Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Smart Vacuum Cleaner

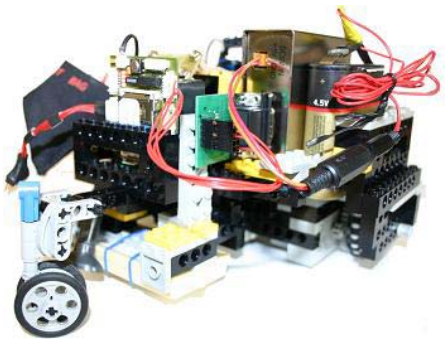
Bau eines autonomen, positionsbewussten Putz-Roboters mit Hilfe der Hitachi µChips

Laborarbeit von Matthias Sala & Julio Pérez
{ salam, jperez } @ student.ethz.ch

Betreuerin: Svetlana Domnitcheva
Professor: Dr. Friedemann Mattern

Institut für Pervasive Computing, ETH Zürich

Oktober 2003 - Februar 2004



Zusammenfassung

Den Traum vom Haushaltsroboter gibt es schon lange. Bisher konnte er nicht verwirklicht werden. Doch der heutige technische Stand ermöglicht den Einsatz ausgeklügelter Technologie in kleinsten Geräten. Um der Verwirklichung dieses Traumes ein wenig näher zu kommen, wurden in den letzten Jahren einige marktreife „intelligente“ Staubsauger präsentiert. Diese sind mit Tast- oder Ultraschallsensoren ausgestattet und verwenden einfache Algorithmen zur Steuerung.

Im Rahmen dieser Arbeit ist ein Prototyp eines smarten Staubsaugers konstruiert worden, welcher für die Positionsbestimmung von der RFID-Technologie Gebrauch macht. Als Rechenplattform dient der an der ETH Zürich entwickelte BTnode, die eigentliche Entwicklungsplattform bildet das LEGO Mindstorms System. Das Ziel war es, den Staubsauger so zu entwickeln, dass er einen mit RFID-Tags belegten Untergrund möglichst autonom, effizient und vollständig reinigen kann. Ausserdem sollte abgeklärt werden, inwiefern sich die RFID-Technologie dafür eignet und welcher Zusatznutzen daraus folgt.

Inhaltsverzeichnis

Liste der Abbildungen

Kapitel 1

Konzept

In diesem Kapitel soll übersichtshalber die Grobstruktur des entwickelten Prototyps erläutert werden. Dafür wird kurz auf die einzelnen Komponenten eingegangen und die Beziehungen zwischen diesen erklärt.

Der Prototyp ist ein batteriebetriebener, mobiler und autonomer Staubsauger, ausgerüstet mit zwei Motoren für den Antrieb der Räder, einer µChip-Antenne zur Erkennung von RFID-Tags und einer Saugvorrichtung für die Reinigung. Im Gehäuse befinden sich die Batterie für den µChip-Reader und die Elektronik zur Steuerung, welche wiederum aus drei Komponenten besteht: Die eine Komponente bildet der RFID-Reader, eine weitere der sogenannte BTnode. Das Herzstück des gesamten Fahrwerks bildet die LEGO Mindstorms Steuereinheit RCX.

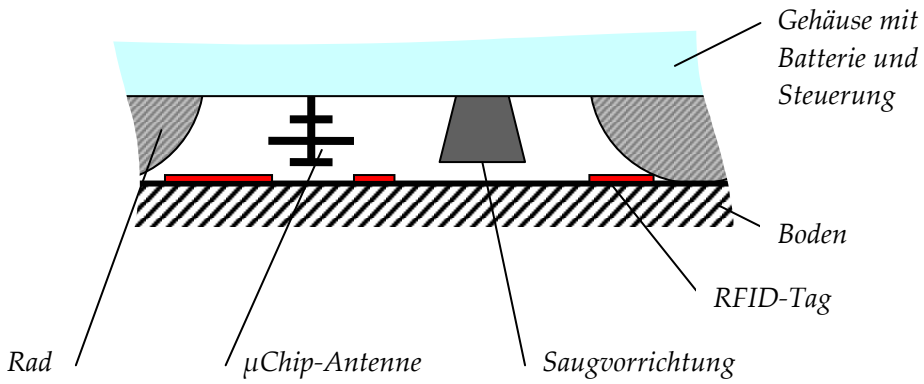


Abbildung 1.1: Aufbau des Prototyps

1.1 Hardware

1.1.1 Der Fahrzeuguntersatz

Der Fahrzeuguntersatz ist aus LEGO-Steinen aufgebaut. Die beiden Hinterräder sind jeweils mit einem Motor verbunden, welcher vom RCX gespiesen wird. Das Vorderrad ist frei drehbar, unangetrieben und ist für die Lenkung verantwortlich.

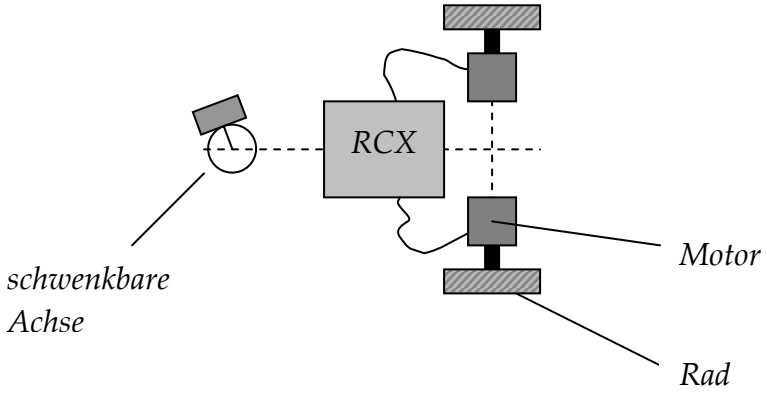


Abbildung 1.2: Anordnung der Achsen und Räder

1.1.2 Kommunikationsverbindungen zwischen den Komponenten

Die µChip-Antenne ist über ein mit der µChip-Ausrüstung mitgeliefertes Kabel an den µChip-Reader angeschlossen. Der Reader reicht die von der Antenne erhaltenen Daten über eine serielle Schnittstelle an den BTnode weiter. Letzterer interpretiert daraufhin die Daten und schickt via Infrarot Steuersignale an den RCX. Der BTnode hat zusätzlich ein integriertes Bluetooth-Modul, womit er mit den in der Umgebung befindlichen Bluetooth-Geräten kommunizieren kann. Abbildung 1.3 veranschaulicht diesen Aufbau.

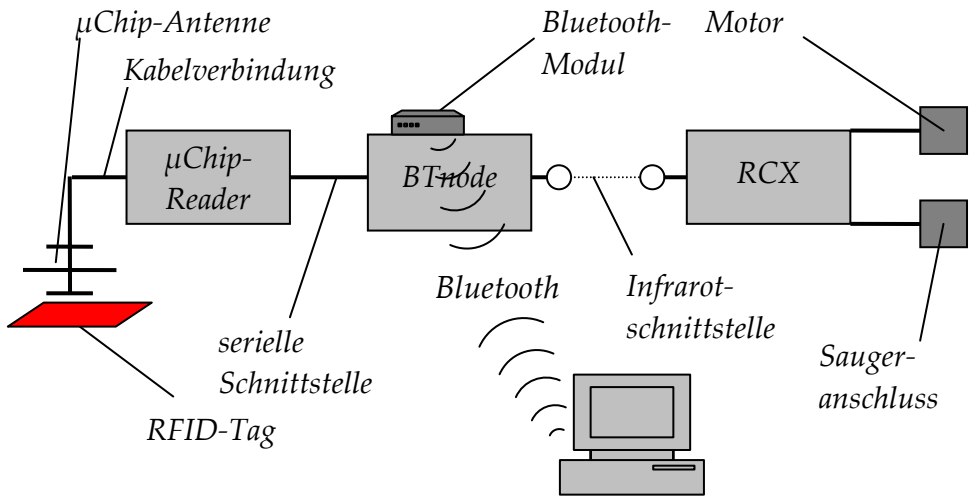


Abbildung 1.3: Schema der digitalen und elektronischen Komponenten

1.2 Software

Weil der Prototyp aus mehreren rechnenden Modulen besteht, umfasst das Projekt mehrere Programmteile auf den jeweiligen Komponenten:

1. Programm auf dem BTnode
2. Programm auf dem RCX
3. Programm auf einem aussenstehenden PC

1.2.1 Relation zwischen den Programmteilen

Die Anwendung auf dem aussenstehenden PC kann mittels eines einfachen Protokolls den Ablauf des Programmes auf dem BTnode konfigurieren. Es sollte hier ausdrücklich erwähnt werden, dass die Anwendung auf dem PC nichts mit dem eingentlichen „Putzalgorithmus“ zu tun hat. Das Programm auf dem BTnode übermittelt demjenigen auf dem RCX Kommandos. Zusätzlich verarbeitet der BTnode die Informationen, welche er vom RFID-Reader bekommt. Der RCX hat die Aufgabe, die Motoren und den Sauger gemäss erhaltenen Kommandos zu steuern.

