

1 Zielsetzung

1.1 Ausgangssituation

Als Grundlage für das hier zu beschreibende Praktikum dient der Roboter ARACHNAE I, ein Roboter, der im Rahmen der Vorlesung und Übung "Anwendungen der Mikroprozessortechnik" am Institut für technische Informatik der technischen Universität Wien gebaut wurde. ARACHNAE I ist eine autonome, sechsbeinige, Laufmaschine, die sich über zwei einfache taktile Sensoren (Endschalter) in einer unbekanntem, nicht vorgegeben Umgebung bewegen kann. Dabei kann sie sowohl stationären als auch mobilen Hindernissen, falls sie sich nicht zu schnell bewegen, ausweichen.

Der mechanische Aufbau besteht aus einer 3-Schichtverleimten Holzträgerplatte als 'Rückgrat' und dazu orthogonal montierten, gelochten Aluminiumprofilen, an denen die Beintriebe befestigt sind. Jedes der sechs Beine besitzt zwei Freiheitsgrade (*DOF*), einen maximalen Hub von 3 cm und einen maximalen Vorschub von 8 cm. Beim *Tripod-Gait* bewegt sich der Roboter mit einer Geschwindigkeit von etwa 10 cm/sec vorwärts.

Gesteuert wird der Roboter von insgesamt 3 Prozessorsystemen. Der Hauptprozessor, eine BASIC Stamp II (ein PIC 16C84 mit integrierten BASIC Interpreter), ist für die Sensordatenverarbeitung und Navigation verantwortlich. Die Ergebnisse der Pfadplanung werden über einen seriellen Bus an zwei Subprozessoren gesendet, von denen jeder die Steuerung der drei Beine einer Seite mit jeweils sechs Motoren übernimmt.

1.2 Ziele

Im Rahmen des Praktikums soll, basierend auf den Erkenntnissen und Erfahrungen mit ARACHNAE I, ein autonomer, mobiler, sechsbeiniger Roboter entwickelt und gebaut werden.

1.2.1 Allgemeines

Damit ein Roboter tatsächlich autonom in einer unbekanntem Umgebung operieren kann, lassen sich eine Reihe grundlegender Anforderungen und Eigenschaften, die er haben muß, identifizieren:

- Möglichkeit, auch in rauhen, unebenen Gelände sich kollisionsfrei zu bewegen.
- 'Klettereigenschaften'; Er muß imstande sein, Objekte von wenigstens der halben Körperhöhe überklettern zu können.
- Sowohl die unmittelbare als auch die entfernte Umgebung müssen von Sensoren erfaßt und verarbeitet werden, um zum einen Kollisionen mit nahen Objekten zu verhindern und zum anderen Pfade durch das Terrain planen zu können.
- Er muß eine On-board Stromversorgung besitzen.
- Er muß auch noch mit eingeschränktem Funktionsumfang operabel bleiben, das heißt, der Ausfall eines oder weniger Subsysteme darf nicht zum Versagen des Gesamtsystems führen.

1.2.2 Hardware

Gesteuert wird der Roboter von einem verteilten Rechnernetz, bestehend aus billigen und einfachen Mikroprozessoren, von denen jeder für eine bestimmte Aufgabe im Rahmen des Gesamtsystems zuständig ist.

Um die oben beschriebenen Mindestanforderungen an die Sensorik zu erfüllen, werden taktile Sensoren zur Erfassung von Hindernissen und Objekten im körpernahen Bereich bis zu 10 cm verwendet. Weitere dieser Sensoren befinden sich an den 'Sohlen' der Beine, um Bodenkontakt beziehungsweise das Fehlen eines solchen zu detektieren (das soll verhindern, daß der Roboter in Bodenebenen oder Gruben fällt, die die anderen Sensoren 'übersehen' haben). Die körperferne Umgebung bis zu etwa 1 m vor dem Roboter wird mit Infrarot-Sensoren abgetastet. Zusätzlich sollen sie parallel zu den taktilen Fühlern auch solche Objekte erkennen, die von den feststehenden mechanischen Schaltern nicht erfasst werden (Im Gegensatz zu ihren massiv beweglichen biologischen Pendanten sind die Fühler bei ARACHNAE II starr montiert). Ein horizontal rotierender Ultraschallsensor scannt den Bereich innerhalb 10m um den Roboter, um prädiktive Navigation und Pfadplanung zu ermöglichen.

Um ein reflexives und reaktives Verhalten der Laufmaschine zu gewährleisten, werden weitere Umweltparameter mittels Sensoren ausgewertet. Das sind vor allem Umgebungshelligkeit und Temperatur.

Um gefährliche Neigungen des Geländes unter dem Roboter erkennen zu können, wird permanent die Inklination in der Ebene, das heißt, in der x- und y-Achse (Längsachse und Querachse des Roboters), gemessen. Eine Erfassung der z-Achse (Hochachse) ist derzeit nicht vorgesehen.

1.2.3 Software

Die Software ist so modular aufgebaut und strukturiert, daß sie zum einen leicht um zusätzliche Leistungsmerkmale erweiterbar ist und zum anderen die verschiedenen Schichten der dem Entwurf zugrundeliegenden Subsumption Architecture [12] repräsentiert.

1.2.4 Mechanik

Um das Bewegen und Klettern im Gelände zu ermöglichen, soll jedes Bein drei DOF erhalten. Damit kann der Roboter beliebige Schrittfolgen und Bewegungsabläufe realisieren, was ihm eine extrem hohe Manövrierbarkeit verleiht. Ein 'Kniegelenk' gestattet es jedem Bein zusätzlich, den Fußaufsetzpunkt mindestens auf die Höhe des eigentlichen Roboterkörpers zu verlagern (Damit ist es, zumindest theoretisch, möglich, Objekte von der Höhe des Roboters selbst zu überklettern). Der Korpus selbst ist aus leichten, aber möglichst stabilen Material aufgebaut, um das Verhältnis Eigengewicht zu Nutzlast zu maximieren.

