Agentenarchitekturen

17. Dezember 2002

Eine Studienarbeit von:

Mirco Gerling

Hochschule für angewandte Wissenschaften
Berliner Tor 3
20099 Hamburg, Germany

http://www.informatik.haw-hamburg.de/~robots

Inhaltsverzeichnis

1 Einleitung

Roboter gewinnen immer mehr an Bedeutung. Ob für den Privatgebrauch oder in der Industrie. Es gibt Roboter die Rasen mähen, Schwimmbäder reinigen oder Waren in einem Lager transportieren.

In dieser Arbeit soll die Metapher eines Büroboten verwendet werden. Es sollen Aufträge an einen Roboter gesendet werden können, der per Funk bzw. Bluetooth mit einem PC oder ähnlichem Gerät verbunden ist.

In dieser Arbeit soll der Unterschied zwischen Modellbasierten und Verhaltensbasierten Agenten gezeigt werden.

2 Agenten

2.1 Charakteristika

Agenten sind in der Regel Programme, denen man einen Auftrag gibt und sie arbeiten diesen selbständig ab. Genauere Informationen gibt es im Handbuch der KI [HandbuchKI].

2.2 Deliberative Architektur (Modellbasierte)

Es gibt zwei unterschiedliche Gruppen von modellbasierten Agenten. Die einen haben eine vorgefertigte Karte ihrer Umgebung und die anderen erkunden ihre Umgebung und fertigen sich ihre Karte selbst an. Beide Systeme versuchen ihr Wissen über die Umgebung logisch umzusetzen. Das größte Problem der modellbasierten Agenten ist die Bestimmung der eigenen Position innerhalb der Umgebung. Nur wenn der Roboter weiß, wo er sich befindet, kann er auch im Bereich einer Karte navigieren. Das Prinzip basiert auf der Planung des Verhaltens mit Hilfe der Karte.

Eine Schwäche liegt in der mangelnden Behandlung von Ereignissen, die in einer dynamischen Umgebung, wie z.B. unerwartete Hindernisse, auftreten können.

2.3 Verhaltensbasierte Architektur

Verhaltensbasierte Agenten benötigen kein Wissen über ihre Umwelt. Sie verhalten sich rein reaktiv. Wenn ein bestimmtes Ereignis eintritt, reagieren sie nach einem vorher für dieses Ereignis definierten Muster. Dies tun sie unabhängig von ihrem Standort. Dieses Verhalten kann durchaus in eine Sackgasse führen.
Aus diesem Grund sollten möglichst alle Ereignisse dem Agenten schon vorher bekannt sein, damit er darauf reagieren kann. So würde man eine möglichst geringe Fehleranfälligkeit erreichen. Verhaltensbasierte Roboter sind nicht in der Lage, auf unvorhergesehene bzw. nicht im Verhaltensmuster festgelegte Ereignisse zu reagieren.

2.4 Hybride Architektur

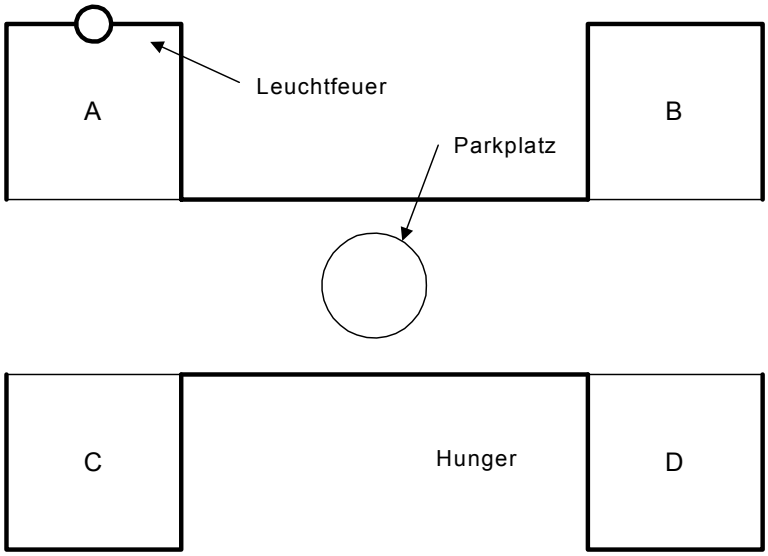
Hier werden modellbasierte und verhaltensbasierte Implementationen konkurrierend parallel betrieben. Über Prioritäten und die Anknüpfung von Verhalten an externe Signale wird nun das Verhalten ausgewählt, das den Roboter kontrollieren soll.

Hybride Architekturen sind nötig, da nur eine Architektur gewisse Nachteile mit sich bringen kann.
Beispielsweise ist es bei einer Verhaltensbasierten Architektur schwer den Roboter relativ zum Weltmodell zu lokalisieren. Rein reaktive Roboter-Systeme sind nicht für alle Anwendungen zu gebrauchen. In der Situation wo eine Welt modelliert werden kann und sich die Welt während der Ausführung garantiert nicht verändert, dort wird eine deliberative Architektur bevorzugt. In der realen Welt wo es biologische Agenten gibt, gibt es keine Bedingungen die rein deliberative Architekturen bevorzugen. Um das volle Potential eines Verhaltensbasierten Roboter-Systems zu nutzen, ist eine Hybride Architektur sehr sinnvoll.

3 Ein Szenario

3.1 Die Aufgabe

Es gibt vier Häuser die angefahren werden können. Diese Häuser sind durch einen Gang verbunden. Es befinden sich jeweils zwei Häuser an jedem Ende des Ganges. Es gibt ein Leuchtfeuer an dem sich der Roboter orientieren kann. Dieses Leuchtfeuer wird z.B. an Haus A montiert. Der Roboter soll Aufträge annehmen können und diese ausführen. Zu einem unbestimmten Zeitpunkt und an einem unbestimmten Ort kann dann eine Lichtquelle eine Futterquelle signalisieren. Zu dieser Quelle soll der Roboter fahren, um Energie zu tanken. Sind die Ressourcen aufgefüllt, nimmt der Roboter wieder seine Arbeit auf und kann wieder Aufträge entgegen nehmen.



3.2 Modellbasierter Anteil

Es soll möglich sein, dem Roboter über eine Anwendung, die z.B. auf einem PC läuft, Aufträge zu senden um z.B. von einem Haus zum Anderen zu fahren bzw. von einem Haus etwas abzuholen und es zu einem anderen Haus zu bringen. Als Orientierung soll ein Leuchtfeuer dienen. So ist es möglich in etwa eine Ahnung zu haben in welche Richtung der Roboter gerade ausgerichtet ist. Bei Eintritt in den modellbasierten Modus soll der Roboter im Gang stehend in Richtung von Haus A und Haus C ausgerichtet sein. Für die folgenden Aufträge soll sich der Roboter merken, in welche Richtung er ausgerichtet ist.

3.3 Verhaltensbasierter Anteil

Der Verhaltensbasierte Anteil soll darin bestehen, dass der Roboter einer Lichtquelle folgt. Dies soll eine Futterquelle oder eine Ladestation simulieren. Der Roboter soll in den verhaltensbasierten Modus übergehen, wenn z.B. eine bestimmte Anzahl an Aufträgen verarbeitet wurde oder eine bestimmte Strecke zurückgelegt wurde. Es soll eine Größe sein, die sich proportional zu den Ressourcen des Roboters verhält. Der Zeitpunkt und der Ort von der Lampe sind jeweils zufällig.
Sense-Act:

Bei Sense-Act werden die Sensorwerte direkt auf die Räder gegeben. Der Roboter

Event-Act:

Es gibt einen 2 stufigen Grad von Hunger:

1. Preferiert laden:
Wenn eine Möglichkeit zum Laden auftritt, dann fährt er dorthin. Sucht aber nicht gezielt danach.

2. Nur laden:
Roboter nimmt keine Aufträge mehr an. Er „denkt“ nur noch ans Laden.

4 Ein Hard/Softwaredesign

4.1 Die Laborumgebung

Bei dem Labor der HAW Hamburg handelt es sich um ein speziell für das Experimentieren mit Robotern eingerichtetes Labor. Es verfügt über eine umfangreiche Ausstattung die zum Bau von autonomen Robotern benötigt wird.

In dem Labor gibt es verschiedene Bereiche an denen gearbeitet wird. Es gibt Roboter, die mit dem MIT 6.270 Board ausgestattet sind bzw. ausgestattet werden können [Martin, MIT6.270]. Seit wenigen Semestern gibt es auch Projekte in denen mit den Mindstorm Baukästen von Lego gearbeitet wird[Lego].

Zudem gibt es noch die Pioneer Roboter. Diese sind fertig gekaufte Roboter von der Firma *Activ*Media Robotics [Pioneer]. Die Pioneer Roboter verfügen über eine gute Ausstattung. Informationen zu dem Labor gibt es unter: [HAWRobots]

4.2 Hardwareumgebung für die Arbeit

4.2.1 Computing Hardware

Die Zentrale Hardware des Roboters ist ein MIT 6.270 Board [MIT6.270]. Dies ist eine Einheit, die speziell für den Einsatz mit Robotern entwickelt wurde. Die CPU ist ein Motorola 68HC11. An dieses Board können diverse Aktuatoren und Sensoren angeschlossen werden. Es können sowohl digitale als auch analoge Sensoren angeschlossen werden. Als Sensoren kommen z.B. Lichtschranken, Photowiderstände und Taster in Frage.

Das MIT 6.270 Board gehört zur Familie des Handyboard und des Lego Mindstorm RCX [LEGO]. Das Handyboard ist der Nachfolger des MIT 6.270 Board.

4.2.2 Die Plattform

Die Plattform basiert auf einer Differentialsteuerung. Dies bedeutet, dass es ein Pendelrad hinten gibt. Auf der linken und der rechten Seite gibt es jeweils ein angetriebenes Rad. Mit diesen Antriebsrädern kann der Roboter angetrieben und gelenkt werden. Dies ist ein großer Vorteil. Die Plattform ist dadurch sehr wendig. Sie kann um den Mittelpunkt der Antriebsachse gedreht werden.

Roboter in der Ansicht von unten

<u>Aktuatoren</u>

Angetrieben werden die Räder über umgebaute Servo Motoren, die aus dem Bereich des Modellbau stammen. Üblicherweise werden sie in Fernsteuermodellen benutzt. In nicht umgebauter Form lassen sich die Servo Motoren um einen festen Winkelbetrag drehen. Bestandteile sind ein DC Motor, ein Getriebe und ein Positionssensor mit dessen Hilfe über eine Schaltung der Drehwinkel kontrolliert werden kann. Die Positionierung geschieht durch Pulsweitenmodulation (PWM) und

die Kontrollinformation durch den Positionssensor. Ein Servo Motor ist durch das eingebaute Getriebe relativ kräftig. Ein weiterer Vorteil ist, dass sie sich leicht montieren und ansteuern lassen.

Sensoren

Die Plattform des Roboters ist mit diversen Sensoren ausgestattet.

Bumper 1

Mit Dip-Schaltern kann man recht einfach Kontakte zum Umfeld registrieren. Ich habe für links und rechts jeweils einen Schalter gewählt, um Kollisionen mit den Wänden zu registrieren. Diese Schalter habe ich mit kurzen Stücken von Kabelbindern verlängert, um möglichst rechtzeitig eine Kollision zu erkennen.

Bumper 2

Zur Erkennung von Frontal-Kollisionen habe ich einen weiteren Sensor montiert. Es handelt sich um zwei Drahtschlingen aus Gittarensaiten, die locker auf Lego Plättchen verklebt sind. Die beiden Drahtschlingen sind mit einem festen Draht verbunden um die Aufprallfläche zu vergrößern. Werden die Drahtschlingen zu sehr durch Druck belastet, so berühren sie einen Drahtkontakt, der dann die Kollision signalisiert.

