hardware, 2012
[5] www.ros.org: Robot Operating System ROS, 2012
[6] Bachfeld, D.: Schaltzentrale, Arduino mit Android-Smartphones koppeln, c't Heft 25, 2011
[7] Tajeddini, D.: Minority Report im Fernsehsessel, c't Magazin für Computer Technik, 11/2011
[8] Willow Garage: TurtleBot Open Source Hardware, http://www.willowgarage.com/ turtlebot
[9] http://www.ifixit.com/Teardown/Microsoft-Kinect-Teardown/4066/2
[10] Bogdan Rusu, R., Cousins, S.: 3D is here: Point Cloud Library (PCL), Willow Garage, 68 Willow Rd., Menlo Park, CA 94025, USA
[11] www.openni.org: Open NI Framework, 2011
[12] Devantech Ltd: Robot Electronics, MD25, http://www.robot-electronics.co.uk/htm/md25tech.htm
[13] iRobot.com: Roomba 500 Open Interface (OI) Specification, 2009
[14] Niemueller, T.; Ferrein, A.; Beck, D.; Lakemayer, G.: Design Principles of the Component-Based Robot Software Framework Fawkes, Second International Conference on Simulation, Modeling, and Programming for Autonomous Robots, Darmstadt, Germany, 2010
[15] Petters, S.; Thomas, D.: RoboFrame – Softwareframework für mobile autonome Robotersysteme, TU Darmstadt Diplomarbeit, 2005
[16] Benz, B.: c't-Bot: Roboter selbst bauen, c't Hacks, c't Spezialausgabe, 1/2012
[17] Nordic Semiconductors: nRF24L01Single Chip 2.4GHz Transceiver Product Specification, 2007
[18] Fischer, M.; Kaminski, R.; Mangler, C.; Neuhoff, U.: BionicTripod mit FinGripper, www.festo.com/net/de_corp/SupportPortal/Downloads/146924, 2011