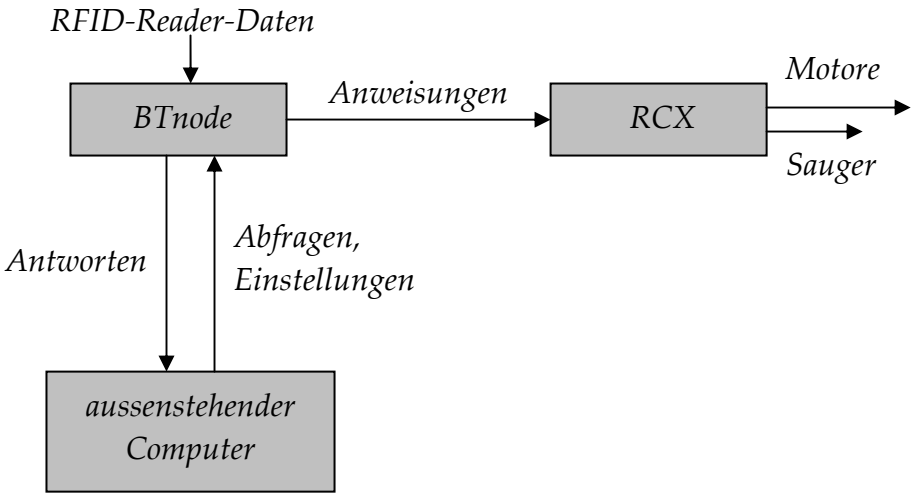


Abbildung 1.4: Beziehung zwichen den einzelnen Programmteilen

1.2.2 Verwendete Programmiersprachen

Die Programmierung des BTnode erfolgt mit der C-Programmiersprache und der Systemsoftware des BTnode-Projektes. Der RCX kann mit Not Quite C, einer C-ähnlichen Sprache, programmiert werden. Für die Applikation auf dem aussenstehenden Computer können prinzipiell alle denkbaren Konzepte verwendet werden. In dieser Arbeit wurde dafür wiederum die gleiche Programmiersprache und Software verwendet, wie für den BTnode.

Kapitel 2: μChip

Kapitel 2

µChip

Nachdem erwähnt wurde, aus welchen Teilen der Prototyp besteht, soll nun auf die einzelnen Komponenten genauer eingegangen werden. Dieses Kapitel beschäftigt sich mit der verwendeten µChip-Ausrüstung.

2.1 Technologie

Die Hitachi µChip-Technologie bietet die Möglichkeit, RFID-Tags (RFID = Radio Frequency Identification) mit einer festen Identifikationsnummer über eine Antenne einzulesen. Dafür erzeugt der Reader ein elektromagnetisches Feld, welches für die induzierte Spannung in der Spule des RFID-Tags verantwortlich ist. Dadurch wird ermöglicht, dass der Tag dem Reader seine Identifikationsnummer übermitteln kann (siehe dazu Abbildung 2.2).

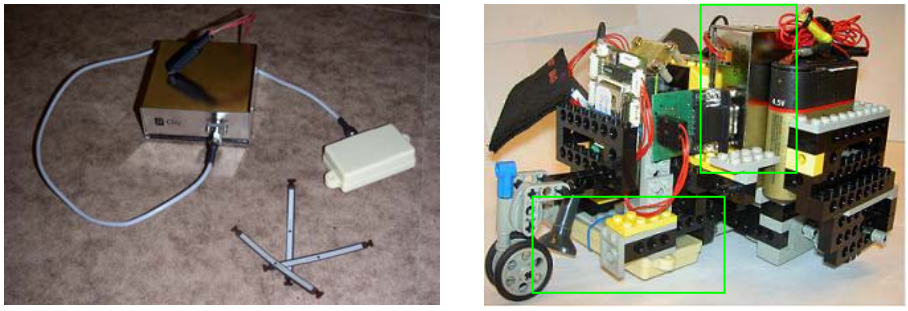


Abbildung 2.1: Hitachi µChip-Reader, -Antenne und -RFID-Tags

Allerdings muss dabei beachtet werden, ob die von der Antenne ausgesendeten Radiowellen linear oder zirkular polarisiert sind. Bei einer linear polarisierten Welle ist die Schwingungsrichtung stets fix (senkrecht zur Ausbreitungsrichtung), bei einer zirkular polarisierten Welle hingegen ändert sich die Bewegungsrichtung in regelmässigen Abständen. Für den in dieser Arbeit erstellten Prototypen war es wichtig, eine kleine Antenne zu benutzen, die an der Unterseite des Fahrzeugs Platz findet (siehe Abbildung 2.1 rechts). Der Nachteil

dabei ist, dass die verwendete Antenne linear polarisierte Wellen ausstrahlt, wodurch ein Tag, der senkrecht zur Antenne liegt, nicht erkannt wird.

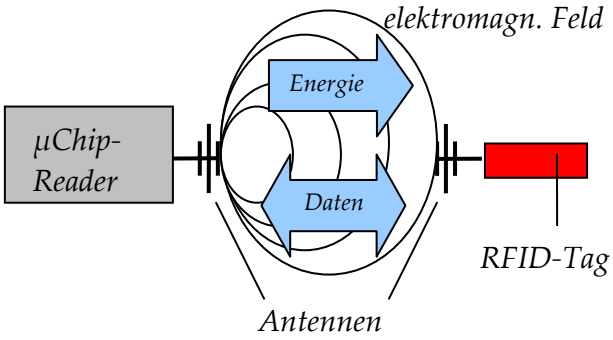


Abbildung 2.2: Funktionsweise der RFID-Technologie

Ein weiteres Problem stellt sich, wenn Tags zu nahe beieinander liegen. Dann tritt der Fall ein, dass mehrere Tags ins erzeugte Magnetfeld gelangen und die Signale vom Reader empfangen. Entsprechend werden mehrere Tags ihre Identifikationsnummer senden. Der Reader müsste dann eine Kollisionsbehandlung anbieten, was beim verwendeten Hitachi-Reader nicht der Fall ist. Deswegen sollte man darauf achten, dass man die Tags nicht zu nahe beieinander positioniert, um jegliche Kollisionen zu vermeiden.

2.2 Protokoll

Hitachi liefert zum µChip-System eine Dynamic Link Library (DLL), anhand welcher man in Windows-Anwendungen mit dem µChip-Reader über die serielle Schnittstelle kommunizieren kann. Eine Beschreibung des dabei verwendeten Kommunikations-Protokolls fehlt leider. Da aber für dieses Projekt eine Implementation des Protokolls auf dem BTnode notwendig war, wurde versucht, dieses mittels Reverse Engineering aufzuschlüsseln. Es wurde ein Port Sniffer [6] verwendet, um den Datenaustausch zwischen der DLL und dem µChip-Reader abzuhören.