Shaft Encoder

Zur Messung von Wegstrecken und Rotationen dienen die Shaft Encoder. Sie bestehen aus kleinen Lochscheiben, die auf der selben Achse wie die Antriebsräder montiert sind. Es gibt in der Regel eine rote LED, die Licht in Richtung der Lochscheibe aussendet. Auf der anderen Seite der Lochscheibe befindet sich ein Phototransistor. Mit diesem werden die Impulse aufgenommen, die durch das Verdecken der LED durch die Lochscheibe entstehen. Die Shaft Encoder werden an die Ports 0 und 1 angeschlossen. Für das Ansteuern und Auslesen ist eine Programmbibliothek in Interactive C vorhanden.

Sie bietet folgende Funktionen:

```
void enable_encoder(int encoder)

void disable_encoder(int encoder)

void reset_encoder(int encoder)

int read_encoder(int encoder)
```

Photowiderstände

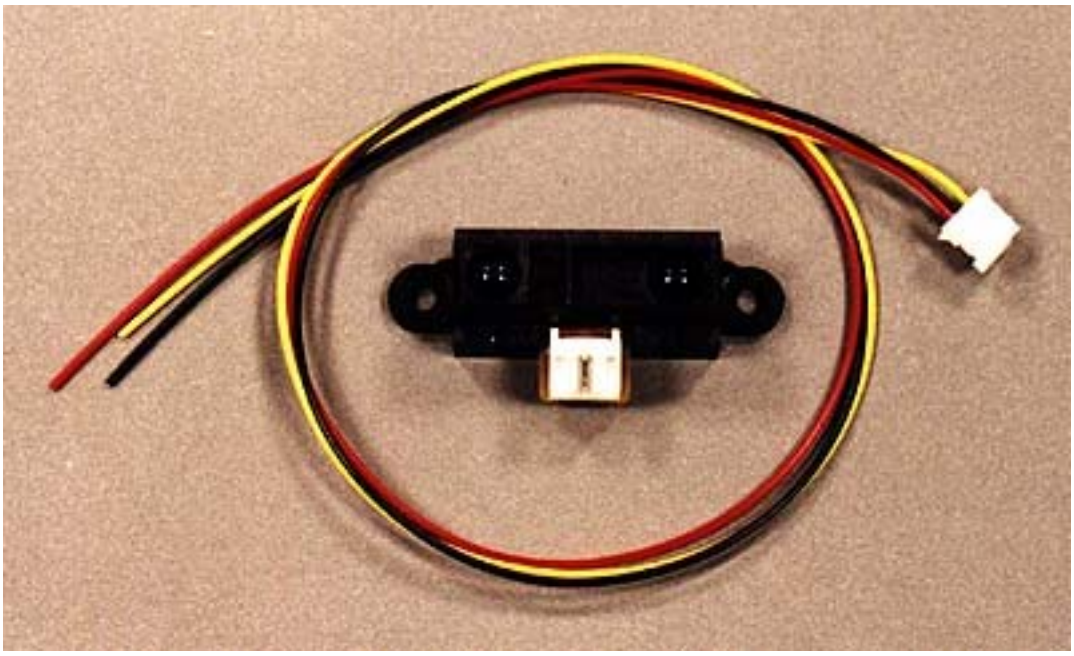
Photowiderstände reagieren auf Licht. Sie verändern ihren Widerstand mit den Lichtverhältnissen. Daher sind sie leicht zu verwenden.

Die Plattform ist mit zwei Photowiderständen ausgerüstet, die auf den Boden gerichtet sind. Damit kann man Unterschiede am Boden erkennen. Am Besten sind sie für schwarz/weiß-Unterschiede geeignet.

Zwei weitere Photowiderstände sind vorne links und rechts am Roboter montiert. Sie sind mit kleinen Reflektorschirmen versehen, damit sich eine Beleuchtung stärker auswirkt.

Abstandssensoren (SHARP GP2D12)

Zur Abstandsmessung zu Hindernissen habe ich Sensoren von der Firma Sharp vom Typ GP2D12 benutzt. Sie haben den großen Vorteil, nicht besonders auf fremdes Licht zu reagieren. Sie arbeiten in einem Bereich von ca. 10 – 80 cm recht zuverlässig. Der Stromverbrauch liegt bei ca. 50 mA/h je Sensor. Versorgt werden die Sensoren über einen Verstärker, der über eine eigene 7,2 V Stromversorgung verfügt. Er besitzt vier Anschlüsse für Sensoren. Die Signale werden über Operationsverstärker an die Ausgänge geleitet und an die analogen Eingänge des 6.270 Boards gelegt.



Abstandssensor (SHARP GP2D12)

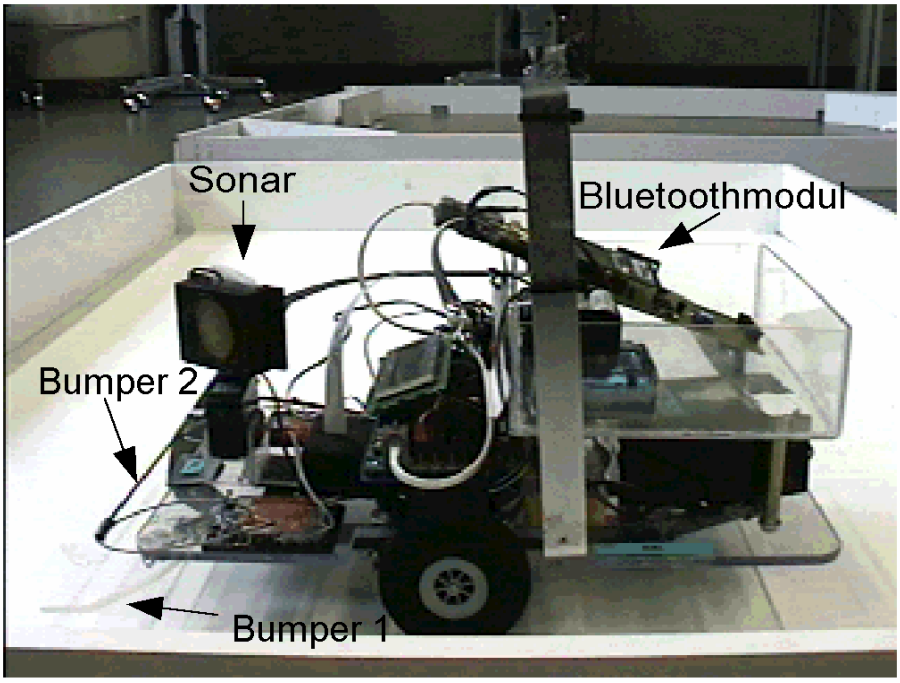
Sonar

Außerdem ist es möglich, einen Sonarsensor anzuschließen. In dieser Arbeit benutze ich einen Sensor von der Firma Polaroid. Es gibt einige Funktionen, die von Oliver Koeckritz und Gunther Klemke entwickelt wurden, die den Zugriff auf den Sonarsensor ermöglichen. Der Sensor ermittelt die Entfernung zu einem Hindernis. Auf dieser Plattform ist der Sensor auf einem Servo montiert. Dadurch lässt sich der Sensor in einem Bereich von180° schwenken

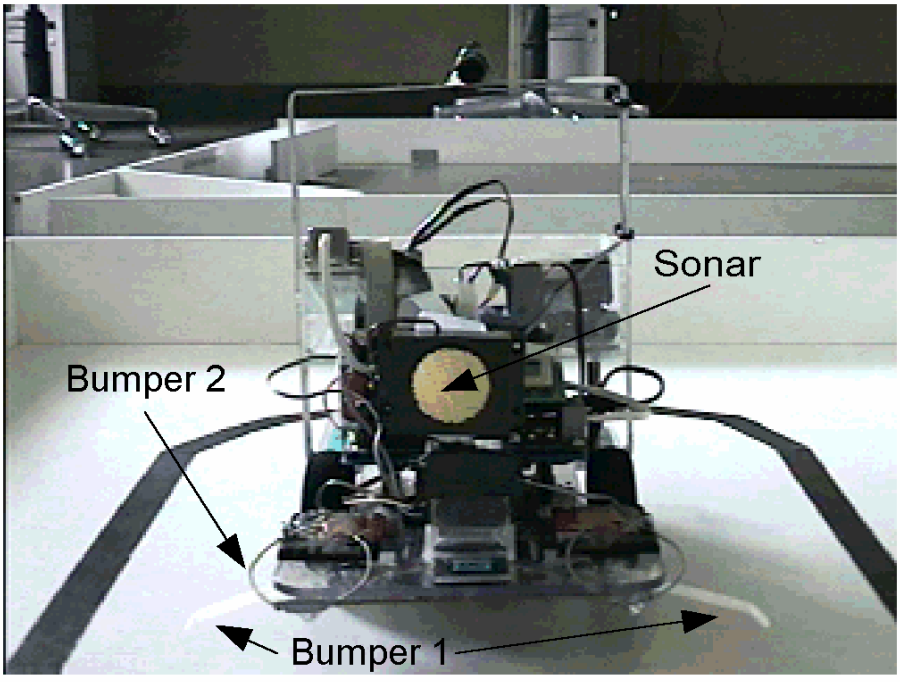
Beacon

Ein weiterer Sensor ist ein Infrarotsensor für moduliertes Infrarot Licht. Er dient zum Finden von sogenannten Infrarot Beacons bzw. Leuchtfeuern. Der Sensor erkennt moduliertes Infrarot Licht mit einer Trägerfrequenz von 40 kHz. Auf die Trägerfrequenz kann eine Frequenz von 100 Hz oder 125 Hz moduliert werden. Die Sensoren können an die Ports 4 bis 7 des 6.270 Boards angeschlossen werden. Sie müssen jedoch abgeschirmt werden, damit nicht fremdes Licht das Ergebnis verfälscht.

Die Kommunikation soll über zwei Bluetooth Module realisiert werden, welche jeweils an die serielle Schnittstelle des PC und des 6.270 Boards angeschlossen werden können.



Seitenansicht des Roboters



Frontansicht des Roboters

4.3 Die Softwareumgebung

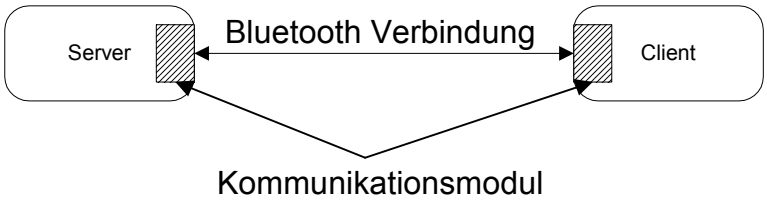
Als Entwicklungsumgebung wird das Programm Interactive C Version 3.2 von den Newton Research Labs verwendet. Interactive C ist ein Subset von der Programmiersprache C. Es bietet eine Bibliothek mit diversen Funktionen, die speziell für das 6.270 Board bestimmt sind. Interactive C bietet leider nur einfache Kontrollstrukturen wie *for, while, if-else* und *break*. Ein einfaches Multitasking ist auch möglich. Es bietet keine Semaphoren oder ähnliche Mechanismen.

In der Interactive C Bibliothek gibt es Funktionen für das Einlesen von Sensordaten und die Bereitstellung von Steuerdaten auf die Ausgänge. Es gibt Zeit-, Ton-, Menü- und Diagnose-funktionen.

Zur Entwicklung einer Windows Anwendung benutze ich Visual Basic 6.0. Mit dieser Entwicklungsumgebung lassen sich recht einfach und schnell kleinere Windows Anwendungen erstellen.