2 Systemeinsatz und Umgebung

Um das Funktionieren des Roboters zu gewährleisten, muß, so paradox es auch klingen mag, die Umwelt, in der er agieren soll, ausreichend komplex und strukturiert sein. Das bedeutet, daß für eine zufriedenstellende Navigation und Pfadplanung hinreichende Informationen über Topologie, Art und Umfang der Umwelt vorhanden sein müssen. Letztlich bedeutet das nichts anderes, als daß geeignete Sensordaten erfaßbar sein müssen, also die Umgebungshelligkeit ausreichend hoch ist, Temperaturunterschiede meßbar sind und/oder genügend Ultraschall reflektierende Objekte und Landmarken vorhanden sind.

3 Funktionale Anforderungen

3.1 Mechanischer Aufbau

3.1.1 Korpus

Der Roboterkorpus besteht aus zwei bodenparallelen, profilierten und verwindungssteifen Kunststoffplatten, die einerseits die gesamte Steuerelektronik, Teile der Sensorik und die Spannungsversorgung tragen und andererseits als Träger für die Beinmechaniken dienen. Der Vorteil dieses Werkstoffs besteht vor allem aus der leichten Bearbeitbarkeit, der hohen Festigkeit und dem geringen Eigengewicht mancher Kunststoffsorten.

Die folgende Abbildung zeigt die mechanischen Abmessungen des Roboters.



Abb.1: Mechanischer Aufbau von ARACHNAE II

3.1.2 Beine

Die Beine bestehen im wesentlichen aus zwei Segmenten, die über ein Gelenk miteinander verbunden sind. Jedes der beiden Segmente ist über einen Servo in der Vertikalen bewegbar. Zusätzlich ist das gesamte Bein über einen dritten Servo in der Horizontalen schwenkbar, wodurch ein kompetter Bewegungsablauf mit "Bein abheben - Bein vorwärts / rückwärtsbewegen - Bein absetzen - Körper vorwärts / rückwärtsbewegen" ermöglicht wird.

Als Material kommen wie beim Korpus nur extrem leichte, aber gleichzeitig harte und verwindungssteife Materialien in Frage. Zum jetzigen Zeitpunkt ist noch keine Festlegung auf einen bestimmten Werkstoff möglich, wobei allerdings die eingeschränkten Bearbeitungsmöglichkeiten für eher exotische Plaste und ähnliches die Alternativen auf Aluminium oder Holzverbunde begrenzen.

Die beiden Segmente für einen Prototyp werden aus 3mm Sperrholzplatte hergestellt.

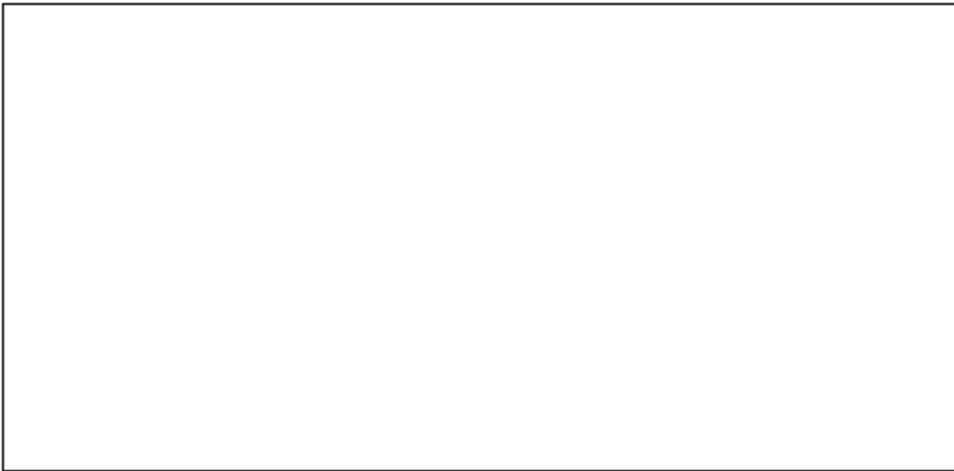


Abb.2: Schematische Darstellung der Beinmechanik

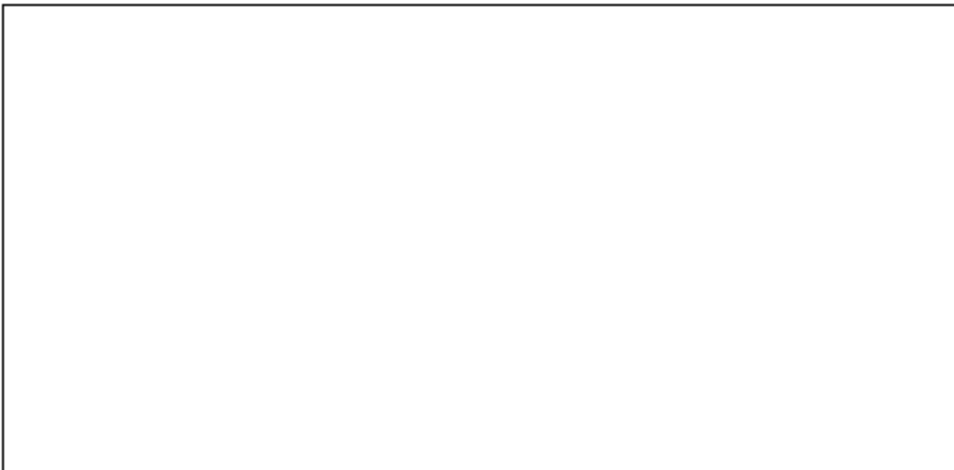


Abb.3: Workspace eines Beins

3.1.3 Servoantriebe

Als Antriebe für die Beine werden handelsübliche Servomotoren verwendet, wie sie vor allem im RC-Modellbau zur Anwendung kommen. Allerdings benötigen nicht alle drei Servos das gleiche Haltemoment. Um vorzeitige Abnutzung und übermäßigen Verschleiß vorzubeugen, werden nur Servos mit gehärteter, kugelgelagerter Antriebswelle sowie Metallgetriebe verwendet.