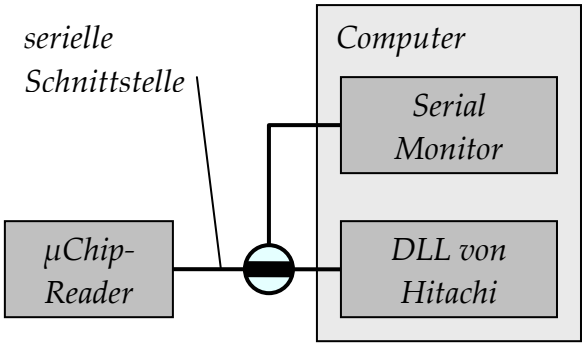


Abbildung 2.3: Anordnung für Serial Monitoring

Das Protokoll funktioniert nach dem Request-Respond-Prinzip. Die Kommunikation läuft mit 34500 baud und verwendet keinen Flowcontrol. Der Dialog beginnt mit der 3 Byte langen Anfrage {0x02, 0x0b, 0x03}, die vom BTnode an den Reader gesendet wird. Befindet sich an der Antenne ein RFID-Tag, antwortet der µChip-Reader mit einem 19 Byte langen Stream, angeführt vom Header {0x02, 0x00}, gefolgt von der 16 Byte langen Identifikationsnummer des RFID-Tags und abgeschlossen mit dem Terminator 0x03. Wird kein RFID-Tag erkannt, antwortet der µChip-Reader stattdessen mit {0x02, 0xff, 0x00, ..., 0x03}, d.h. die ID wird durch Nullen und das letzte Headerbyte durch 0xff ersetzt. Das zweite Byte zeigt also an, ob sich ein RFID-Tag in Reichweite der Antenne befindet: 0x00 wenn dies der Fall ist, 0xff sonst.

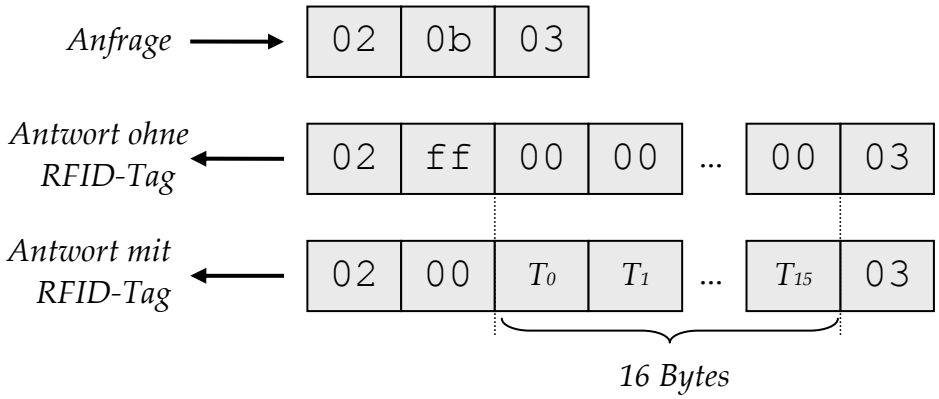


Abbildung 2.4: Protokoll mit Anfrage- und Antwortbytestreams

Für unsere Zwecke reicht diese gegen Fehler nicht abgesicherte Kommunikation aus - stimmen Header und Terminator mit der oben genannten Form überein, wird die Antwort akzeptiert, andernfalls wird sie verworfen. Denkbar wäre aber sicherlich noch die Überprüfung der in der Identifikationsnummer enthaltenen Quersumme, um Übertragungsfehler aufzuspüren.

2.3 Verteilung der RFID-Tags auf dem Untergrund

Gemäss den bis anhin genannten Eigenschaften der µChip-Ausrüstung sind folgende Punkte bei der Verteilung der RFID-Tags auf dem Untergrund zu beachten:

1. Dichte der Verteilung: - dicht
   - locker
2. Anordnung der Tags : - zufällig
   - geordnet

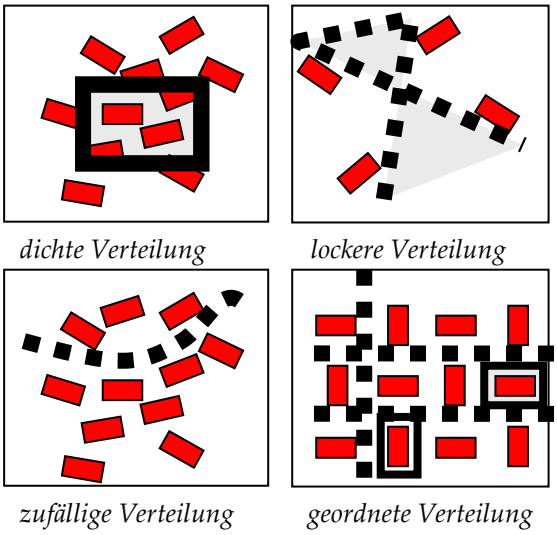


Abbildung 2.5: Mögliche Verteilungen der RFID-Tags

2.3.1 Dicht

Eine dichte Verteilung der RFID-Tags hat den Vorteil, dass man genauer lokalisieren bzw. vollständiger reinigen kann. Andererseits hat es den Nachteil, dass die Antenne kleiner sein muss, damit sich nicht mehrere RFID-Tags zur gleichen Zeit im Feld der Antenne

befinden. Der minimale Abstand zwischen zwei RFID-Tags sollte nie kleiner sein als die Grösse der Antenne.

2.3.2 Locker

Bei einer lockeren Verteilung besteht die Gefahr, dass über eine längere Zeit keine RFID-Tags überfahren werden und die Lokalisierung dadurch verunmöglicht wird. Um dennoch die ganze Fläche reinigen zu können, muss der smarte Staubsauger eine grössere Saugbewegung ausführen, welche die Umgebung um den RFID-Tag abdeckt. Damit all diese Bereiche flächendeckend werden, muss der Prototyp den mittleren Abstand der RFID-Tags kennen. Weiter ist diese Verteilung deshalb problematisch, weil sie viel Energie des RCX verbraucht, da dieser grössere Putzbewegungen ausführen muss. Während dieser Bewegungen müssen jeweils alle Ausgänge mit Strom versorgt werden (zwei Ausgänge für den Antrieb der Räder, einer für den Betrieb der Saugvorrichtung). Die Batterien des RCX sind entsprechend schon nach kurzer Zeit aufgebraucht.

2.3.3 Zufällig

Ein Untergrund mit zufällig darauf verteilten RFID-Tags wäre einfacher in der Herstellung. Es ist auch eher unwahrscheinlich, dass es gerade Routen gibt, welche keine RFID-Tags enthalten. Allerdings kann es vorkommen, dass beim ersten Überfahren eines Tags dieses nicht erkannt wird (wenn der Tag senkrecht zur Antenne steht) und erst bei einem späteren Überfahren detektiert wird (vorausgesetzt die Antenne steht dann nicht wieder rechtwinklig zum Tag).

2.3.4 Geordnet

Bei einer geordneten Verteilung wie in der Abbildung 2.5 unten rechts, existieren regelmässig Pfade, welche keine Tags enthalten. Dafür tritt der Fall, wo Antenne und Tag senkrecht zueinander stehen, nicht ein, bzw. er tritt zwar ein, aber da benachbarte Tags im rechten Winkel zum gerade überfahrenen Tag stehen, werden diese erkannt. Wenn dann die Verteilung noch genügend dicht ist, wird auch die Fläche des nicht erkannten Tags gereinigt.

Falls die RFID-Tags sequentiell angeordnet sind, dh. die Identifikationsnummern auch geordnet sind, könnte sich der Staubsauger zusätzlich orientieren. Es stellt sich aber die Frage, ob das realistisch ist: Werden die Teppiche durchnummeriert produziert, dann muss auf die Nachbarn der verlegten Teppichplatten geachtet werden, damit die ganze Ordnung der Tag-IDs noch stimmt. Oder wenn es sich um veränderbare RFID-Tags handelt (also Tags, die nicht nur read-only sind), müssten sämtliche Tags in der gesamten Wohnung strikt durchnummeriert werden.