5 Design

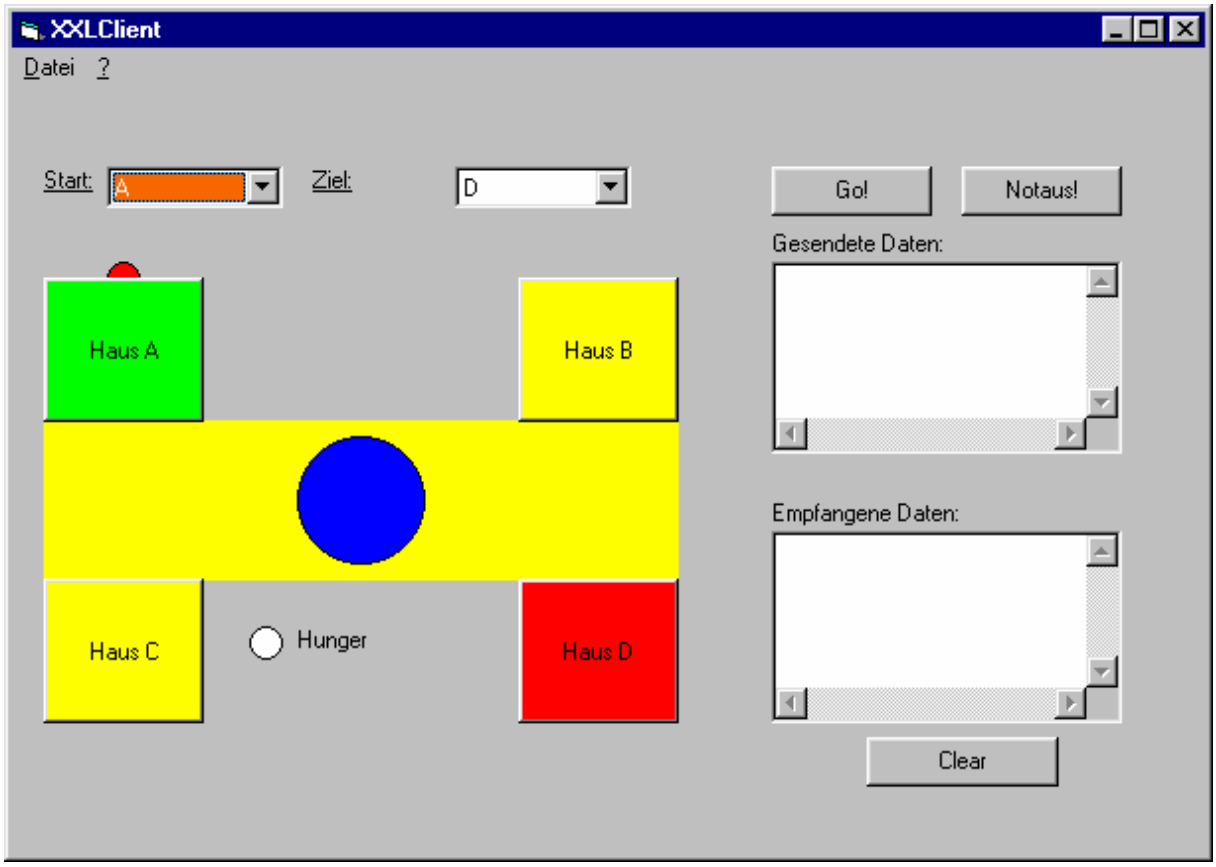
Der Roboter und die Windows Anwendung stellen eine Client-Server Architektur dar. Der Roboter stellt der Server dar. Er nimmt Aufträge und Anfragen entgegen. Die Windows Anwendung stellt den Client dar. Sie sendet Aufträge und Anfragen an den Roboter.



5.1 Client

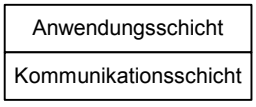
Als Client-Software dient eine Windows Anwendung. Sie bietet die Möglichkeit Befehle an den Server (Roboter) abzusetzen. Die Übertragung der Daten soll über eine serielle Verbindung über Bluetooth erfolgen. Die Anwendung bietet Anzeigen für gesendete und empfangene Daten.

Die Anwendung könnte z.B. auch auf einem PDA mit integriertem Bluetooth laufen. So könnte man auch mit einem Mobilen Gerät Aufträge absetzen.



5.1.1 2-Schichten Modell

Der Client besteht aus zwei Schichten, der Anwendungsschicht und der Kommunikationsschicht. Die Anwendungsschicht besteht aus der graphischen Oberfläche und ihren Funktionalitäten. Die Kommunikationsschicht wird durch ein Kommunikationsobjekt von Visual Basic und die benötigten Funktionen dargestellt.



Die Anwendungsschicht greift über eine Schnittstelle auf die Kommunikationsschicht zu. Es gibt Funktionen (Dienste) zum Öffnen und Schließen einer Verbindung und zum Senden und Empfangen bzw. auslesen von empfangenen Daten.

Die Kommunikationsschicht stellt die Funktionen zum Austausch vom Informationen zur Verfügung. Durch die Kapselung soll es möglich sein, die Implementierung der Kommunikationsschicht beliebig zu ändern. Der Anwendungsschicht müssen lediglich die selben Dienste angeboten werden. Dadurch wird eine Transparenz erzielt, die es auch zum Beispiel ermöglicht statt einer seriellen Verbindung eine TCP/IP Verbindung zu verwenden.

5.1.2 Protokoll

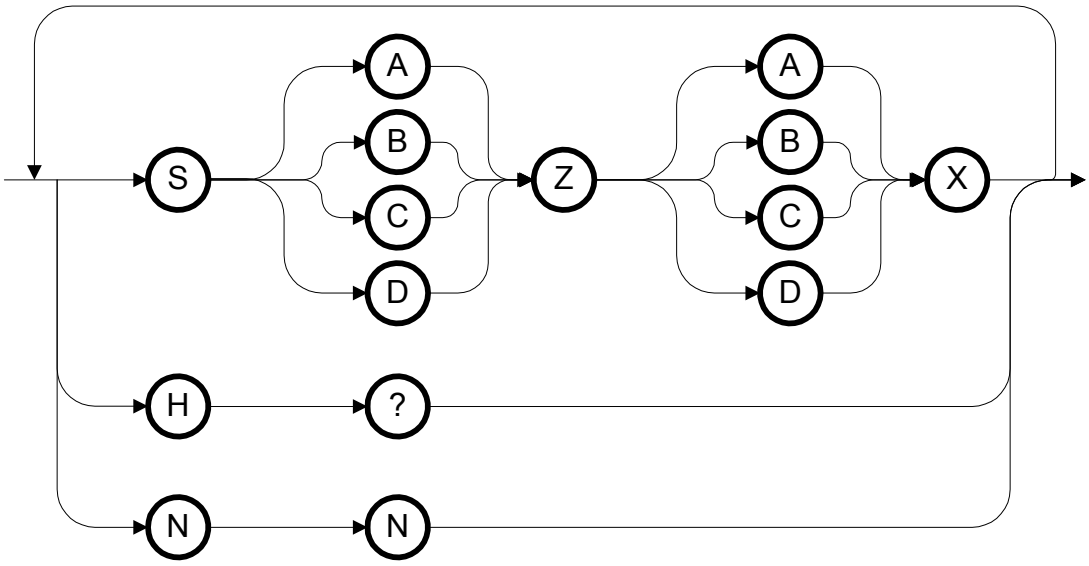
Die Übertragung erfolgt nach einem sehr einfachen Protokoll. Zur Flusskontrolle antwortet der Server mit der gleichen Anzahl von Zeichen, die er empfangen hat. Bei

Befehlen wird die gleiche Zeichenkette zurück geschickt und bei Anfragen wird das erste Zeichen der Anfrage und als zweites Zeichen die Antwort geschickt.

Als erstes Zeichen wird ein „S“ für den Start übertragen. Dann folgt das Haus z.B. „A“. Das Ziel beginnt mit „Z“ und darauf folgt das Haus z.B. „D“. Abschließend folgt ein Endzeichen „X“. So soll es möglich sein Übertragungsfehler auf der Seite des Roboters zu erkennen. Zusätzlich schickt der Roboter die Zeichen zurück. So kann über die Anwendung kontrolliert werden, ob alle Zeichen richtig angekommen sind. Falls ein Fehler aufgetreten ist, können die Daten erneut gesendet werden.

Für weitere Funktionalitäten ist es möglich noch weitere Befehle bzw. Abfragen an den Server zu senden. Es gibt noch die Möglichkeit einen Notstop-Befehl zu senden. Dies geschieht mit der Zeichenkette „NN“ und wird entsprechend mit „NN“ bestätigt. Um den Hungerstatus abzufragen wird die Zeichenkette „H?“ an den Server gesendet. Dies wird mit einem „HN“ für keinen Hunger, „HM“ für mittleren Hunger und „HB“ für großen Hunger beantwortet. Die Hungerabfrage geschieht über die Anwendung in zyklischen Abständen. Die Zeitpunkte werden über einen Timer gesteuert. Sobald keine anderen Zeichen gesendet werden, wird der Timer für die Hungerabfrage gestartet. Läuft der Timer ab, so wird die Zeichenkette für die Hungerabfrage gesendet. Die Antwort wird im Fenster für die empfangenen Daten angezeigt und farbig wird der Hunger auch dargestellt.

Zur Kommunikation mit dem Roboter kann man auch auf ein Terminalprogramm wie z.B. das Hyperterminal zurückgreifen. Dort kann man auch die Zeichenketten wie im Protokoll eingeben und somit Aufträge und Abfragen senden.



5.2 Server

5.2.1 Auftragsabwicklung

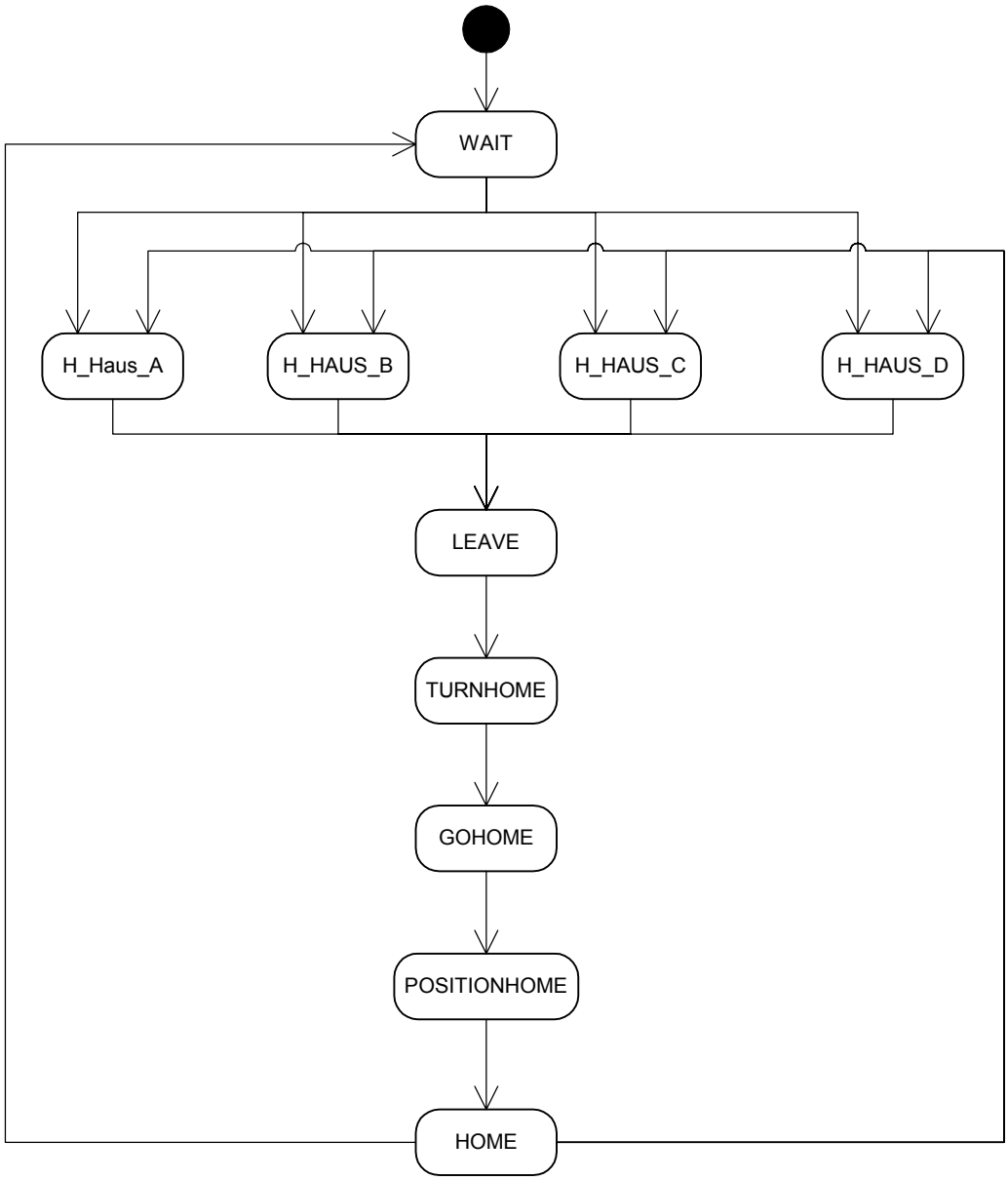
Der Server bzw. Roboter nimmt jeweils einen Auftrag an. Er kann erst einen weiteren Auftrag annehmen, wenn er nicht mehr beschäftigt ist. Ist der Roboter gerade im verhaltensbasierten Modus, so nimmt er ebenfalls keine Aufträge an.