- ◆ Servo 1: Bei einem erwarteten Gesamtgewicht von etwa 3.5 bis 4 kg des Roboters muß dieser Servo imstande sein, ein Gewicht von zirka 1.5 kg zu tragen. Das gilt für den Tripod Gait, die Gangart, bei der jeweils 3 Beine Bodenkontakt haben und den Roboter tragen, während die drei anderen Beine sich auf ihre jeweils nächsten Fußaufsetzpunkte bewegen, das heißt, drei Beine müssen das Gewicht der Laufmaschine tragen. Alle anderen (stabilen) Gangarten haben immer mehr als drei Beine am Boden. Servo 1 ist den stärksten Belastungsschwankungen ausgesetzt und benötigt deshalb ein hohes Haltemoment. Aus diesem Grund kommt nur eine Ausführung mit mindestens 150Ncm Kraftmoment zum Einsatz.
- ◆ Servo 2: Dieser Antrieb ist für die Vorwärts- beziehungsweise Rückwärtsbewegung des Beines zuständig. Dabei treten insbesondere bei unebenen Gelände auch starke Wechselbelastungen auf, aber erfahrungsgemäß weit geringer als bei Servo 1. Das Haltemoment darf nicht kleiner sein als 60 Ncm.
- ◆ Servo 3: Der Servo, der jenes Beinsegment bewegt, welches den Roboter auf dem Boden abstützt, benötigt ein Haltemoment von nur ungefähr 30 Ncm, da er bloß das Gewicht des Beinsegments selbst tragen muß, während sich das Bein auf der berechneten Trajektorie an den neuen Fußaufsetzpunkt bewegt.



Abb.4: Beinsegmente und Servos

3.2 Hardware

3.2.1 Allgemeines

Dieser Abschnitt beschreibt die Hardwarestruktur des verteilten Rechnernetzes zur Steuerung des Roboters. Sie ist als Mehrrechner-System mit loser Kopplung ausgeführt. Dabei sind die einzelnen Subsysteme über einen seriellen Bus mit RS485/422 Protokoll im Halbduplexbetrieb verbunden. Jedes Rechteck ist als eigenständige Prozessoreinheit mit eigener Peripherie und Anbindung an den seriellen Bus anzusehen und wird ab nun nur mehr als Knoten bezeichnet.