Es sind sicherlich auch andere geordnete Verteilungen möglich, die wiederum ihre Vor- und Nachteile haben. Interessierte Leser könnten sich im Rahmen einer Arbeit damit beschäftigen.

Kapitel 3

BTnode

Als Hauptsteuereinheit und -Rechenplattform des Prototyps dient der BTnode. In diesem Kapitel soll diese Komponente des Smart Vacuum Cleaners (SVC) genauer behandelt werden.

3.1 Technologie

Der an der ETH Zürich entwickelte BTnode [2] ist ein programmierbarer Computer in der Grösse einer Zündholzschachtel. Er besitzt einen Microcontroller (CPU) mit einer Taktrate von 8 MHz, und verfügt über verschiedene Speicher (RAM, Flash ROM, EEPROM). Anhand eines eingebauten Bluetooth-Moduls kann er mit aussenstehenden bluetoothfähigen Geräten kommunizieren und bietet zudem 6 Schnittstellen für den Anschluss von Sensoren u.ä. an. Die Programmierung des BTnode erfolgt mittels eines AVR-GCC-Compilers[1] und einem Hardware-Programmierer. Es existiert eine Systemsoftware, welche verschiedene Funktionalitäten bereits anbietet.

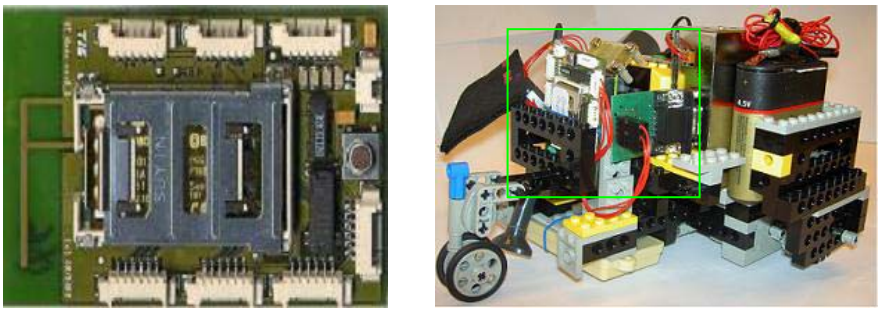


Abbildung 3.1: BTnode

[1] Erhältlich über die BTnode Homepage [2]

3.2 Bluetooth

Der BTnode verfügt über ein standardisiertes Bluetooth-Modul von Ericsson. Die für den Standard notwendigen Algorithmen werden in der Systemsoftware mitgegeben. Für diese Arbeit wurde das L2CAP-Protokoll verwendet, um mit einem anderen Bluetooth-Modul kommunizieren zu können.

3.3 Infrarot

Andres Erni und Stefan Reichmuth haben in einer Semesterarbeit [8] an der ETH Zürich ein Infrarotmodul für den BTnode entwickelt. Über diese Schnittstelle kann der BTnode mit der LEGO Mindstorms Steuereinheit kommunizieren. Es wurde versucht, von dieser Schnittstelle so wenig wie möglich Gebrauch zu machen, da sich die Kommunikation über Infrarot in beide Richtungen als extrem fehleranfällig herausstellte.

3.4 Serielle Schnittstelle

Die UART-Schnittstelle des BTnode kann weitgehend konfiguriert werden, so dass eine Kommunikation mit dem µChip-Reader problemlos möglich ist. Über diese Schnittstelle wird das in Abbildung 2.4 dargestellte Protokoll abgewickelt.

3.5 EEPROM

Um einzelne Informationen auch nach einem Reset des BTnode oder nach dem laden eines neuen Programmes aufrufen zu können, gibt es den sogenannten EEPROM (Electrically Erasable Programmable Read-Only Memory). Es handelt sich dabei um einen permanenten Speicher, der, wie die Festplatte eines herkömmlichen Rechners, seine Daten auch nach dem Ausschalten der Stromzufuhr beibehält. Auf den 4 Kilobyte grossen Speicher kann mittels drei Funktionen (`eeprom_ready()` (bereit für Zugriff), `eeprom_wb(Adresse, Byte)` (Byte in den EEPROM schreiben), `eeprom_rd` (Adresse, Byte) (Byte aus dem EEPROM lesen)) uneingeschränkt zugegriffen werden. In unserem Fall machen wir von diesem Speicher Gebrauch, um die Grenztags der zu reinigenden Fläche permanent abzuspeichern, damit nicht nach jedem Update des BTnode die Grenze wieder eingelesen werden muss.

Kapitel 4: LEGO Mindstorms

Kapitel 4

LEGO Mindstorms

Nachdem das µChip-System und der BTnode besprochen wurden, sollte nun die letzte für den Putzroboter verwendete Komponente beschrieben werden.

4.1 Technologie

Das System von LEGO bietet die Möglichkeit, schnell und unkompliziert einfache, herumfahrende Roboter mit primitiver Logik zu entwickeln. Dazu wird eine Steuereinheit (RCX) zur Verfügung gestellt, welche drei Ein- und drei Ausgänge anbietet. Die Eingänge sind für den Anschluss von Sensoren. Für dieses Projekt war die Verwendung der Eingänge und daran angeschlossener Sonsoren nicht notwendig.

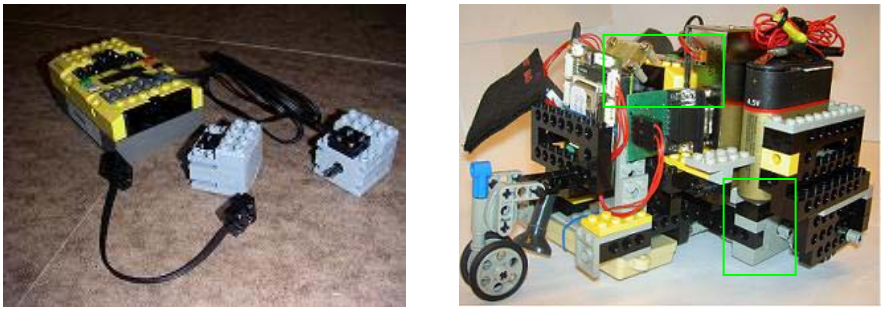


Abbildung 4.1: LEGO Mindstorms Steuereinheit (RCX), Kabelverbindung und Motore

An den Ausgängen werden mittels mitgelieferter Kabel üblicherweise die Motore für den Radantrieb oder eventuell sonstige elektronische Komponenten angeschlossen. Für den Smart Vacuum Cleaner haben wir zwei Ausgänge für den Antrieb der Motore, und ein Ausgang für die Speisung des eigentlichen Staubsaugers verwendet.

Wie schon weiter oben erwähnt, wurde für die Programmierung der RCX-Einheit Not Quite C [1] verwendet, das einfache Schnittstellen für die Steuerung der Ausgänge anbietet. Beispiele

hierfür sind Funktionen, welche die Drehrichtung oder auch den Leistungspegel der angeschlossenen Motore regeln. Weiter werden Schnittstellen für die Kommunikation über den eingebauten Infrarotsender angeboten.

4.2 Infrarot

Die über die Infrarotschnittstelle gesendeten Nachrichten haben eine Länge von einem Byte. Bei anfänglich durchgeführten Tests ist aufgefallen, dass trotz dem Mitsenden der Quersumme und der Komplementärbytes, auffällig viele Übertragungsfehler auftreten. Zur Abhilfe wurde eine eigene Fehlerkorrektur implementiert, welche in Kapitel 5 näher beschrieben wird. Für Details zum Protokoll des LEGO Mindstorms sei auf [8] verwiesen.