Wird der Roboter in Bereitschaft versetzt, so richtet er sich mit Hilfe des Beacon in Richtung von Haus A aus und wartet auf einen Auftrag.

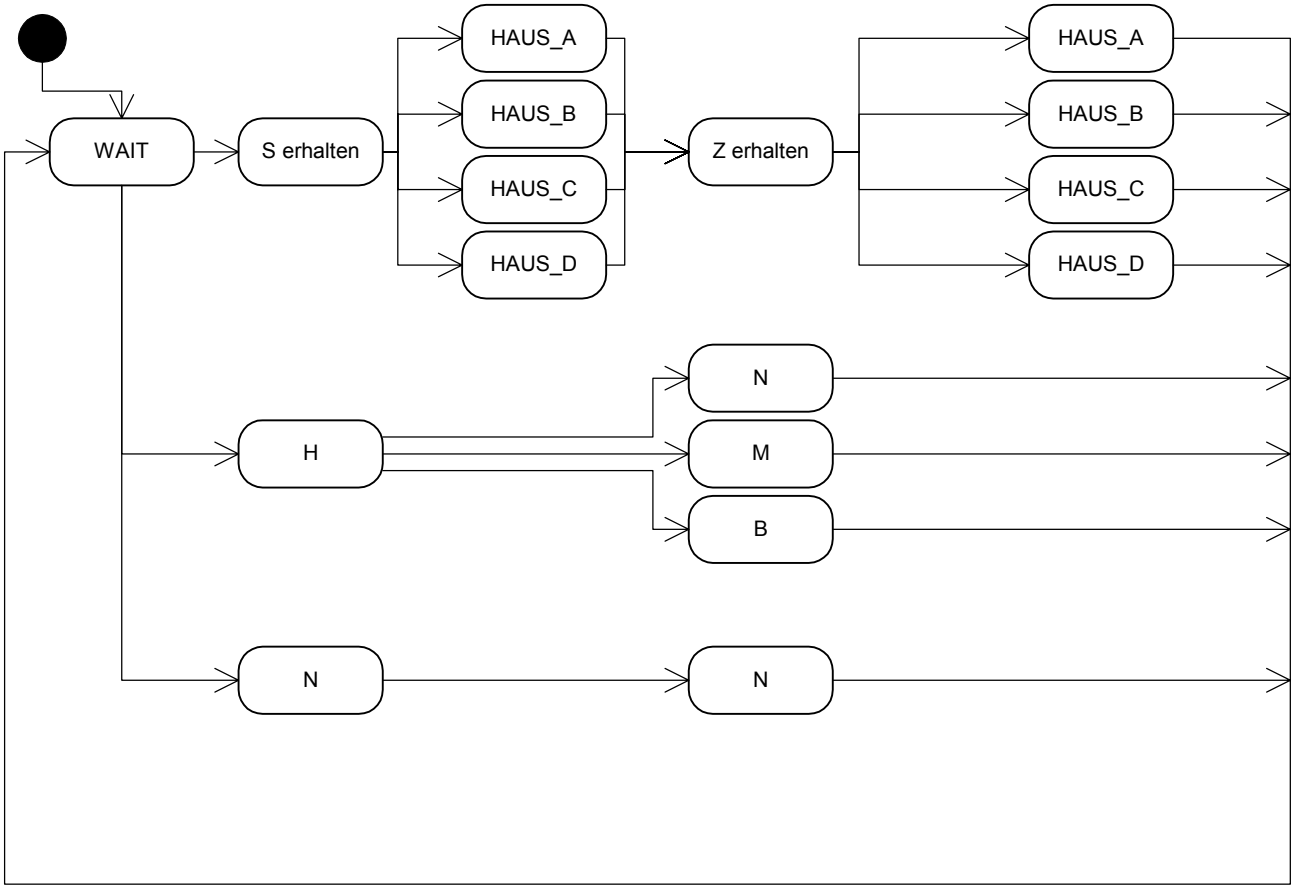
5.2.2 Prozesse

Der Server läuft mit drei Prozessen. Ein Prozess ist für die Auswertung der Sensordaten zuständig. Ein weiterer ist mit der Auftragsabwicklung bzw. dem Anfahren der Häuser „beauftragt“. Der dritte Prozess ist für die Kommunikation über die serielle Verbindung zuständig. Es ist somit jederzeit möglich interne Zustände des Roboters zu erfragen.

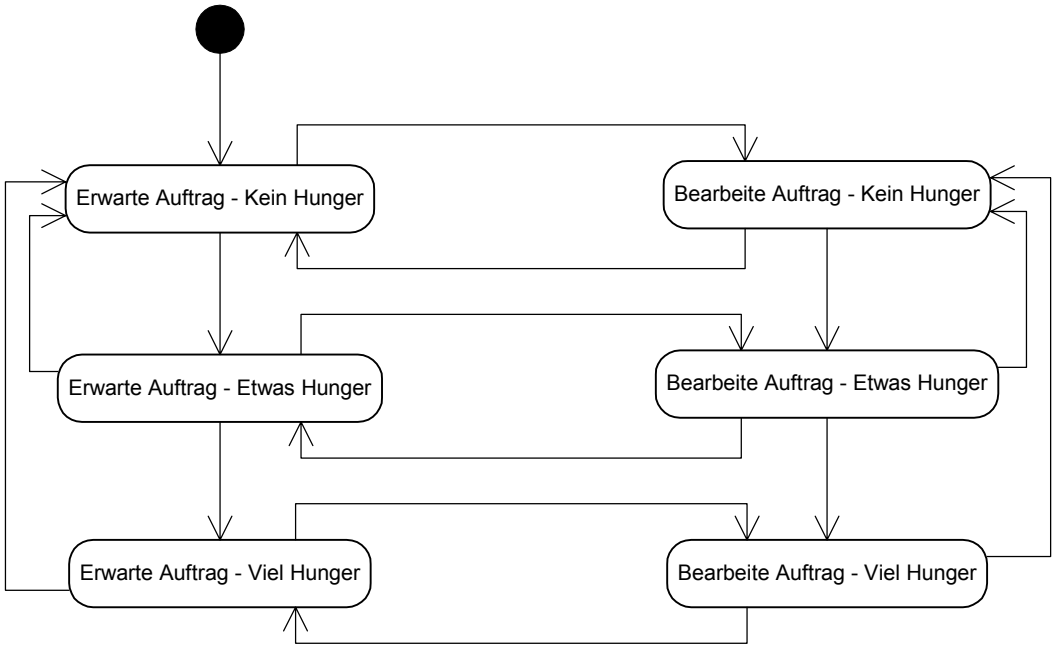
Der Zustandsautomat für den Hauptprozess sieht wie folgt aus:



Kommunikationsprozess:

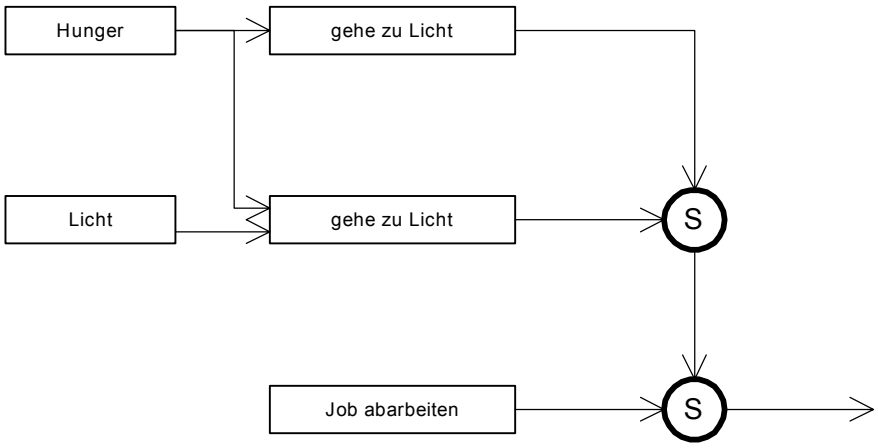


Die Zustände des Systems:



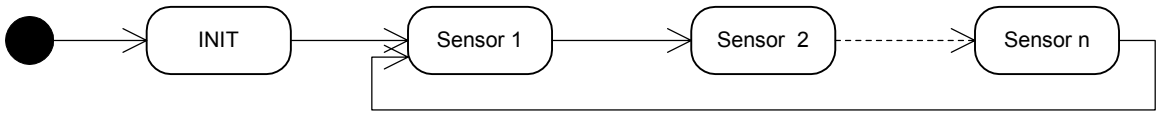
Bei der Realisierung in der Subsumption Architektur wird der Hunger anfangs ganz überlagert. Sobald etwas Hunger da ist, wird der Hunger im Zusammenwirken mit dem Licht durchgelassen. Ist der Hunger am stärksten, so spielt nur noch der Hunger eine Rolle.

Möglichkeit einer Realisierung in der Subsumption Architektur:



Der Sensorprozess fragt permanent der Reihe nach alle Sensoren ab. Wird ein Sensorwert ausgelesen, so wird er in einer Variablen gespeichert. Bei einigen Sensoren wird der Mittelwert aus mehreren Messungen ermittelt. Vom Sonarsensor werden sieben Abstandswerte gespeichert. Der Sensor wird in jedem Durchlauf des Sensorprozesses um 30 Grad weiter gedreht.

Schematische Darstellung des Sensorprozesses:



6 Die Realisierung

6.1 Die Programmiersprache Interactive C

Die für die Steuerung des Roboters verwendete Programmiersprache Interactive C bietet diverse Funktionen. Zu den wichtigsten für diese Arbeit zählen z.B. die Funktionen für das Shaftencoding. Hiermit lassen sich die Wegstrecken bestimmen.

void enable_encoder(int encoder)
void disable_encoder(int encoder)
void reset_encoder(int encoder)
int read_encoder(int encoder)

Zusätzlich sind die Funktionen für die Beaconerkennung von Bedeutung. Die Funktionen sind für den Empfang von Infrarot Signalen bestimmt.

void ir_receive_on()
void ir_receive_off()
void set_ir_receive_frequency(int f)
int ir_counts(int p)

Für die serielle Kommunikation werden folgende Befehle für den Speicherzugriff benötigt.