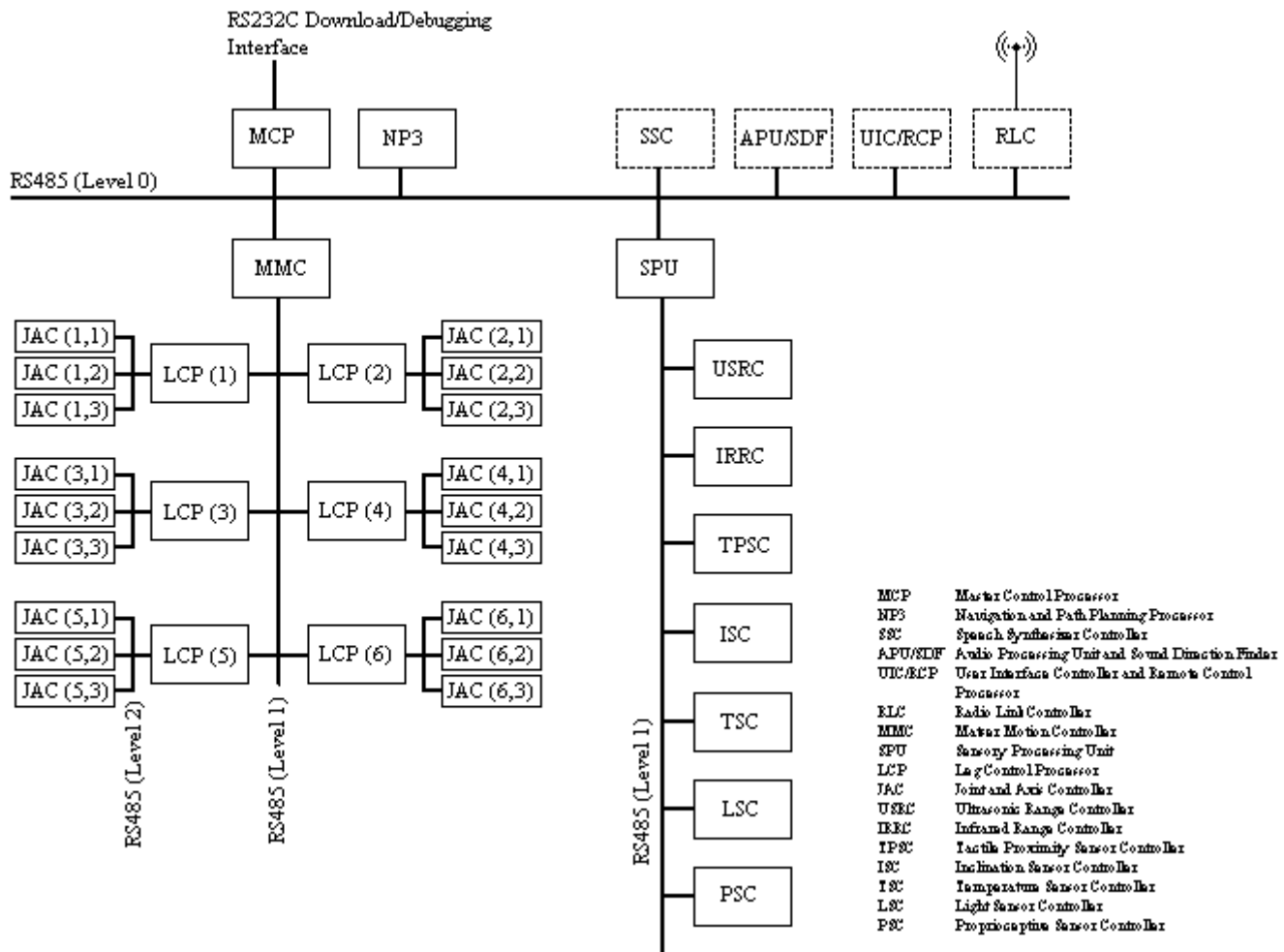


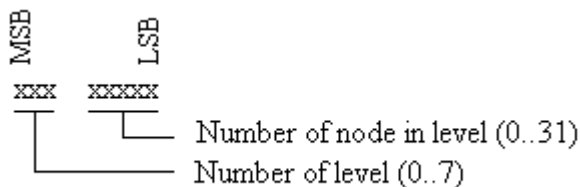
Abb.5: Verteiltes Rechnernetz zur Steuerung von ARACHNAE II

Knotenadressierung

Jeder Knoten hat eine eindeutige 8 Bit lange Adresse (ADR), die natürlich nur einmal im Netz vergeben werden darf und als Konstante im Knotenrechner hardcodet ist.

Die Adressen sind vorgegeben und in den folgenden Abschnitten bei der Beschreibung der jeweiligen Knoten in hexadezimaler Notation angegeben.

Die Konvention für die Vergabe der Adressen ist wie folgt:



Broadcast Address

Die Adresse 255 ist für Broadcasting reserviert und darf nicht als Node Address vergeben werden. Legt der MCP diese Adresse an den Bus, fühlen sich alle Nodes angesprochen. Broadcasting wird

für Messages verwendet, die gleichzeitig an alle Nodes gesendet werden müssen (zum Beispiel: Initiieren von Selbstdiagnose-Routinen in allen Nodes, MCP initiiertes Reset, usw).

Knotenname (kurz)	Knotenname (lang)	Adresse (Hexadezimal)	Adresse (Binär)
MCP	Master Control Processor	0x01	000 00001
NP3	Navigation and Path Planning Processor	0x02	000 00010
MMC	Master Motion Controller	0x03	000 00011
SPU	Sensor Processing Unit	0x04	000 00100
SSC	Speech Synthesizer Controller	0x05	000 00101
APU/SDF	Audio Processing Unit / Sound Direction Finder	0x06	000 00110
UIC/RCP	User Interface Controller / Remote Control Processor	0x07	000 00111
RLC	Radio Link Controller	0x08	000 01000
LCP(1)	Leg Control Processor #1	0x21	001 00001
LCP(2)	Leg Control Processor #2	0x22	001 00010
LCP(3)	Leg Control Processor #3	0x23	001 00011
LCP(4)	Leg Control Processor #4	0x24	001 00100
LCP(5)	Leg Control Processor #5	0x25	001 00101
LCP(6)	Leg Control Processor #6	0x26	001 00110
JAC(1,1)	Joint and Axis Controller Leg #1, Joint #1	0x41	010 00001
JAC(1,2)	Joint and Axis Controller Leg #1, Joint #2	0x42	010 00010
JAC(1,3)	Joint and Axis Controller Leg #1, Joint #3	0x43	010 00011
JAC(2,1)	Joint and Axis Controller Leg #1, Joint #1	0x44	010 00100
JAC(2,2)	Joint and Axis Controller Leg #1, Joint #2	0x45	010 00101
JAC(2,3)	Joint and Axis Controller Leg #1, Joint #3	0x46	010 00110
JAC(3,1)	Joint and Axis Controller Leg #1, Joint #1	0x47	010 00111
JAC(3,2)	Joint and Axis Controller Leg #1, Joint #2	0x48	010 01000
JAC(3,3)	Joint and Axis Controller Leg #1, Joint #3	0x49	010 01001
JAC(4,1)	Joint and Axis Controller Leg #1, Joint #1	0x4A	010 01010
JAC(4,2)	Joint and Axis Controller Leg #1, Joint #2	0x4B	010 01011
JAC(4,3)	Joint and Axis Controller Leg #1, Joint #3	0x4C	010 01100
JAC(5,1)	Joint and Axis Controller Leg #1, Joint #1	0x4D	010 01101
JAC(5,2)	Joint and Axis Controller Leg #1, Joint #2	0x4E	010 01110
JAC(5,3)	Joint and Axis Controller Leg #1, Joint #3	0x4F	010 01111
JAC(6,1)	Joint and Axis Controller Leg #1, Joint #1	0x50	010 10000
JAC(6,2)	Joint and Axis Controller Leg #1, Joint #2	0x51	010 10001
JAC(6,3)	Joint and Axis Controller Leg #1, Joint #3	0x52	010 10010
USRC	Ultrasonic Range Controller	0x27	001 00111
TPSC	Tactile Proximity Sensor Controller	0x28	001 01000
IRRC	Infrared Range Controller	0x29	001 01001
ISC	Inclination Sensor Controller	0x2A	001 01010
TSC	Temperature Sensor Controller	0x2B	001 01011
LSC	Light Sensor Controller	0x2C	001 01100
PSC	Proprioceptive Sensor Controller	0x2D	001 01101
All Nodes		0xFF	111 11111

Serial Bus Layers

Das Netz selber ist in mehrere, voneinander unabhängige Ebenen gegliedert. Der Grund dafür ist, das an einen RS485/422-Bus nur maximal 32 Rechner angeschlossen werden können. Diese Anzahl wird zwar in der aktuell geplanten Ausbaustufe bei weitem nicht erreicht, aber einer eventuellen Erweiterung sind doch sehr rasch Grenzen gesetzt. Ein weiterer Grund für das Layering ist, das ein Rechner, der als Slave am Bus hängt, für einen weiteren, untergeordneten seriellen Bus als Master arbeiten kann. Die an diesen Second-Level-Bus angeschlossenen Subsysteme belasten somit nicht die Bandbreite am First-Level-Bus.