4.3 Steuerung fremder Geräte

Der RCX bietet keine direkte Möglichkeit an, fremde Geräte anzusteuern. Da es aber für den Smart Vacuum Cleaner eine Reinigungsvorrichtung anzuschliessen galt, wurde versucht, das Sauggerät über ein mitgeliefertes Kabel (siehe Abbildung 4.1 links unten) direkt an einen freien Ausgang des RCX anzuschliessen und die mit dem Sauggerät mitgelieferten Batterien nicht zu verwenden (d.h. das Gerät sollte nur durch die Stromzufuhr des RCX gespiesen werden). Dabei trat das Problem auf, dass die resultierende Saugstärke zu schwach ausfiel. Deshalb wurde der Batteriebehälter des Sauggeräts doch angeschlossen, wodurch aber ein anderes Problem auftauchte: Sobald einmal der Staubsauger mit Strom versorgt wurde (d.h. der entsprechende RCX-Ausgang aktiviert wurde), konnte der Stromkreis auch nicht durch Ausschalten des Ausgangs unterbrochen werden, die Saugvorrichtung stand also ständig unter Strom. Die Lösung dieses Problems besteht darin, ein Relais zu verwenden, welches über den Ausgang des RCX kontrolliert wird und den Stromkreis des Staubsaugermotors öffnet oder schliesst. Wenn der Ausgang des RCX aktiviert ist, schliesst das Relais den Stromkreis zwischen ihm und dem Staubsauger, ansonsten wird der Stromkreis unterbrochen. Abbildung 4.2 veraunschaulicht diesen Aufbau.

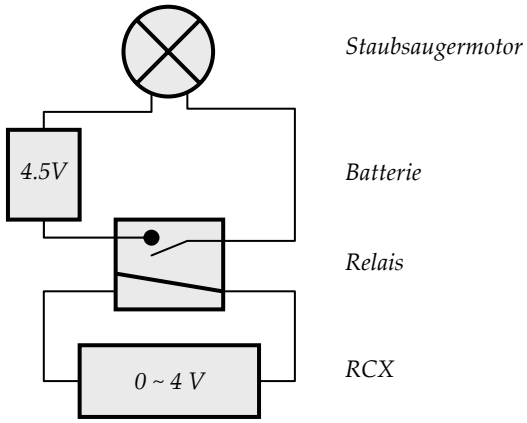


Abbildung 4.2: Schaltplan mit Relais für die Steuerung fremder Geräte

Kapitel 5: Algorithmus

Kapitel 5

Algorithmus

Nachdem alle verwendeten Technologien besprochen wurden, beschäftigt sich dieses Kapitel mit dem implementierten Algorithmus.

5.1 Das Covering Problem

Die Aufgabe des Staubsaugers besteht darin, eine bestimmte Fläche auf kürzestem Weg vollständig abzudecken, so dass überall geputzt und die Fläche nie verlassen wird. In der Literatur wird dies als Covering Problem bezeichnet. Es existieren viele Arbeiten zur Lösung des Problems. Die meisten Ansätze gehen aber davon aus, dass der Cursor bzw. das Fahrzeug komplett positionsbewusst ist und so das Gebiet mit einer Matrix kartographiert. Diese Überlegung lässt sich auf hiesigen Prototypen nur bedingt übertragen, da er die Winkel beim Abbiegen nicht bestimmen kann und deswegen komplett auf eine Kartographie verzichten muss. Der einzige Sensor des Staubsaugers ist die μChip-Antenne. Somit muss man sämtliche Rückschlüsse aus den auf dem Boden verteilten RFID-Tags gewinnen.

Um nicht zusätzliche Tastsensoren verwenden zu müssen, wird eine Grenze um das zu reinigende Gebiet benötigt. Diese muss während einer Markierungsphase abgefahren werden. Alle in dieser Phase eingelesenen RFID-Tags werden später, beim eigentlichen Putzen, als Grenztags erkannt. Dabei ist zu beachten, dass die Tags so nahe beieinander liegen müssen, dass ein „Ausbrechen“ des Fahrzeugs verunmöglicht wird, d.h. an der Grenze immer ein Tag detektiert wird. Andererseit darf die Verteilung nicht zu dicht sein, da sonst nicht-behandelte Kollisionen auftreten.

Weiter muss dafür gesorgt sein, dass die Grenztags so angeordnet sind, dass egal in welchem Winkel der Roboter auf die Grenze stösst, die Grenztags erkannt werden, d.h. nicht senkrecht zur Antenne liegen (da dann der Tag nicht detektiert wird).

Aus diesen Überlegungen folgt, dass eine mögliche Lösung für die Anordnung der Grenztags die in Abbildung 5.1 dargestellte ist.

Es scheint also recht umständlich zu sein, anhand von Tags die Grenze eines Gebietes zu markieren. Folglich eignet sich dafür die RFID-Technologie eher nicht, Tastsensoren scheinen viel geeigneter zu sein.

Abbildung 5.1: Anordnung der Grenztags

5.2 Rückschlüsse

Beim Herumfahren stösst der Staubsauger immer wieder auf RFID-Tags mit einer eindeutigen Identifikationsnummer. Daraus lässt sich direkt ableiten, ob sich das Gerät schon mal an dieser Position befunden hat, ob es sich um einen Grenztag handelt, oder ob es ein neuer, bisher unbekannter RFID-Tag ist. Es werden also drei Klassen von Tags unterschieden:

- Grenztags
- Innere, schmutzige RFID-Tags
- Innere, saubere RFID-Tags

Für jeden Tag wird deshalb gespeichert, ob er ein Teil der Grenze ist, und falls dies nicht zutrifft, ob er „sauber“ ist, also dort wo sich der Tag befindet, schon gereinigt wurde. Ob sich ein RFID-Tag inner- oder ausserhalb des umgrenzten Gebietes befindet, kann der Staubsauger nicht feststellen. Deshalb ist es wichtig, dass er das Gebiet nie verlässt.

Zudem können zu allen RFID-Tags weitere 2 Informationen gespeichert werden, die der Algorithmus berücksichtigt:

- Jeweils zuvor gelesene RFID-Tags
- Drehrichtung nach dem Rückwärtsfahren

Die erste Information, nennen wir sie A, ist folgendermassen zu verstehen: Jedes Mal, wenn irgendein Tag, im Folgenden X genannt,

angetroffen wird, soll der Index des direkt davor vom SVC gelesenen Tags im zu X gehörenden Speicherbereich auf dem BTnode notiert werden (siehe dazu Abbildung 5.2. Im Detail-Bereich sind 10 Bytes dafür vorgesehen, jeder Index ist ein Byte gross). Wozu dies gebraucht wird, soll später erklärt werden. Der zweite Informationseintrag, nennen wir ihn B, ist nur für Grenztags gedacht und wird ebenfalls später genauer erläutert.
Das Verhalten des Roboters hängt also sowohl von den vom μChip-Reader gelieferten Daten wie auch von den gespeicherten Informationen ab.

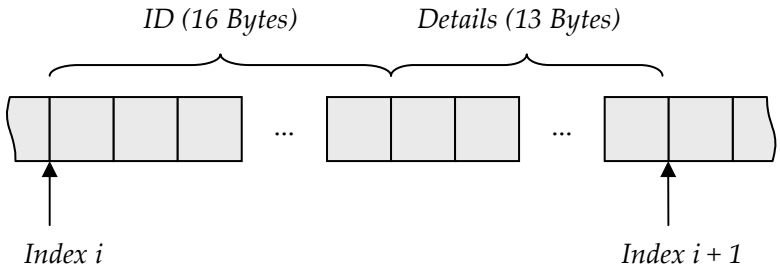


Abbildung 5.2: Datenlayout der RFID-Tags Datenbank auf dem BTnode

5.3 Ereignisse

Jedes Mal, wenn der SVC auf ein Tag stösst, d.h. der Reader ein solches detektiert, muss festgestellt werden, zu welcher Klasse der Tag gehört und es müssen die zu diesem Tag gehörenden Informationen überprüft werden. Anhand dieser Informationen wird bestimmt, welches der in den folgenden Abschnitten beschriebenen Ereignisse eingetreten ist, um anschliessend der Situation entsprechend zu handeln.

Wenn von einem aussenstehenden Gerät eine Bluetooth-Verbindung mit dem BTnode existiert, wird dieses Gerät bei jedem Ereignis vom BTnode eine Nachricht empfangen, die das aufgetretene Ereignis und dessen Behandlung wiedergibt.