Zur Aktivierung der seriellen Kommunikation wird die folgende Funktion benötigt.

void poke(int loc, int byte)

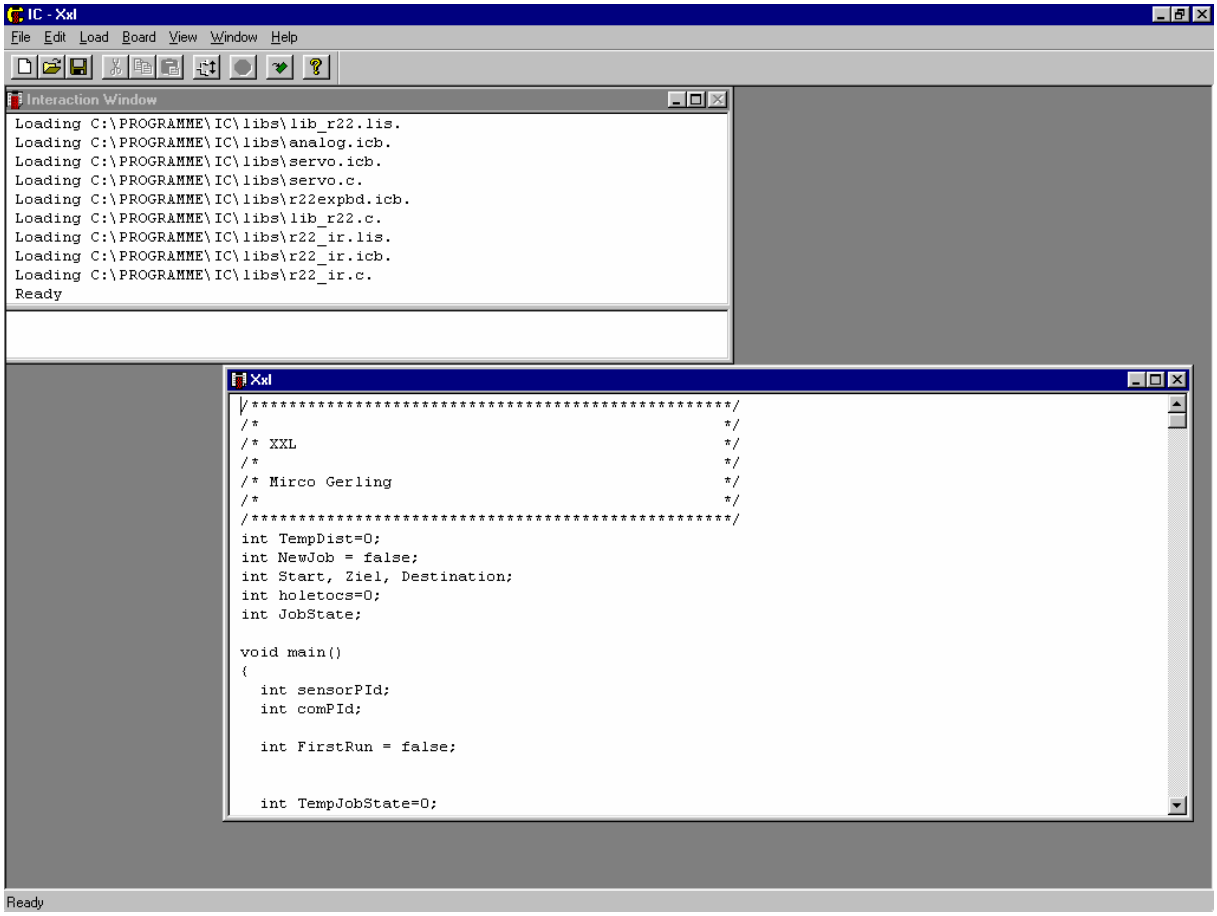
Zum Senden und Auslesen wird folgende Funktion benötigt.

int peek(int loc)

Die Entwicklungsumgebung von Interactive C bietet eine grafische Oberfläche zur Entwicklung und Übertragung von Programmen auf den Roboter. Es gibt ein Interaction Window, in dem Statusmeldungen während der Übertragung von Code zu sehen sind. Im unteren Bereich des Interaction Window gibt es einen kleinen Abschnitt, in dem man direkt Funktionen aufrufen kann und damit die Funktionstüchtigkeit des Roboters überprüfen kann.

Die Entwicklungsumgebung versucht beim Start das Board des Roboters anzusprechen und den PCode für den Grundbetrieb des Boards herunterzuladen. Ist kein Board vorhanden, so kann man in den Einstellungen angeben, dass das Board ignoriert werden soll. So kann man auch ohne Board an dem Code arbeiten.

Weiter Funktionalitäten der Entwicklungsumgebung von Interactive C sind einfache Editorfunktionen wie Find, Replace und Goto Line. Eine Hilfe zu den Funktionen wird ebenfalls angeboten. Die hier verwendete Version der Software ist die Version 3.2 [Newton].



Entwicklungsumgebung Interactive C

6.2 Realisierung des Servers

Auf dem Roboter sind drei Prozesse implementiert. Der erste Prozess ist für die Auftragsabwicklung zuständig. Der zweite Prozess ist für die Kommunikation über die serielle Schnittstelle zuständig. Der dritte Prozess erledigt das Auslesen und Speichern der Sensordaten. Im Folgenden werden die Hungerfunktion und der Sensorprozess erläutert.

6.2.1 Hungerfunktion

Die Hungerfunktion des Roboters soll als Subsumption Architektur realisiert werden. Solange der Roboter keinen Hunger hat, arbeitet er eingehende Aufträge ab. Es werden keinerlei Lichtquellen registriert. Bekommt der Roboter etwas Hunger, so hält er Ausschau nach einer Lichtquelle. Trifft er auf keine Lichtquelle, so werden die Aufträge weiter abgearbeitet. Trifft er auf Licht, so unterbricht er die Auftragsabwicklung ab und fährt zur Lichtquelle. Wenn er großen Hunger bekommt, so unterbricht er die Auftragsabwicklung unabhängig von einer Lichtquelle ab und bleibt stehen und hält Ausschau nach Licht.

Hat er Licht entdeckt, so fährt er im verhaltensbasierten Modus auf die Lichtquelle zu. Die Sensorwerte der Photowiderstände werden direkt auf die Räder gegeben. Hat er genug Energie aufgenommen, so sucht er wieder die Startposition und richtet sich nach dem Beacon aus. Dies ist nötig um wieder in den Automaten der Auftragsabwicklung einzusteigen. Von dort wird der begonnene Auftrag wieder

aufgenommen. Hat er das erste Haus bereits angefahren, so fährt er nur noch das zweite Haus an. Dies bedeutet, dass er nur noch einen halben Auftrag ausführt.

Der Hunger, der in diesem Fall proportional zur gefahrenen Strecke ist, kann auch als eine Art Sensor verstanden werden. Die Funktion, die den Grad des Hungers zurück gibt, sollte ursprünglich regelmäßig im Sensorprozess aufgerufen werden. Das Auslesen der Shaftencoder kostet jedoch relativ viel Rechenzeit. Daher habe ich mich entschieden, die Hungerabfrage immer nur in einem der Häuser zu machen, da ich dort sowieso die zurückgelegte Strecke speichere um in etwa eine Ahnung zu haben, wie weit ich vom Startpunkt entfernt bin. Die Abfrage geschieht also im Prozess der Auftragsabwicklung. Je nach Grad des Hungers wird ein Zustand gesetzt, nachdem in der Auftragsabwicklung entschieden wird, ob der aktuelle Auftrag abgebrochen wird oder nicht. Hat der Roboter starken Hunger, so bleibt er in dieser Realisierung im jeweiligen Haus stehen.

6.2.2 Realisierung der endlichen Automaten

Die Prozesse, die auf dem Roboter laufen, sind als einzelne Automaten implementiert. Sie kommunizieren über globale Variablen, da Interactive C leider keine Prozesskommunikation zu Verfügung stellt. Das Prozessscheduling ist sehr einfach gehalten. Man kann beim Start eines Prozesses eine bestimmte Anzahl von Ticks an Rechenzeit zuweisen und bestimmen wie viel Stack dem Prozess zugewiesen werden soll.

Dies bedeutet, dass z.B. der Sensorprozess sämtliche Sensorwerte permanent ausliest und sie in globalen Variablen speichert. Benötigt der Prozess für die Ablaufsteuerung einen bestimmten Sensorwert, so liest er die globale Variable aus.

Die Automaten können vom Prinzip her als switch-case Verteiler interpretiert werden. Die Verteiler sind jedoch als if-else Konstrukt implementiert, da es in Interactive C keine switch-case Konstrukte gibt.

6.3 Realisierung des Client

Die Anwendungsschicht der Client-Anwendung nimmt die Befehle entgegen, die der Anwender an den Roboter übermitteln möchte. Wenn z.B. der Button „Go!“ gedrückt wird, werden die Start und Ziel Comboboxen ausgelesen. Start und Ziel werden in Strings gespeichert.

```
Private Sub ImpReadControls()
   strStart = Me.cboStart.Text
   strZiel = Me.cboZiel.Text
End Sub
```

Diese Daten werden an die Kommunikationsschicht übergeben, indem die Funktion ImpSendData aufgerufen wird.

```
ImpSendData "S" & strStart & "Z" & strZiel & "X"
```

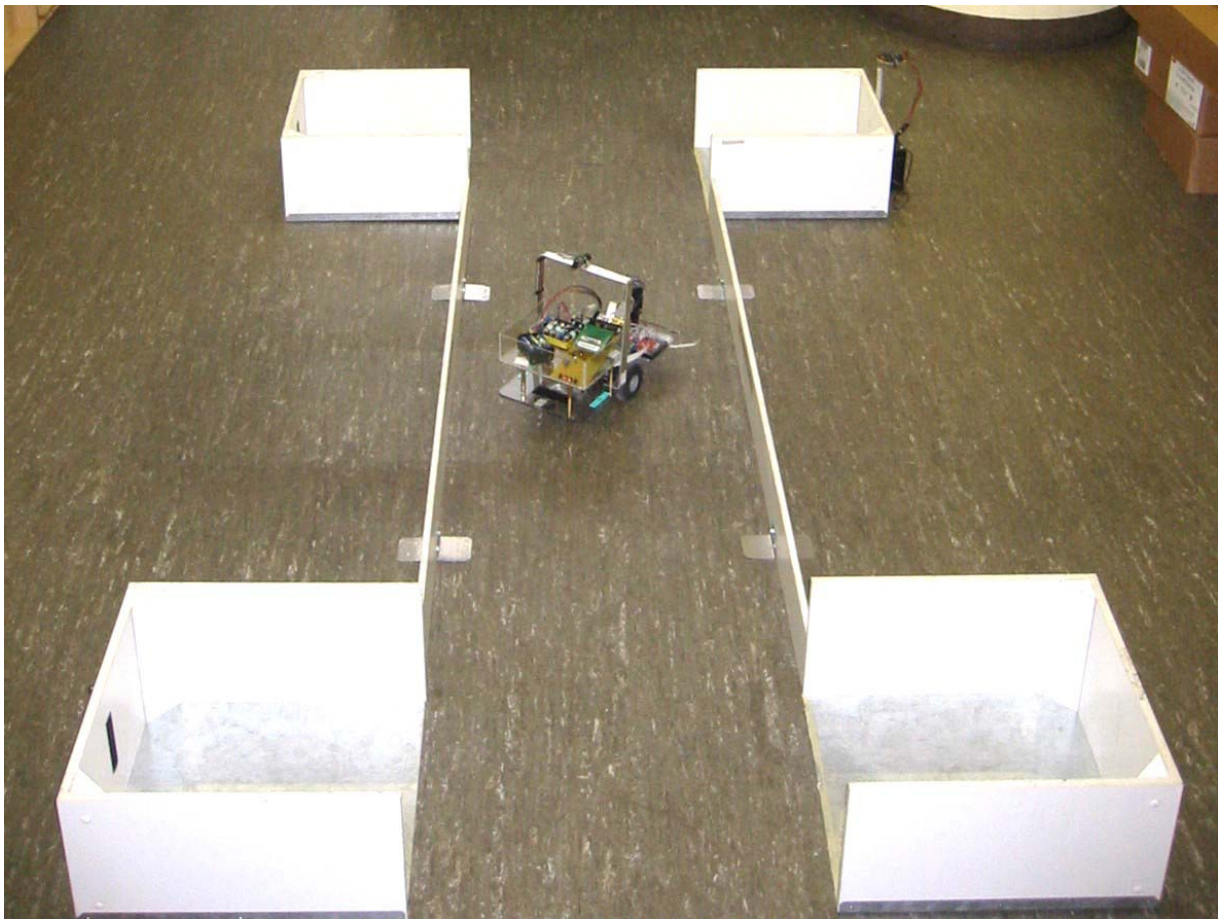
Die Funktion ImpSendData ist mit einer Flusskontrolle kombiniert. Sie sendet immer nur ein weiteres Zeichen, wenn der Roboter mit einem Zeichen geantwortet hat. Antwortet der Roboter nicht, so läuft ein Timer ab und es wird ein Timeout gemeldet.