Es erscheint sinnvoll, die prinzipiell eigenständigen Aufgaben Motorsteuerung und Sensordatenverarbeitung busmäßig zu entkoppeln und in zwei voneinander unabhängige Systeme (Second Level Layers) zu konzentrieren. MMC und SPU hängen jeweils als Slave am First-Level-Bus mit dem MCP als Master und arbeiten wiederum für ihre untergeordneten Buses als Master. Eine dritte Busebene existiert bei den einzelnen Beinen. Die sechs LCP's hängen als Slaves am Second-Level-Bus (mit dem MMC als Master) und bedienen ihrerseits als Master einen Third-Level-Bus, an dem jeweils 3 JAC's als Slaves hängen.

3.2.2 MCP (Master Control Processor)

Typ: Level 0 - Knoten
Adresse: 0x01
Status: Notwendig

Der MCP ist ein Koordinationsprozessor, zuständig für Initialisierung und Administration des Netzes und Message Scheduling.

Der MCP führt zur Administration des Netzes eine Tabelle NODES, in der alle existierenden Knoten enthalten sind und als Aktiv oder Inaktiv markiert sind. Zum Zeitpunkt des Einschaltens des Systems sind alle Knoten automatisch als Inaktiv geführt. Erst durch einen Initialisierungsvorgang werden die Knoteneinträge entsprechend modifiziert.

Initialisierung

Bei jedem Systemstart, das heißt, anlegen der Versorgungsspannung oder Reset, sendet der MCP für jeden existierenden Knoten eine Message Identification Request <ID_REQ Addr> über den seriellen Bus. Dabei ist Addr die fest eingestellte, eindeutige Adresse des angesprochenen Knotens. Dieser erkennt nun seinerseits den Request und sendet als Bestätigung seine Adresse an den MCP zurück (ID_ACK Addr). Bekommt der MCP innerhalb von 10 ms kein Acknowledge (ID_ACK Addr) zurück, sendet er noch einmal ein ID_REQ. Kommt dann noch immer kein ACK zurück, wird der Knoten als inaktiv in der Tabelle NODES eingetragen. Abhängig von der Wichtigkeit des Knotens für die Gesamtfunktionalität des Roboters wird dann entweder der Roboter angehalten und eine Fehlermeldung ausgegeben oder der Betrieb mit eingeschränkter Funktionalität aufgenommen. In der hier beschriebenen Ausbaustufe von ARACHNAE II wird lediglich zwischen notwendigen und optionalen Knoten unterschieden, wobei das Fehlen eines einzigen als notwendig deklarierten Knotens bereits zum Stillstand des Systems führt. Die Angabe, ob notwendig oder nicht, ist jeweils bei der Beschreibung der einzelnen Knoten als Status angegeben.

Message Scheduling

Die Hauptaufgabe des MCP ist die des Message Scheduling. Das heißt, alle Daten von einem Knoten (Source) zu einem anderen (Target) laufen immer über den MCP. Der Grund dafür ist das im Vergleich zu einem System mit mehreren Mastern dazu notwendige, wesentlich einfachere Protokoll.

Exemplarischer Ablauf einer Session:

Der Slave-Rechner prüft durch Abfrage seiner Sendeleitung, ob der Bus benutzt wird oder nicht. Ist der Bus frei, sendet er eine Aufforderung (XMIT_REQ) an den MCP und empfängt gleichzeitig im Halbduplexbetrieb diesen Request wieder. Damit kann er feststellen, ob die Daten verfälscht wurden. In diesem Fall sendet er neuerlich einen XMIT_REQ an den MCP. Der MCP selektiert den ersten durchgekommenen Slave und ruft dessen Datensatz ab.

Dieser Record enthält als Parameter die Adresse des gewünschten Zielknotens, also jenes Knotens, der die Daten erhalten soll. Der MCP legt nun seinerseits die Adresse des Zielknotens auf den Bus. Alle Slave-Rechner empfangen diese Adresse und vergleichen sie mit ihrer. Derjenige Slave, der sich durch diese Adresse angesprochen fühlt, empfängt anschließend die Daten vom MCP. Nach Beendigung des Datentransfers wird die Verbindung aufgelöst.

3.2.3 NP3 (Navigation and Path Planning Processor)

Typ: Level 0 - Knoten
Adresse: 0x02
Status: Notwendig

Der NP3 erhält Daten von den entfernungsgebenden Sensoren USRC und IRRC, dem Lichtsensor LSC, dem Temperatursensor TSC und dem Kollisionssensorsystem TPSC. Daraus berechnet er für den jeweils aktuellen Auftrag (Wand folgen, Licht vermeiden, Licht suchen, usw.) den optimalen Pfad. Das Ergebnis wird Positionsdatensatz an den MMC gesendet, der daraus seinerseits entsprechende Steuerkommandos für seine untergeordneten LCP's berechnet.

Initialisierung

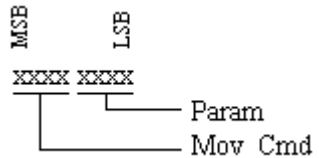
Initialisiert wird der NP3 durch Anlegen der Versorgungsspannung oder durch den MCP. Dabei werden alle Parameter der internen Repräsentation des Weltkoordinatensystems auf Null gesetzt.

Messages vom MCP ® NP3

Message	Beschreibung
0x02 INIT	Initialisierungsrequest vom MCP nur an den NP3.
0xFF INIT	Initialisierungsrequest vom MCP via broadcasting.
0x02 ID_REQ	Der MCP sendet eine Identification Request Message an den NP3.
0x02 STATUS_REQ	Der MCP fordert vom NP3 eine Rückmeldung über seinen derzeitigen internen Status an. Diese Statusmeldung enthält derzeit nur eine Aussage darüber, ob der NP3 gerade in einer Navigationsberechnung ist (BUSY) oder nicht (NOT BUSY).
0x02 DATA Address Value	Der NP3 erhält Daten von den entfernungsgebenden und ambientalen Sensorsystemen. Address ist die Knotenadresse der Datenquelle. Value ist der gemessene oder berechnete Wert, der von diesem Knoten

gesendet wird.