5.3.1 Grenztag

Stellt der BTnode fest, dass die vom Reader gelieferte ID zu einem Grenztag gehört, wird dies dem RCX mitgeteilt (Infrarot-Schnittstelle). Der RCX muss anschliessend dafür sorgen, dass die Motore so gesteuert werden, dass der SVC innerhalb des zu

reinigenden Gebietes bleibt. Dafür sollte sicherlich zuerst rückwärts gefahren werden. Daraufhin könnte sich der SVC im Grunde genommen in eine beliebige Richtung drehen. Da aber unter anderem erreicht werden soll, dass die ganze Fläche abgefahren wird, wird von der oben erwähnten Information B Gebrauch gemacht. Diese gibt an, in welche Richtung sich der SVC das letzte Mal, als dieser Grenztag angetroffen wurde, gedreht hat. Entsprechend entscheidet der SVC, in die andere Richtung zu drehen und aktualisiert den Informationseintrag.

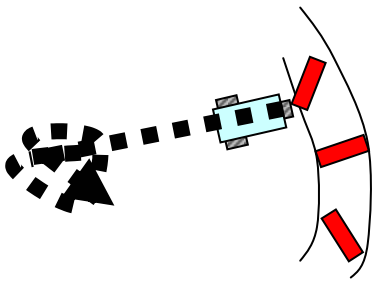


Abbildung 5.3: Verhalten beim Anstossen an die Grenze

5.3.2 Innerer, schmutziger Tag

Es gibt zwei Möglichkeiten, warum der BTnode einen eingelesenen Tag als schmutzig klassifiert: Entweder die Tag-ID ist nicht im Speicher aufzufinden oder die zum Tag gespeicherten Informationen besagen, dass an der Stelle, wo sich der Tag befindet, noch nie gereinigt wurde (warum letzterer Fall eintreten kann, wird gleich im Anschluss erläutert).

Wenn der Roboter einen solchen Tag detektiert, meldet der BTnode dem RCX dieses Ereignis. Das löst bei der Legosteuereinheit das Einschalten der Saugvorrichtung und der Motore aus, so dass eine Zick-Zack-Bewegung ausgeführt wird (siehe Abbildung 5.4). Nachdem die Bewegung bei eingeschaltetem Staubsauger ausgeführt wurde, stellt der RCX den Staubsauger wieder ab und fährt geradeaus weiter. Der BTnode speichert den Tag als gereinigt ab und wartet auf die nächste vom Reader übermittelte ID.

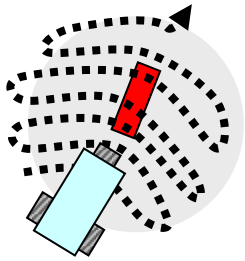


Abbildung 5.4: Verhalten beim Erkennen eines schmutzigen Gebietes

Würde der Algorithmus wie eben beschrieben implementiert, könnte keine Rücksicht auf Übertragungsfehler bei der Infrarot-Kommunikation zwischen dem BTnode und dem RCX genommen werden. Wenn also der RCX die vom BTnode versendete Nachricht nicht oder verfälscht bekommt, wird der RCX nicht zu reinigen beginnen. Der BTnode wüsste aber nichts vom Übertragungsfehler und würde den Tag als sauber abspeichern, die entsprechende Teilfläche würde somit in der Zukunft fälschlicherweise als gereinigt klassifiziert. Um dem entgegenzuwirken, wurde ein Acknowledgement (ACK) eingeführt, welches vom RCX in einer solchen Situation an den BTnode verschickt wird. Erhält der BTnode dieses ACK, weiss er, dass die Fläche gereinigt wurde und kann den Tag als sauber abspeichern.

Nun kann es allerdings sein, dass der Übertragungsfehler bei der Kommunikation vom RCX zum BTnode, also beim Verschicken des ACKs, auftritt. Dann wird die Stelle gereinigt, der BTnode speichert jedoch den Tag als schmutzig ab, da er kein ACK bekommen hat. Dieser Fall ist aber wesentlich günstiger als wenn man kein Acknowledgement benützt und somit riskiert, dass eine Fläche fälschlicherweise als gereinigt abgespeichert wird.

Nach dem ganzen Prozedere überprüft der BTnode noch, ob ein Déjà-vu aufgetreten ist. Was damit gemeint ist, wird in Abschnitt 5.4 erklärt.

5.3.3 Innerer, sauberer Tag

Wenn vom BTnode erkannt wird, dass die Stelle, wo sich der Tag mit der vom Reader gelieferten ID befindet, schon einmal in der Vergangenheit gereiningt wurde (dies geschieht anhand der zum

Tag gespeicherten Informationen), gibt es nichts zu tun, ausser wiederum zu überprüfen, ob ein Déjà-vu eingetreten ist.

5.4 *Déjà-vu*

Da eines der Ziele des SVC darin besteht, möglichst die ganze zu reinigende Fläche abzudecken und auch zu vermeiden, mehrmals die gleichen Strecken abzufahren, wurde ein sogenanntes Déjà-vu-Ereignis eingeführt. Dieses ist folgendermassen definiert: Sei Y der vorletzte und X der zuletzt vom SVC angetroffene Tag. Anhand der in Abschnitt 5.2 als A bezeichneten Information kann der BTnode überprüfen, ob man in der Vergangenheit den Tags Y und X schon einmal in dieser Reihenfolge begegnet ist (dazu muss der BTnode lediglich feststellen, ob der Index des Tags Y im Detail-Speicherbereich des Tags X vorkommt). Wenn dies der Fall ist, sprechen wir von einem Déjà-vu. In einer solchen Situation sollte vermieden werden, geradeaus zu fahren, da diese Richtung in der Vergangenheit schon eingeschlagen wurde, nämlich beim ersten Befahren dieser Strecke. Deswegen sendet der BTnode dem RCX eine entsprechende Nachricht, so dass letzterer dafür sorgen kann, in eine andere Richtung zu drehen. Abbildung 5.5 veranschaulicht diesen Vorgang.

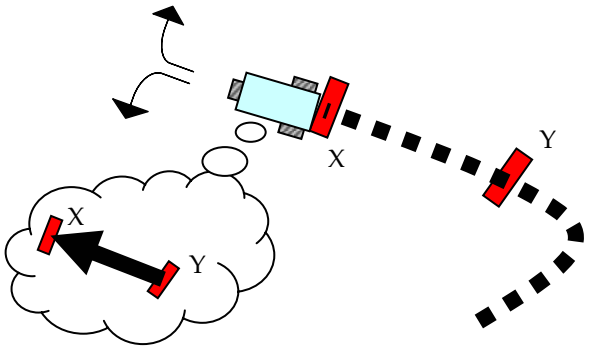


Abbildung 5.5: Verhalten bei einem Déjà-vu

Kapitel 6

Fazit

In diesem Kapitel soll kurz auf die gesammelten Erfahrungen eingegangen werden. Der darauffolgende Abschnitt soll darüber Aufschluss geben, inwiefern sich die RFID-Technologie unserer Meinung nach für die Entwicklung eines smarten, positionsbewussten Putzroboters eignet.

6.1 Zusammenspiel der Hardwarekomponenten

In dieser Laborarbeit haben wir sicherlich eine gewisse Erfahrung mit verschiedenen technischen Komponenten und ihrem Zusammenspiel sammeln können. Dabei ist uns bewusst geworden, wie problematisch solche Komponenten bzw. die Kommunikation zwischen ihnen sein kann und wieviel Zeit dabei verloren geht, diese Probleme zu beheben. Um nur einige Beispiele zu nennen sei auf die Infarotkommunikation zwischen BTnode und RCX oder auch auf die Kabelverbindung zwischen μChip-Reader und Antenne hingewiesen. Sehr ärgerlich war zu Beginn unserer Arbeit der Versuch, die Software-Tools und die BTnode System Software auf einem Windows-Rechner zu installieren. Deshalb haben wir schlussendlich entschieden, mit Linux zu arbeiten, von da an ging alles recht glatt. Das einzige Gerät welches überhaupt keine Mängel aufzuweisen hatte, ist der selbstgekaufte Tastaturstaubsauger (Kompliment an die Herstellerfirma).