Weitere Funktionen der Kommunikationsschicht sind die Funktionen *ClosePort()*, *OpenPort(nPort)* und *ImpReceiveData(Buffer)*.

```
Private Sub ClosePort()
   If Me.MSComm.PortOpen = True Then MSComm.PortOpen = False
End Sub

Private Function OpenPort(nPort As Integer) As Boolean
   On Error Resume Next
   ClosePort

   If Me.MSComm.PortOpen = False Then
      MSComm.CommPort = nPort
      MSComm.Settings = "9600,N,8,1"
      MSComm.InputLen = 0
      MSComm.PortOpen = True
   End If
   If Err.Number = 0 Then OpenPort = True
End Function

Private Function ImpReceiveData(Buffer As String) As String
   ImpReceiveData = Buffer & MSComm.Input
End Function
```

6.4 Der Versuchsaufbau

Die Realisierung des Projektes wurde während der gesamten Entwicklung an einem Aufbau getestet. Er besteht aus vier Häusern, die aus Holz gefertigt sind und eine Bodenplatte aus Metall besitzen. Die Trennwände sind ebenfalls aus Holz und haben kleine Aluminium Füße, damit sie nicht umfallen. An einem der Häuser ist der Beacon montiert. Er wird mit einem Bleiakku betrieben.



Roboter in dem Testumfeld

7 Literaturverzeichnis

1 Pioneer : *ActiveMedia Robotics*.
online. - URL www.activrobots.com.
- Zugriffsdatum: 02.12.2002

2 Newton : *Newton Research Labs*.
online. - URL www.newtonlabs.com/ic.
- Zugriffsdatum: 02.12.2002

3 Lego Lego Mindstorms.
online. - URL mindstorms.lego.com.
- Zugriffsdatum: 02.12.2002

4 HAWRobots IKS-project (Integration Kognitiver Systeme).
online. - URL www.informatik.haw-hamburg.de/~robots.
- Zugriffsdatum: 02.12.2002

5 HAWLego Lego.
online. - URL www.informatik.haw-hamburg.de/~lego.
- Zugriffsdatum: 02.12.2002

6 Martin MARTIN, Fred: 6.270: The MIT LEGO Robot Design Competition.
online. - URL www.media.mit.edu/people/fredm/projects/6270.
- Zugriffsdatum: 02.12.2002

7 MIT6.270 6.270 - MIT´s Autonomous Robot Design Competition.
online - URL web.mit.edu/6.270.
- Zugriffsdatum: 02.12.2002

8 HandbuchKI Luck, Kai von: Anwendungen der Künstlichen Intelligenz
online - www.informatik.haw-hamburg.de/~luck/
- Zugriffsdatum: 03.12.2002

9 Arkin R.C. Arkin, Behavior-Based Robotics
MIT Press 1998
(Lehrbuch über Agenten/Robots)

10 Görz G. Görz, C.-R. Rollinger, J. Schneeberger (Hrsg.)
Handbuch der Künstlichen Intelligenz (3rd Ed.)
Oldenbourg Verlag 2000

11 Nilsson Nils J. Nilsson, Artificial Intelligence: A New Synthesis
Morgan Kaufmann 1998

Anhang A – Konfiguration BlueRS+

Modul das die Verbindung aufbaut:

```
BRS0001 V1.107 Jan 08 2002 14:12:41
#show
cim
 cmds: 8-ConnectAlways

Bluetooth
 bname: Stollmann BlueRS+                bpin: ****
 brestr: 0-off bcrypt: 0-off brv: 0-Data link          brad: 0080371622E4
 brsch: 1      brname:                          boad:
 bsname: BlueRS+ serial port

vmoddi
 bsize: 2048   idle: 0

at
 s0: 0        s2: 43       s3: 13       s4: 10       s5: 8        s7: 30
 s91: 0       opt: 5       rcs: 4

serasy
 br: 4-9600    dbits: 8      sbits: 1      prty: 0-none
 cdtr: 0-ignore            cdcd: 1-connected          ccts: 1-on
 cdsr: 0-on    flc: 0-none

ca
 cato: 15                  capa: 3

#
```

Modul auf dem Roboter:

```
BRS0001 V1.107 Jan 08 2002 14:12:41
#show
cim
 cmds: 12-IncomingCallsOnly

Bluetooth
 bname: Stollmann BlueRS+                bpin: ****
 brestr: 0-off bcrypt: 0-off brv: 0-Data link          brad: 0080371503F8
 brsch: 1      brname:                          boad:
 bsname: BlueRS+ serial port

vmoddi
 bsize: 2048   idle: 0

at
 s0: 0        s2: 43       s3: 13       s4: 10       s5: 8        s7: 30
 s91: 0       opt: 5       rcs: 4

serasy
 br: 4-9600    dbits: 8      sbits: 1      prty: 0-none
 cdtr: 0-ignore            cdcd: 1-connected          ccts: 1-on
 cdsr: 0-on    flc: 0-none

ca
 cato: 15                  capa: 3

#
```

Anhang B – Serielle Schnittstelle des 6.270 Boards

Sending/receiving serial from 6.270 board

When IC runs, the 6.270 board's serial port is set for 9600 baud, no parity, 1 stop bit. Normally the serial port is used for communication between the pcode on the board and IC running on the host. However, if you disconnect the board from IC on the host, it is possible for a program on the board to take over the serial port.

Transmittings Characters

Transmitting characters is pretty easy; you basically just poke directly to the 6811's UART hardware. Here's how:

```
void serial_putchar(int c)
{
  while (!(peek(0x102e) & 0x80));  /* wait until serial transmit empty */
  poke(0x102f, c);  /* send character */
}
```

Remember to stop IC (or disconnect the board) before you send characters out, since any characters you send will most likely confuse IC quite a bit.

Also, if you connect the board to a terminal emulator or other program/device which might send characters to the board, be sure to disable the pcode's handling of incoming serial (described in the next section). Otherwise, as soon as you send a character to the board, the board will wedge waiting for the rest of the packet (in the format IC would send). (One sign this is happening is that the heartbeat stops.)

Receiving Characters

Receiving characters is only slightly more hard. You first have to tell the pcode running on the board to not pick up the characters itself. Note that this will disable downloading and other interaction if you hook back up to IC. This means you have to either reset the board or re-enable the pcode serial to get IC to interact with the board.

Here's the code:

```
void disable_pcode_serial()    /* necessary to receive characters using serial_getchar
*/
{
  poke(0x3c, 1);
}
void reenable_pcode_serial()   /* necessary for IC to interact with board again */
{
  poke(0x3c, 0);
}
int serial_getchar(int c)
{
  while (!(peek(0x102e) & 0x20)); /* wait for received character */
  return peek(0x102f);
}
```

Randy Sargent rsargent@media.mit.edu

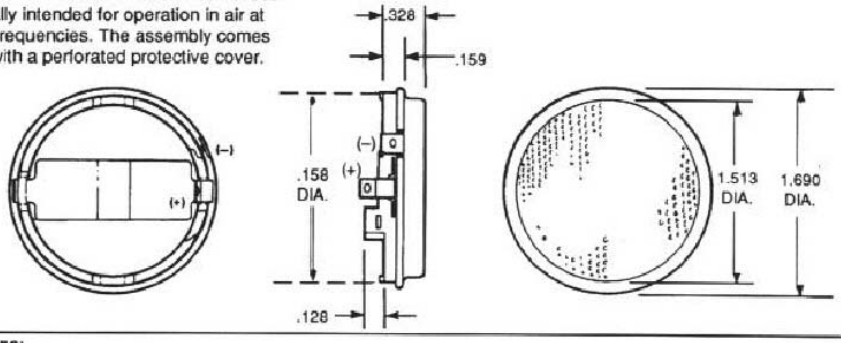
Anhang C – Datenblatt Sonar Polaroid

Technical Specifications for

600 Series
Instrument Grade Electrostatic Transducer

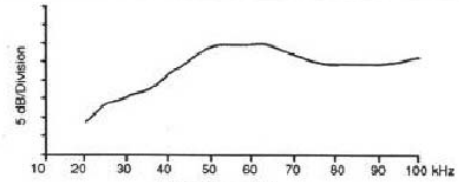
Polaroid

This Instrument Grade electrostatic transducer is specifically intended for operation in air at ultrasonic frequencies. The assembly comes complete with a perforated protective cover.

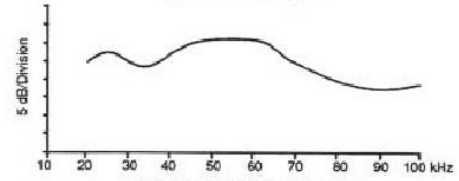


Specifications:

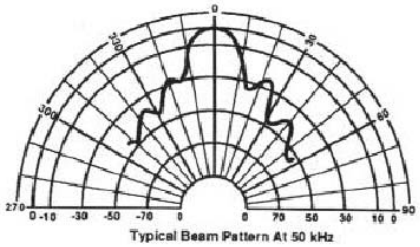
Usable Transmitting Frequency Range	See Graph	Maximum Combined Voltage	400V
Usable Receiving Frequency Range	See Graph	Capacitance at 1 kHz (Typical)	400-500 pf
Beam Pattern	See Graph	150 vdc bias	
Minimum Transmitting Sensitivity at 50 kHz	110 dB	Operating Conditions	
300 vac pk-pk, 150 vdc bias		Temperature	-20°-160°F
(dB re 20 µPa at 1 meter)		Relative Humidity	5%-95%
Minimum Receiving Sensitivity at 50 kHz	–42dB	Standard Finish	
150 vdc bias (dB re lv/Pa)		Foil	Gold
		Housing	Flat Black Cold Roll Steel
Suggested DC Bias Voltage	150V		
Suggested AC Driving Voltage (peak)	150V	Specifications subject to change without notice.	



Typical Transmit Response



Typical Free Field Receive Response



Typical Beam Pattern At 50 kHz

Note: dB normalized to on-axis response.
Note: Curves are representative only. Individual responses may differ.

6/95

Technical Specifications for

Polaroid

6500 Series Sonar Ranging Module

Features:

- Accurate Sonar Ranging from 6" to 35 ft.
- Drives 50 kHz Electrostatic Transducer with No Additional Interface
- Operates from Single Supply
- Accurate Clock Output Provided for External Use
- Selective Echo Exclusion
- TTL-Compatible
- Multiple Measurement Capability
- Uses TI TL851 and Polaroid 614906 Sonar Chips
- Socketed Digital Chip
- Convenient Terminal Connector
- Variable Gain Control Potentiometer

Description:

The 6500 Series is an economical sonar ranging module that can drive all Polaroid electrostatic transducers with no additional interface. This module, with a simple interface, is able to measure distances from 6" to 35 ft. The typical absolute accuracy is ± 1% of the reading over the entire range.

This module has an external blanking input that allows selective echo exclusion for operation on a multiple echo mode. The module is able to differentiate echos from objects that are only 3" apart. The digitally controlled-gain, variable-bandwidth amplifier minimizes noise and side-lobe detection in sonar applications.

The module has an accurate ceramic resonator-controlled 420 kHz time-base generator. An output based on the 420 kHz time base is provided for external use. The sonar transmit output is 16 cycles at a frequency of 49.4 kHz.

The 6500 Series module operates over a supply voltage range from 4.5 volts to 6.8 volts and is characterized for operation from 0° C to 40° C.

Absolute Maximum Ratings:

Voltage from any pin to ground (see Note 1) ..... 7 V
Voltage from any pin except XDCR to Vcc (see Note 1) ..... -7 to 0.5 V
Operating free-air temperature range ..... 0° C to 40° C
Storage temperature range ..... -40° C to 85° C

NOTE 1: The XCDR pin may be driven from -1 volt to 400 volts typical with respect to ground

Specifications subject to change without notice.