Messages vom NP3 ® MCP

Message	Beschreibung
0x02 ID_ACK	Der NP3 quittiert den Identification Request des MCP durch zurücksenden seiner Knotenadresse.
0x02 STATUS_ACK <i>Status</i>	Rückmeldung des NP3 an den MCP über seinen derzeitigen internen Status. <i>Status:</i> BUSY ... Der NP3 ist momentan mit Navigations- und Pfadplanung beschäftigt. NOT BUSY ... Der NP3 kann Daten senden.
0x02 MOVE <i>Position</i>	Der NP3 berechnet anhand der empfangenen Sensorenwerte einen neuen Pfad und sendet entsprechende Bewegungssteuerungskommandos an den MMC. Der Parameter <i>Position</i> ist folgendermaßen aufgebaut:  Die obersten 4 Bit enthalten Mov_Cmd, den eigentlichen Befehl für die den aktuellen Gegebenheiten angepaßte Sequenz von Beinbewegungen. Param ist ein 4 Bit langer Block, der optionale Parameter für die Bewegungssteuerung beinhalten kann. Dieses Nibble wird in der aktuellen Version von ARACHNAE II nicht verwendet.

Parameter <position> für Bewegungssteuerungskommando MOVE

MOV_CMD	Binärwert	PARAM	Binärwert	Beschreibung
STOP	0000	----	0000	Der Roboter wird angehalten
FORWARD	0001	----	0000	Der Roboter bewegt sich mit der aktuell eingestellten Gangart vorwärts, bis ein neuer Befehl empfangen wird.
BACKWARD	0010	----	0000	Der Roboter bewegt sich mit der aktuell eingestellten Gangart rückwärts, bis ein neuer Befehl empfangen wird.
TURN RIGHT	0011	----	0000	Der Roboterkörper wird nach rechts gedreht, bis ein neuer Befehl empfangen wird.
TURN LEFT	0100	----	0000	Der Roboterkörper dreht sich nach links, bis ein neuer Befehl empfangen wird.
SIDESTEP RIGHT	0101	----	0000	Der Roboter bewegt sich im 'Krabbengang' seitwärts nach rechts, bis ein neuer Befehl empfangen wird.
SIDESTEP LEFT	0110	----	0000	Der Roboter bewegt sich im 'Krabbengang' nach links, bis ein neuer Befehl empfangen wird.

TRIPOD GAIT	0111	----	0000	Die Gangart wird auf Tripod Gait umgestellt. Die aktuelle Bewegungsrichtung des Roboters wird davon nicht beeinflusst.
RIPPLE GAIT	1000	----	0000	Die Gangart wird auf Ripple Gait umgestellt. Die aktuelle Bewegungsrichtung des Roboters wird davon nicht beeinflusst.
WWAVE GAIT	1001	----	0000	Die Gangart wird auf Wave Gait umgestellt. Die aktuelle Bewegungsrichtung des Roboters wird davon nicht beeinflusst.

3.2.4 MMC (Master Motion Controller)

Typ: Level 0 - Knoten
 Adresse: 0x03
 Status: Notwendig

Der MMC erhält Bewegungssteuerungskommandos vom NP3 (s. 3.2.3) und setzt diese seinerseits in entsprechende Steuerbefehle an seine untergeordneten LCP's um.

Initialisierung

Initialisiert wird der MMC durch Anlegen der Versorgungsspannung oder durch den MCP.
 Eine Initialisierungssequenz des MMC erfordert auch eine Reinitialisierung der einzelnen LCP's mit ihren jeweils untergeordneten JAC's (s. 3.2.10).

Messages vom MCP ® MMC

Message	Beschreibung
0x03 INIT	Initialisierungsrequest vom MCP nur an den MMC.
0xFF INIT	Initialisierungsrequest vom MCP via broadcasting.
0x03 ID_REQ	Der MCP sendet eine Identification Request Message an den MMC.
0x03 STATUS_REQ	Der MCP fordert eine Rückmeldung über den aktuellen Status des MMC an. Diese Statusmeldung enthält derzeit nur Informationen über die aktuell eingestellte Gangart und Richtung der Bewegung.
0x03 MOVE Address Position	<p>Der MCP sendet die vom NP3 berechneten Bewegungssteuerungskommandos an den MMC.</p> <p><i>Address</i> ist die Knotenadresse jenes Knotens, von dem dieser Datensatz für die Bewegungssteuerung stammt. In der hier beschriebenen Version von ARACHNAE II kommt als solche Datenquelle nur der NP3 in Frage. Deshalb wird dieser Parameter auch nicht verwendet beziehungsweise berücksichtigt.</p> <p>Der Parameter <i>Position</i> ist folgendermaßen aufgebaut:</p>

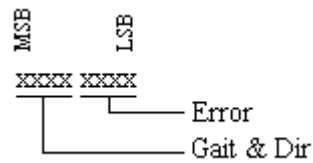
Die obersten 4 Bit enthalten Mov_Cmd, den eigentlichen Befehl für die den aktuellen Gegebenheiten angepaßte Sequenz von Beinbewegungen. Param ist ein 4 Bit langer Block, der optionale Parameter für die Bewegungssteuerung beinhalten kann. Dieses Nibble wird in der aktuellen Version von ARACHNAE II nicht verwendet.