Genau weil es so schwierig ist, alles zusammen so zum Laufen zu bringen, wie man es gerne hätte, sind wir mit dem Endprodukt recht zufrieden und unser Interesse, an weiteren solchen Projekten mitzuarbeiten, ist mindestens so stark wie vor Beginn der Arbeit.

6.2 Zusatznutzen der RFID-Technologie

Es war interessant, praktische Erfahrung mit der RFID-Technologie zu sammeln, von der wir schon einiges in Vorlesungen und Zeitschriften gehört hatten. Deshalb möchten wir uns nochmals herzlichst bei Hitachi bedanken, dass sie uns die μChip-Ausrüstung zur Verfügung gestellt haben.

Der Zusatznutzen, den unser mobiler Staubsauger aus der μChip-Ausrüstung ziehen kann, ist jedoch, wenn man die Vorschläge in Kapitel 7 nicht berücksichtigt, mehr oder weniger beschränkt. Es macht z.B. wenig Sinn, für die Grenze ebenfalls RFID-Tags zu verwenden, da man die Tags in einer bestimmten Struktur anordnen muss, damit ein aus iregendeiner Richtung herannahender SVC die Grenztags detektiert. Viel natürlicher wäre dafür, die von LEGO mitgelieferten Tastsensoren an den RCX anzuschliessen. Als Gebietsgrenze würden dann z.B. die Wände eines Zimmers dienen.

Für die Positionierung innerhalb der zu reinigenden Fläche scheint die RFID-Technologie jedoch eher einsetzbar zu sein. Dabei muss jedoch beachtet werden, dass nur dort gereinigt werden kann, wo sich ein RFID-Tag befindet (oder in der Nähe eines solchen). Weiter hat ein auf RFID basierter Smart Vacuum Cleaner, verglichen mit einem Putzroboter, der mit Winkelsensoren ausgerüstet ist, schon ein paar Nachteile. Letzterer kann nämlich das zu reinigende Gebiet kartographieren und somit schon im Voraus berechnen, wie er die zu bewältigende Fläche effizient abfahren muss. Ohne Winkelsensoren hingegen kann man erst bei der Detektierung eines Tags feststellen, wie man sich verhalten muss. Eine Abhilfe wäre da sicherlich eine sequetielle Anordnung der RFID-Tags. Dies ist aber andererseits eher aufwändig in der Herstellung.

Wenn man Hitachi μChip um ein Kollisionsbehandlungs-Protokoll erweitert und die Antenne zirkular polarisierte Wellen aussendet, könnte aus der RFID-Technologie sicherlich noch mehr Nutzen gezogen werden. Bis es soweit ist, sollte man sich auf die Erweiterung und Verbesserung des Putz-Algorithmus konzentrieren, um so von der RFID-Technologie das Bestmögliche rauszuholen.

Kapitel 7: Vorschläge für zukünftige Arbeiten

Kapitel 7

Vorschläge für zukünftige Arbeiten

In diesem Kapitel sollen ein paar Vorschläge, wie man den SVC verbessern könnte, aufgezählt werden. Die Liste ist selbstverständlich nicht vollständig und könnte beliebig erweitert werden.

7.1 Direkte Steuerung

Die Verwendung zweier Plattformen (LEGO-Steuergerät und BTnode) macht die Entwicklung der Software sehr aufwändig. Die Überlegung, dass LEGO Mindstorms die Konstruktion des Staubsaugers vereinfacht, hat sich eher nicht bewahrheitet. Bei einem zukünftigen Prototypen sollte man sich überlegen, die Steuerung von Motoren direkt mit dem BTnode über Relais vorzunehmen. Dadurch kann man sich auch die äusserst fehlerhafte Kommunikation per Infrarot ersparen.

7.2 Virtual Counterpart

Um der Vision des smarten Gerätes von Mark Weiser näher zu kommen, würde sich ein Virtual Counterpart aufdrängen. Beispielsweise könnte eine Überwachungskamera den Putzmodus aktivieren, sobald eine Bewegung einer Person beobachtet oder eine Verschmutzung festgestellt wird. Zudem wäre es möglich, dass mehrere Staubsauger miteinander kommunizieren und so gemeinsam die gleiche Fläche putzen.

Wünschenswert wäre sicherlich auch eine einfachere Handhabung bei der Eingabe von komplizierten Grenzen. Möglicherweise könnte zu jeder Teppichplatte die dazugehörige Karte mitgeliefert werden und man könnte dann bequem mit der Maus am PC die Grenze nachziehen.

7.3 Kontrolle per Mobiltelefon

Da der BTnode ein Bluetooth-Modul besitzt, könnte man sich vorstellen, statt mit dem PC mit einem anderen bluetoothfähigen

Gerät, z.B. mit einem Mobiltelefon, auf den Staubsauger zuzugreifen bzw. seine ausgesendeten Nachrichten zu empfangen. Wir haben das soweit implementiert, dass jedes Bluetooth-Modul unser Staubsaugergerät erkennt (getestet an einem Nokia 6310 Mobiltelefon). Es kann aber noch keine Verbindung mit einem Dienst aufgebaut werden.

7.4 Kombinieren mit weiteren Sensoren

Bei den Testläufen ist uns aufgefallen, dass die Abgrenzung über RFID-Tags nicht unbedingt praktisch ist. Vielleicht sollte man vielmehr eine Linie mit einem Markierungsstift zeichnen und am RCX die Lichtsensoren verwenden, oder man könnte die natürliche Abgrenzung der Wände berücksichtigen. Ohnehin müsste der Staubsauger mit Tastsensoren ausgestattet sein, falls das abgegrenzte Gebiet Inseln in Form von Möbeln enthält oder sich irgendwie verändert, sprich ein Gegenstand hineingesetzt wird.

7.5 Kombinieren mit weiteren µChip-Antennen

Wenn man als Grenzen des zu reinigenden Gebietes Gegenstände oder die Wände eines Raumes nimmt, wäre die Platzierung von RFID-Tags an diesen Gegenständen/Wänden sinnvoll. Man könnte dann z.B. eine zweite, senkrecht aufgestellte Antenne an der Front des Fahrzeugs befestigen, um so die Tags an den Gegenständen/Wänden zu erkennen.

Denkbar wäre auch, dass man mit weiteren µChip-Antennen entferntere RFID-Tags erkennen und im Voraus reagieren könnte. Oder man befestigt die µChip-Antenne an einer beweglichen Vorrichtung am Fahrzeug, so dass sowohl der Boden als auch vertikal verlaufende Oberflächen gelesen werden.

7.6 Aktive Sensorstimulation

Ein garantierter Zusatznutzen hätte die aktive Sensorstimulation (siehe dazu [7]), bei welcher der Sensor im Falle eines Reizes zusätzlich über die Oberfläche (in näherer Umgebung) bewegt wird. Diese Taktik verwenden beispielsweise Ratten mit ihren Fühlern beim Analysieren von Oberflächen.

Der Staubsauger könnte durch die Erkundung der Umgebung eines bestimmten RFID-Tags einen Graphen aufbauen und das Gebiet über logische Punkte kartographieren. Es stellt sich aber die Frage, ob sich mit diesem relativ hohen Rechen- und Speicheraufwand die Reinigungszeit optimieren lässt, oder ob nicht mit simpleren Methoden gearbeitet werden sollte.

Referenzen

[1] Not Quite C - Software & Dokumentation
http://bricxcc.sourceforge.net/nqc/

[2] BTnode - Hardware, Software & Dokumentation
http://www.btnode.ethz.ch

[3] Electrolux - Präsentation des kommerziellen Staubsaugers
http://www.electrolux.com

[4] LEGO Mindstorms - Software, Dokumentation & Community
http://www.lego.com

[5] Hitachi μChip-System - Vertrauliches Dokument

[6] Smart Protocol Analyzer – Software
http://www.imperioustech.com

[7] Plüss, Matthias (2004). Kopflose Intelligenz. Weltwoche Artikel, Ausgabe 01/04.