Recommended Operating Conditions

		MIN.	MAX.	UNIT
Supply Voltage, Vcc		4.5	6.8	V
High-level Input Voltage, V_{IH}	BLNK, BINH, INIT	2.1		V
Low-level Input Voltage, V_{IL}	BLNK, BINH, INIT		0.6	V
ECHO and OSC Output Voltage			6.8	V
Delay Time, Power Up to INIT High		5		ms
Recycle Period		80	ms	
Operating Free-air Temperature, T_A		0	40	°C

6500 Series Sonar Ranging Module (Cont.)

Electrical Characteristics Over Recommended Ranges of Supply Voltage and Operating Free-Air Temperature (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
Input Current	BLNK, BINH, INIT	V_I = 2.1 V			1	mA
High-level Output Current, I_{OH}	ECHO, OSC	V_{OH} = 5.5 V			100	µA
Low-level Output Voltage, V_{OL}	ECHO, OSC	I_{OL} = 1.6 mA			0.4	V
Transducer Bias Voltage		T_A = 25° C		200		V
Transducer Output Voltage (peak-to-peak)		T_A = 25° C		400		V
Number of Cycles for XDCR Output to Reach 400 V		C = 500 pF			7	
Internal Blanking Interval				2.38†		ms
Frequency During 16-pulse	OSC output			49.4†		kHz
Transmit Period	XMIT output			49.4†		
Frequency After 16-pulse	OSC output			93.3†		kHz
Transmit Period	XMIT output			0		
	During transmit period				2000	mA
Supply Current, I_{CC}	After transmit period				100	

†These typical values apply for a 420 kHz ceramic resonator.

Operation With Polaroid Electrostatic Transducer:

There are two basic modes of operation for the 6500 Series sonar ranging module: single-echo mode and multiple-echo mode. The application of power (Vcc), the activation of the Initiate (INIT) input, and the resulting transmit output, and the use of the Blanking Inhibit (BINH) input are basically the same for either mode of operation. After applying power (Vcc) a minimum of 5 milliseconds must elapse before the INIT input can be taken high. During this time, all internal circuitry is reset and the internal oscillator stabilizes. When INIT is taken high, drive to the Transducer XDCR output occurs. Sixteen pulses at 49.4 kHz with 400-volt amplitude will excite the transducer as transmission occurs. At the end of the 16 transmit pulses, a DC bias of 200 volts will remain on the transducer as recommended for optimum operation.

In order to eliminate ringing of the transducer from being detected as a return signal, the Receive (REC) input of the ranging control IC is inhibited by internal blanking for 2.38 milliseconds after the initiate signal. If a reduced blanking time is desired, then the BINH input can be taken high to end the blanking of the Receive input anytime prior to internal blanking. This may be desirable to detect objects closer than 1.33 ft. corresponding to 2.38 milliseconds and may be done if transducer damping is sufficient so that ringing is not detected as a return signal.

In the single-echo mode of operation (Figure 1), all that must be done next is to wait for the return of the transmitted signal, traveling at approximately 0.9 milliseconds per foot out and back. The returning signal is amplified and appears as a high-logic-level echo output. The time between INIT going high and the Echo (ECHO) output going high is proportional to the distance of the target from the transducer. If desired, the cycle can now be repeated by returning INIT to a low-logic level and then taking it high when the next transmission is desired.

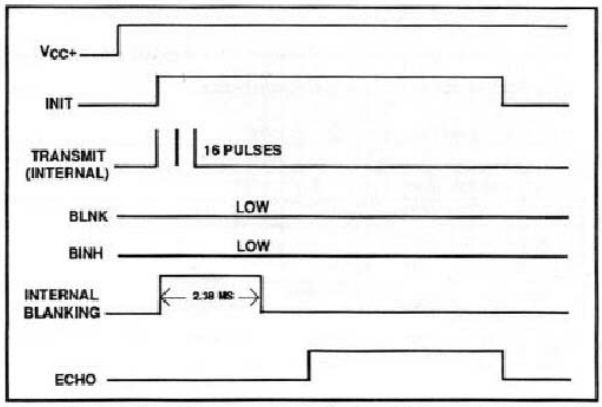
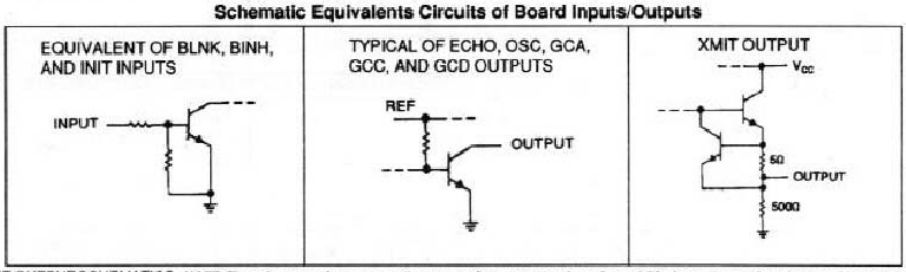


FIGURE 1: EXAMPLE OF A SINGLE-ECHO-MODE CYCLE WITHOUT BLANKING INPUT

6500 Series Sonar Ranging Module (Cont.)

Schematic Equivalents Circuits of Board Inputs/Outputs



INPUT/OUTPUT SCHEMATICS ***NOTE: The echo output is an open collector transistor output and requires a 4.7 k ohm pull up resistor between Vcc and the output.***

If there is more than one target and multiple echos will be detected from a single transmission, then the cycle is slightly different (Figure 2). After receiving the first return signal which causes the ECHO output to go high, the Blanking (BLNK) input must be taken high then back low to reset the ECHO output for the next return signal. The blanking signal must be at least 0.44 milliseconds in duration to account for all 16 returning pulses from the first target and allow for internal delay times. This corresponds to the two targets being 3" apart.

During a cycle starting with INIT going high, the receiver amplifier gain is incremented higher at discrete times (Figure 3) since the transmitted signal is attenuated with distance. At approximately 38 milliseconds, the maximum gain is attained. For this reason, sufficient gain may not be available for objects greater than 35 ft. away. Although gain can be increased by varying R1 (Figure 4), there is a limit to which the gain can be increased for reliable module operation. This will vary from application to application. The modules are "kitted" prior to their final test during manufacture. This is necessary because the desired gain distribution is much narrower than the module gain distribution if all were kitted with one value resistor. As kitted, these modules will perform satisfactorily in most applications. As a rule of thumb, the gain can be increased up to a factor of 4, if required, by increasing R1 correspondingly. Gain is directly proportional to R1.

Potentiometer VR1 (Figure 4) provides an interstage gain adjustment for the module. It can be used to trim the overall range of gain set by fix resistor R1.

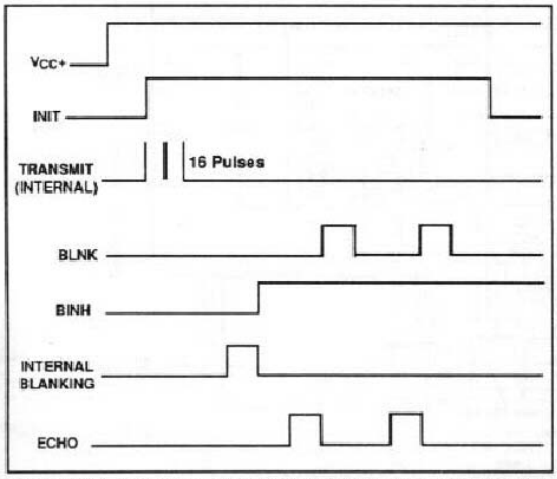


FIGURE 2: EXAMPLE OF A MULTIPLE-ECHO-MODE CYCLE WITH BLANKING INPUT

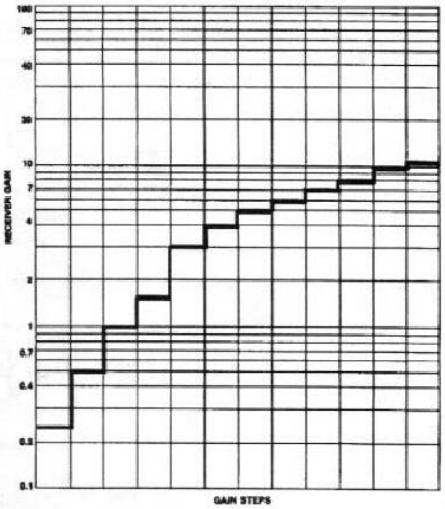


FIGURE 3: RECEIVER GAIN VS GAINSTEP NUMBERS

6500 Series Sonar Ranging Module (Cont.)

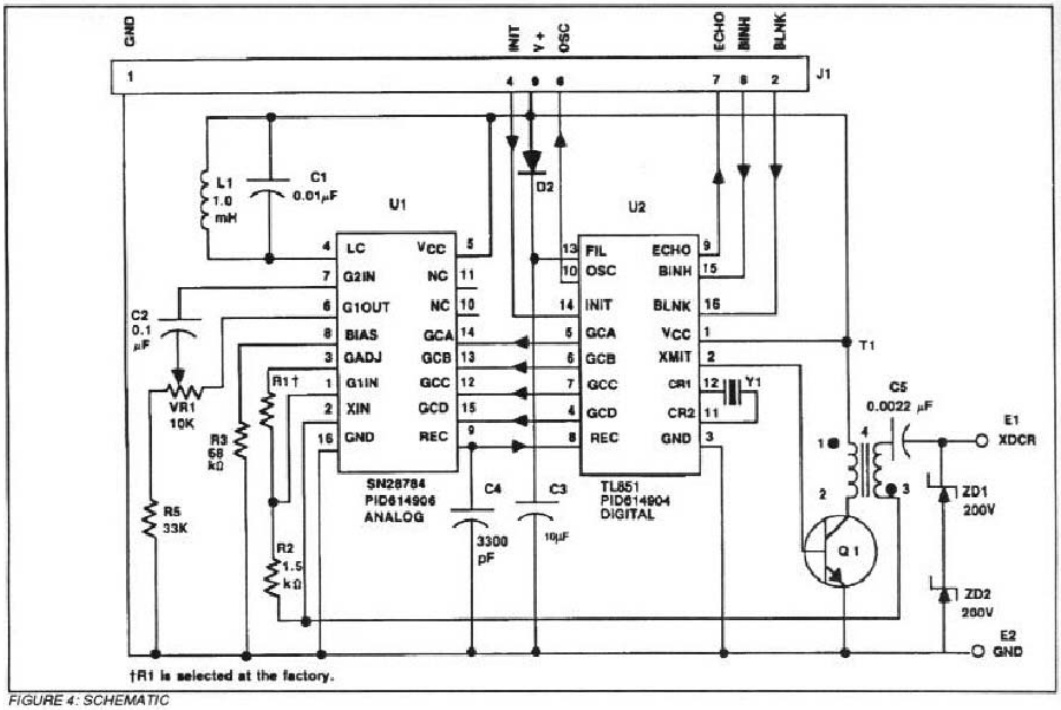


FIGURE 4: SCHEMATIC

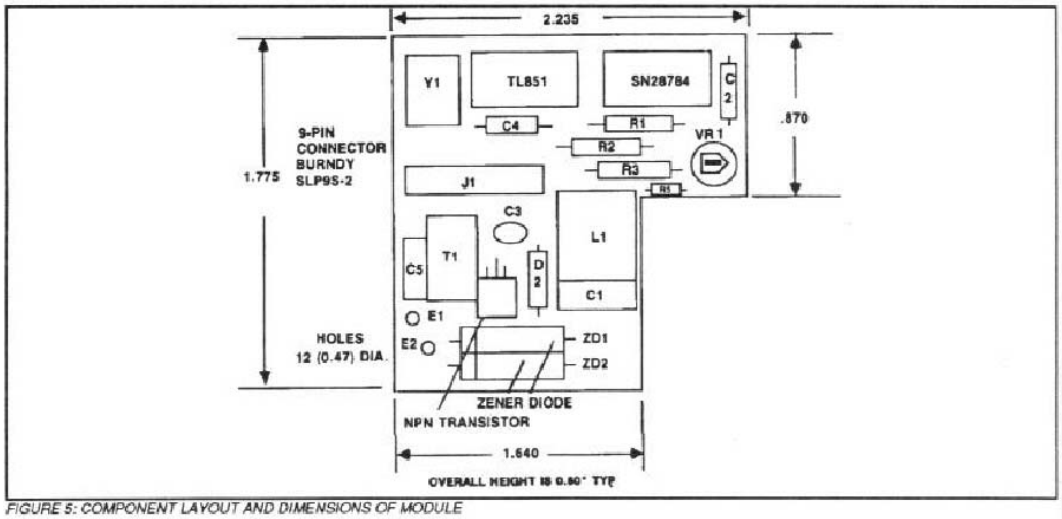


FIGURE 5: COMPONENT LAYOUT AND DIMENSIONS OF MODULE

Anhang D – Datenblatt Sharp Sensor GP2D12

GP2D12/GP2D15

General Purpose Type Distance Measuring Sensors

■ Features

1. Less influence on the color of reflective objects, reflectivity
2. Line-up of distance output/distance judgement type
   Distance output type (analog voltage) : **GP2D12**
   Detecting distance : 10 to 80cm
   Distance judgement type : **GP2D15**
   Judgement distance : 24cm
   (Adjustable within the range of 10 to 80cm)
3. External control circuit is unnecessary
4. Low cost