Parameter <position> für Bewegungssteuerungskommando MOVE

POS_CMD	Binärwert	PARAM	Binärwert	Beschreibung
STOP	0000	----	0000	Der Roboter wird angehalten
FORWARD	0001	----	0000	Der Roboter bewegt sich mit der aktuell eingestellten Gangart vorwärts, bis ein neuer Befehl empfangen wird.
BACKWARD	0010	----	0000	Der Roboter bewegt sich mit der aktuell eingestellten Gangart rückwärts, bis ein neuer Befehl empfangen wird.
TURN RIGHT	0011	----	0000	Der Roboterkörper wird nach rechts gedreht, bis ein neuer Befehl empfangen wird.
TURN LEFT	0100	----	0000	Der Roboterkörper dreht sich nach links, bis ein neuer Befehl empfangen wird.
SIDESTEP RIGHT	0101	----	0000	Der Roboter bewegt sich im 'Krabbengang' seitwärts nach rechts, bis ein neuer Befehl empfangen wird.
SIDESTEP LEFT	0110	----	0000	Der Roboter bewegt sich im 'Krabbengang' nach links, bis ein neuer Befehl empfangen wird.
TRIPOD GAIT	0111	----	0000	Die Gangart wird auf Tripod Gait umgestellt. Die aktuelle Bewegungsrichtung des Roboters wird davon nicht beeinflusst.
RIPPLE GAIT	1000	----	0000	Die Gangart wird auf Ripple Gait umgestellt. Die aktuelle Bewegungsrichtung des Roboters wird davon nicht beeinflusst.
WAVE GAIT	1001	----	0000	Die Gangart wird auf Wave Gait umgestellt. Die aktuelle Bewegungsrichtung des Roboters wird davon nicht beeinflusst.

Messages vom MMC ® MCP

Message	Beschreibung
0x03 ID_ACK	Der MMC quittiert den Identification Request des MCP durch zurücksenden seiner Knotenadresse.
0x03 STATUS_ACK <i>Status</i>	Statusrückmeldung vom MMC an den MCP, mit Information über die aktuell eingestellte Gangart und die Richtung der Bewegung. Der Parameter <i>Status</i> ist folgendermaßen aufgebaut:



Gait & Dir:

- TRIPOD
- RIPPLE
- WAVE
- STOP
- FORWARD
- BACKWARD
- TURN RIGHT
- TURN LEFT
- SIDESTEP RIGHT
- SIDESTEP LEFT

Error:

Wird zurückgesendet, wenn im MMC selber oder in seinen Slaves ein Problem auftritt. (Beinantriebe blockieren oder fallen aus, JAC's oder LCP's gehen Out-Of-Order).

Parameter ERROR in STATUS_ACK

Wert (Binär)	Beschreibung
0000	Kein Fehler aufgetreten.
0001	Beinantrieb blockiert
0010	Beinantrieb ausgefallen
0011	JAC Out-Of-Order
0100	LCP Out-Of-Order

3.2.5 SPU (Sensor Processing Unit)

Typ: Level 0 - Knoten
 Adresse: 0x04
 Status: Notwendig

Die SPU ist der Leitreechner für alle Knoten, die Sensordaten erfassen und auswerten. Sie administriert als Master für den Second-Level Bus die daran angeschlossenen Knotenrechner und führt eine Vorverarbeitung der von diesen Subprozessoren gesendeten Sensordaten durch.

Initialisierung

Initialisiert wird die SPU durch Anlegen der Versorgungsspannung oder durch den MCP.

Messages vom MCP ® SPU

Message	Beschreibung
0x04 INIT	Initialisierungsrequest vom MCP nur an die SPU.
0xFF INIT	Initialisierungsrequest vom MCP via broadcasting.
0x04 ID_REQU	Der MCP sendet eine Identification Request Message an die SPU.
0x04 STATUS_REQ	Der MCP fordert eine Rückmeldung über den aktuellen Status der SPU an. Derzeit sollen nur die beiden Stati BUSY und NOT BUSY implementiert werden.

Messages von SPU ® MCP

Message	Beschreibung
0x04 ID_ACK	Die SPU quittiert den Identification Request des MCP durch zurücksenden ihrer Knotenadresse.
0x04 STATUS_ACK <i>Status</i>	Statusrückmeldung von der SPU an den MCP. Derzeit sind für <i>Status</i> nur die beiden Messages BUSY und NOT BUSY in Verwendung. NOT BUSY ... Die SPU ist für Kommunikation mit MCP frei. BUSY ... Die SPU ist mit Sensordatenverarbeitung beschäftigt.

3.2.6 SSC (Speech Synthesizer Controller)

Typ: Level 0 - Knoten
Adresse: 0x05
Status: Optional

Der SSC ist ein Knoten, der einen einfachen und billigen Sprachsynthesizer (Phonemsynthesizer) zur sprachlichen Ausgabe von Alarm- und Statusmeldungen beziehungsweise ganz allgemein für eine natürlichere Mensch-Maschine-Kommunikation steuert.

Initialisierung

Initialisiert wird der SSC durch Anlegen der Versorgungsspannung oder durch den MCP.

Messages vom MCP ® SSC

Message	Beschreibung
0x05 INIT	Initialisierungsrequest vom MCP nur an den SSC.
0xFF INIT	Initialisierungsrequest vom MCP via broadcasting.
0x05 ID_REQU	Der MCP sendet eine Identification Request Message an den SSC.
0x05 STATUS_REQ	Der MCP fordert eine Rückmeldung über den aktuellen Status des SSC an. Derzeit sollen nur die beiden Stati BUSY und NOT BUSY implementiert werden.