[8] Andres Erni und Stefan Reichmuth. Bluetooth Anbindung für Lego Mindstorms. Semesterarbeit, ETH Zürich, Schweiz, 2002.

Anhang A: Installations-Anleitung

Anhang A

Installations-Anleitung

Für die Verwendung von NQC für LEGO Mindstorms muss der NQC-Compiler verwendet werden. Diesen findet man unter http://bricxcc.sourceforge.net/nqc/ und auf der mitgelieferten CD-ROM.

Für den BTnode müssen die benötigten Tools wie Compiler, Libraries etc. und die BTnode System Software installiert werden. Unter http://www.btnode.ethz.ch/tools/tools_index.html kann alles, was man dazu braucht, heruntergeladen werden. Auf der mitgelieferten CD-ROM sind diese Tools ebenfalls vorhanden.

A.1 LEGO Mindstorms

Den NQC-Compiler gibt es nur für Windows und Mac, wir haben die Windows-Version verwendet, die man, wie oben schon erwähnt, unter http://bricxcc.sourceforge.net/nqc/ findet.

A.2 BTnode

Für unser Projekt haben wir einen Linux-Rechner verwendet, nachfolgende Anweisungen beziehen sich deshalb auf die Linux-Installation der Tools und der BTnode System Software. Für Windows sei auf die oben erwähnte Website verwiesen.

A.2.1 Tools

Zu den benötigten AVR GNU Tools gehören:

- die AVR Binutils
- der AVR GCC Compiler
- die Library für den AVR Microcontroller und
- die Utilities für den Hardware-Programmierer (UISP)

Nachdem diese Tools heruntergeladen wurden, müssen sie zwingend ins /usr/local/avr/-Verzeichnis entpackt werden, z.B. mit:

- cd /usr/local
- tar xvfz /tmp/avr-binutils-XXX.tgr.gz
- tar xvfz /tmp/avr-gcc-XXX.tar.gz

- tar xvfz /tmp/avr-libc-XXX.tar.gz
- tar xvfz /tmp/uisp-XXX.tar.gz

XXX gibt die Versionsnummer der Tools an, /tmp/ bezeichnet das Verzeichnis, in welches die Tools heruntergeladen wurden.

Anschliessend sollte der Pfad /usr/local/avr/bin zur $PATH und /usr/local/avr/man zur $MANPATH Variable hinzugefügt werden.

A.2.2 BTnode System Software

Die System Software kann via CVS heruntergeladen werden (man findet sie aber auch auf der CD-ROM). Dafür ist ein CVS-Client notwendig.

Für den Login ist folgendes Kommando notwendig:

cvs -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/btnode login

Wenn ein Passwort verlangt wird, einfach Enter oder OK drücken. Danach kann die Software mit folgendem Kommando heruntergeladen werden:

cvs -z3 -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/btnode co btnode_system

Am besten ändert man den Namen des Verzeichnisses in btnode um und verschiebt es nach /usr/local/. Nun gilt es folgende Kommandos auszuführen, um die Software zu konfigurieren, kompilieren und zu installieren:

- cd /usr/local/btnode
- ./bootstrap
- ./configure
- make
- make install

Anhang B

Bedienungs-Anleitung

B.1 LEGO Mindstorms

Auf der CD findet man unter /Lego/ den Compiler und die Programme, welche auf den RCX geladen werden sollen:

- nqc.exe ist der Compiler
- svc.nqc ist das Hauptprogramm für den Smart Vacuum Cleaner
- clean_movement.nqc wird von svc.nqc verwendet und definiert die Bewegung des SVC beim Putzen

Wir haben den im LEGO Minstorms Set mitgelieferten USB Tower unter Windows verwendet, um Programme auf den RCX zu laden. Deshalb sollte folgendes Kommando ausgeführt werden (siehe readme.txt für genauere Angaben):

- set RCX_PORT=usb

Danach sollte der USB Tower in die Nähe des Infrarotsenders des RCX gebracht werden und mit folgendem Kommando das Programm kompiliert und auf den RCX geladen werden (der RCX muss eingeschaltet sein):

- nqc –d –TRCX2 svc.nqc

TRCX2 gibt an, dass das RCX2 Produkt von LEGO Mindstorms benutzt wird (anhand von nqc –help können alle anderen möglichen Optionen für den NQC-Compiler aufgelistet werden).

B.2 BTnode

Zuerst sollte der svc-Ordner (auf der CD unter /BTnode/) nach /usr/local/btnode/examples der Installation unter A.2.2 kopiert werden. Im /usr/local/btnode/examples/svc-Verzeichnis kann dann mit folgenden Kommandos die Anwendung kompiliert und auf den BTnode geladen werden (vorausgesetzt der Hardware-

Programmierer ist mit dem PC und dem BTnode verbunden, und der BTnode wird anhand von Batterien mit Strom versorgt):

- make btnode
- nochmals make btnode
- make upload btnode

B.3 PC

Für die Anwendung auf dem PC sollte der Ordner svc-bt-cmd (auf der CD unter /PC/) wiederum z.B. nach /usr/local/btnode/examples verschoben werden. Danach kann in /usr/local/btnode/examples/svc-bt-cmd mittels folgender Kommandos das Programm kompiliert und gestartet werden (es muss dafür ein Bluetooth-Modul an den PC angeschlossen sein):

- make i386
- ./i386/svc-bt-cmd –u0 /dev/ttyS0 57600 fc

Dabei wird angenommen, dass das Bluetooth-Modul mit Port 0 des Rechners verbunden ist. Wenn also oben genanntes Kommando nicht klappt, einfach das Modul an den anderen Port anschliessen.

Nun können Kommandos eingegeben werden. Zuerst sollte ein Inquiry gemacht werden, um den BTnode aufzuspüren. Danach kann eine L2CAP-Verbindung mit dem BTnode erstellt werden:

- > inq
  Nun sollte eine Liste mit MAC-Adressen von gefundenen Bluetooth-Geräten erscheinen, darin sollte auch der BTnode vorkommen. Wenn dies nicht der Fall ist, sollte nochmals ein Inquiry gemacht werden.
- > con X 101
  X gibt an, mit welchem Element der von inq gelieferten Liste wir eine Verbindung aufbauen wollen. Wenn also z.B. der BTnode das einzig gefundene Bluetooth-Gerät ist, sollte con 0 101 verwendet werden. 101 gibt die PSM (Protocol Service Multiplexer) an (eine PSM ist ähnlich zu einem Port und gibt an, welchen Service wir vom BTnode angeboten haben möchten. Siehe dazu auch die main-Methode in svc.c).

Wenn alles geklappt hat, sollte ein Menu erscheinen, welches angibt, welche Kommandos an den BTnode geschickt werden können (bevor aber irgendwelche Kommandos verschickt werden, sollte man überprüfen, ob auf dem RCX das heruntergeladene Programm läuft (auf Run drücken)).

Zuerst sollte in den Marking Mode gewechselt werden:

- send *local_cid* m

Die local_cid wird jeweils angezeigt, z.B. 67. Nun kann die Grenze eingelesen werden, einfach die µChip-Antenne des SVC über die Grenztags halten. Bei jedem eingelesenen Tag sollte der RCX Töne von sich geben.

Wenn die Grenze fertig eingelesen wurde, kann in den Cleaning Mode gewechselt werden. Dafür sollte sich der SVC innerhalb der zu reinigenden Fläche befinden. Mit folgendem Kommando wird der Putzvorgang aktiviert:

- send *local_cid* c

Es sollten jeweils auf dem PC Meldungen erscheinen, die beschreiben, was der SVC jeweils macht. Weiter kann auch vom RCX-Display abgelesen werden, welche Meldungen der RCX bekommt (für die Bedeutung der Meldungen sei auf den Source-Code svc.nqc verwiesen).

Alle weiteren, vom PC aus versendbaren Kommandos können im Source-Code svc.c eingesehen werden.

Index