■ Applications

1. TVs
2. Personal computers
3. Cars
4. Copiers

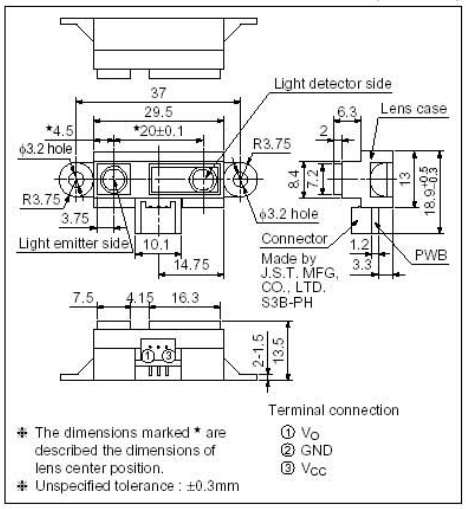
■ Absolute Maximum Ratings

(Ta=25°C, Vcc=5V)

Parameter	Symbol	Rating	Unit
Supply voltage	Vcc	−0.3 to +7	V
Output terminal voltage	Vo	−0.3 to Vcc +0.3	V
Operating temperature	T_{opr}	−10 to +60	°C
Storage temperature	T_{stg}	−40 to +70	°C

■ Outline Dimensions

(Unit : mm)



✣ The dimensions marked * are described the dimensions of lens center position.
✣ Unspecified tolerance : ±0.3mm

■ Recommended Operating Conditions

Parameter	Symbol	Rating	Unit
Operating supply voltage	V_{CC}	4.5 to +5.5	V

■ Electro-optical Characteristics

(Ta=25°C, V_{CC}=5V)

Parameter		Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Distance measuring range		ΔL	*1 *2	10	–	80	cm
Output terminal voltage	GP2D12	V_O	L=80cm *1	0.25	0.4	0.55	V
	GP2D15	V_{OH}	Output voltage at High *1	Vcc –0.3	–	–	V
		V_{OL}	Output voltage at Low *1	–	–	0.6	V
Difference of output voltage	GP2D12	ΔV_O	Output change at L=80cm to 10cm *1	1.75	2.0	2.25	V
Distance characteristics of output	GP2D15	V_O	*1 *2 *4	21	24	27	cm
Average Dissipation current		I_{CC}	L=80cm *1	–	33	50	mA

Note) L : Distance to reflective object.
*1 Using reflective object : White paper (Made by Kodak Co. Ltd. gray cards R-27 - white face, reflective ratio : 90%).
*2 We ship the device after the following adjustment : Output switching distance L=24cm±3cm must be measured by the sensor.
*3 Distance measuring range of the optical sensor system.
*4 Output switching has a hysteresis width. The distance specified by Vo should be the one with which the output L switches to the output H.

Fig.1 Internal Block Diagram

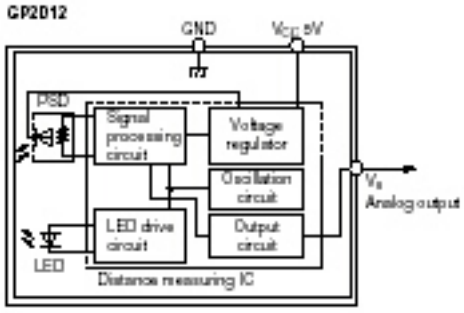


Fig.2 Internal Block Diagram

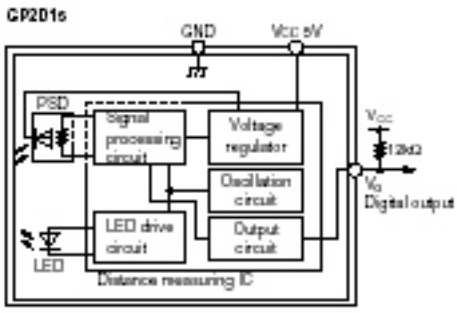


Fig.3 Timing Chart

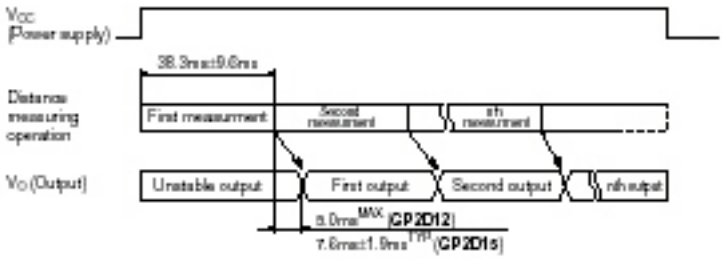


Fig.4 Distance Characteristics

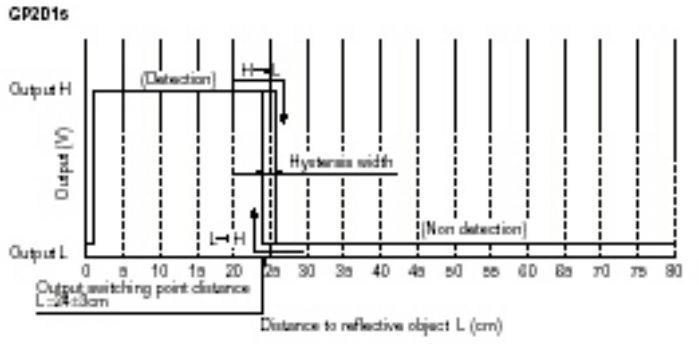


Fig.5 Analog Output Voltage vs. Surface Illuminance of Reflective Object

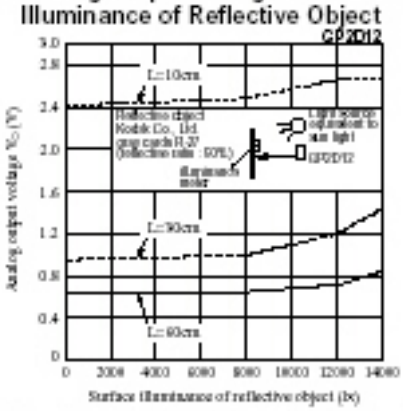


Fig.6 Analog Output Voltage vs. Distance to Reflective Object

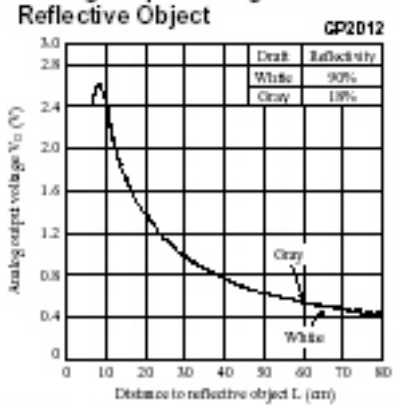


Fig.7 Analog Output Voltage vs. Ambient Temperature

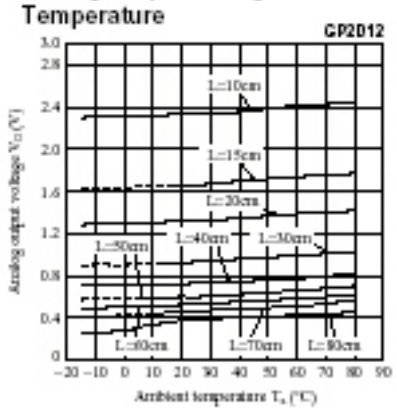
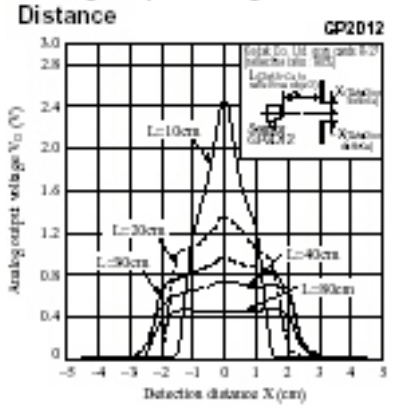


Fig.8 Analog Output Voltage vs. Detection Distance



Application Circuits

NOTICE

- The circuit application examples in this publication are provided to explain representative applications of SHARP devices and are not intended to guarantee any circuit design or license any intellectual property rights. SHARP takes no responsibility for any problems related to any intellectual property right of a third party resulting from the use of SHARP's devices.
- Contact SHARP in order to obtain the latest device specification sheets before using any SHARP device. SHARP reserves the right to make changes in the specifications, characteristics, data, materials, structure, and other contents described herein at any time without notice in order to improve design or reliability. Manufacturing locations are also subject to change without notice.
- Observe the following points when using any devices in this publication. SHARP takes no responsibility for damage caused by improper use of the devices which does not meet the conditions and absolute maximum ratings to be used specified in the relevant specification sheet nor meet the following conditions:

(i) The devices in this publication are designed for use in general electronic equipment designs such as:
— Personal computers
— Office automation equipment
— Telecommunication equipment [terminal]
— Test and measurement equipment
— Industrial control
— Audio visual equipment
— Consumer electronics

(ii) Measures such as fail-safe function and redundant design should be taken to ensure reliability and safety when SHARP devices are used for or in connection with equipment that requires higher reliability such as:
— Transportation control and safety equipment (i.e., aircraft, trains, automobiles, etc.)
— Traffic signals
— Gas leakage sensor breakers
— Alarm equipment
— Various safety devices, etc.

(iii) SHARP devices shall not be used for or in connection with equipment that requires an extremely high level of reliability and safety such as:
— Space applications
— Telecommunication equipment [trunk lines]
— Nuclear power control equipment
— Medical and other life support equipment (e.g., scuba).

- Contact a SHARP representative in advance when intending to use SHARP devices for any "specific" applications other than those recommended by SHARP or when it is unclear which category mentioned above controls the intended use.
- If the SHARP devices listed in this publication fall within the scope of strategic products described in the Foreign Exchange and Foreign Trade Control Law of Japan, it is necessary to obtain approval to export such SHARP devices.
-
- Contact and consult with a SHARP representative if there are any questions about the contents of this publication.

Anhang E – Schaltplan Verstärker

Der Verstärker für die Sharp Sensoren beruht auf einer Schaltung die im Roboter Labor erstmals von Rainer Balzerowski gebaut wurde. Informationen zu der Schaltung gibt es auf den Seiten der HAW Hamburg [HAWLego]. Dort gibt es eine Anleitung, für den Anschluss von 3 Sharpsensoren an einen Eingang eines RCX der Lego Mindstorms.

Der hier verwendete Verstärker ist eine Erweiterung von Gunther Lemm auf vier Eingänge. Das Layout der Platine wurde von Timo Storjohann mit dem Programm Eagle erstellt. Die Platinen wurden zum Ätzen gegeben. Dadurch sind Platinen entstanden, die angenehm zu löten sind.