<p>0x05 SETUP <i>Data</i></p>	<p>Der verwendete Speechsynthesizer SSI263 von Silicon Systems erlaubt eine Parametrisierung der Sprachausgabe hinsichtlich der Sprechgeschwindigkeit, Lautstärke, Tonhöhe und Artikulation. Eine detaillierte Beschreibung des Parametersatzes <i>Data</i> ist bei der Schaltung des SSC angegeben.</p>
<p>0x05 SPEAK <i>Repeat</i> <i>Length</i> <i>String</i></p>	<p>Der MCP sendet einen String an den SSC, der mit den aktuellen Einstellungen des Speech Synthesizers ausgegeben werden soll. Neben der freien Angabe von nahezu beliebigen Texten können mit der Angabe der <i>Length=0</i> vordefinierte, oft verwendete Messages ausgegeben werden (s. folgende Tabelle). Das reduziert die Busbelastung auf dem First-Level-Bus.</p> <p><i>Repeat:</i> Dieser Parameter erlaubt die Angabe einer Wiederholrate, das heißt, wie oft der gesendete Text ausgegeben werden soll. 0: Der Text wird solange wiederholt ausgegeben, bis ein neuer String gesendet wird. 1..255: Anzahl der maximal möglichen Wiederholungen.</p> <p><i>Length:</i> Die Länge des folgenden Strings. Bereich: 1..255</p> <p><i>String:</i> Textstring mit <i>Length</i> Characters, die vom Betriebssystem des SSC als Phoneme interpretiert werden.</p>

SSC-residente Messages

Wird im Command <SPEAK 0x05 Length String> der Parameter Length auf 0 gesetzt, wird die Nummer im darauffolgenden Parameter String als Index auf eine im SSC vordefinierte Message interpretiert, die dann ausgegeben wird.

Da der Speech Synthesizer seine amerikanische Herkunft nicht verleugnen kann, sind auch bei sorgfältiger Auswahl der Phoneme alle Sprachausgaben eindeutig englisch 'gefärbt'. Deshalb sind diese fix vorgegebenen Messages englischsprachig.

Index	Message
1	Malfunction
2	Performing initialization
3	User input required
4	Enter command
5	Obstacle detected
6	Performing Environment Scan
7	Scanning finished
8	Leg blocked
9	JAC out of order
10	LCP out of order

11	
12	
13	
14	
15	
16	

Messages von SSC ® MCP

Message	Beschreibung
0x05 ID_ACK	Der SSC quittiert den Identification Request des MCP durch zurücksenden seiner Knotenadresse.
0x05 STATUS_ACK <i>Status</i>	Statusrückmeldung vom SSC an den MCP. <i>Status:</i> NOT BUSY ... Der SSC ist für Kommunikation mit MCP frei. BUSY ... Der SSC ist gerade mit Sprachausgabe beschäftigt.

3.2.7 APU/SDF (Audio Processing Unit / Sound Direction Finder)

Typ: Level 0 - Knoten
 Adresse: 0x06
 Status: Optional

Die APU/SDF ist ein Knoten für Sprachsignalverarbeitung. In der vorliegenden Version von ARACHNAE II wird nur der SDF-Teil implementiert, der eine einfache Erkennung der Richtung eines Schallereignisses ermöglicht.

Initialisierung

Initialisiert wird die APU/SDF durch Anlegen der Versorgungsspannung oder durch den MCP.

Messages vom MCP ® APU/SDF

Message	Beschreibung
0x06 INIT	Initialisierungsrequest vom MCP nur an die APU/SDF.
0xFF INIT	Initialisierungsrequest vom MCP via broadcasting.
0x06 ID_REQU	Der MCP sendet eine Identification Request Message an die APU/SDF.
0x06 STATUS_REQ	Der MCP fordert eine Rückmeldung über den aktuellen Status der APU/SDF an. Derzeit sollen nur die beiden Stati BUSY und NOT BUSY implementiert werden.

Messages von APU/SDF ® MCP

Message	Beschreibung
0x06 ID_ACK	Die APU/SDF quittiert den Identification Request des MCP durch zurücksenden ihrer Knotenadresse.
0x06 STATUS_ACK <i>Status</i>	Statusrückmeldung von der APU/SDF an den MCP. <i>Status:</i> NOT BUSY ... Die APU/SDF ist für Kommunikation mit MCP frei. BUSY ... Die APU/SDF ist gerade mit Audio Processing beschäftigt.

3.2.8 UIC/RCP (User Interface Controller / Remote Control Processor)

Typ: Level 0 - Knoten
 Adresse: 0x07
 Status: Optional

Der UIC/RCP ist ein optionaler Knotenrechner, der ein einfaches User Interface, bestehend aus Schaltern zur direkten Steuerung des Roboters, LEDs für Statusanzeigen, einem Piezosummer für akustische Alarme und einem LCD zur visuellen Ausgabe von Status- und Fehlermeldungen, implementiert. Der RCP-Teil ist ein Standard RS232C-Interface, an das eine dedizierte Fernbedienung zur manuellen Kontrolle des Roboters angeschlossen werden kann.

Initialisierung

Initialisiert wird der UIC/RCP durch Anlegen der Versorgungsspannung oder durch den MCP.

Messages vom MCP ® UIC/RCP

Message	Beschreibung
0x07 INIT	Initialisierungsrequest vom MCP nur an den UIC/RCP.
0xFF INIT	Initialisierungsrequest vom MCP via broadcasting.
0x07 ID_REQU	Der MCP sendet eine Identification Request Message an den UIC/RCP.
0x07 STATUS_REQ	Der MCP fordert eine Rückmeldung über den aktuellen Status des UIC/RCP an. Derzeit sollen nur die beiden Stati BUSY und NOT BUSY implementiert werden.

Messages von UIC/RCP ® MCP

Message	Beschreibung
0x07 ID_ACK	Der UIC/RCP quittiert den Identification Request des MCP durch zurücksenden seiner Knotenadresse.
0x07 STATUS_ACK <i>Status</i>	Statusrückmeldung vom UIC/RCP an den MCP. <i>Status:</i> NOT BUSY ... Der UIC/RCP ist für Kommunikation mit MCP frei. BUSY ... Der UIC/RCP ist momentan beschäftigt.

3.2.9 RLC (Radio Link Controller)

Typ: Level 0 - Knoten
Adresse: 0x08
Status: Optional

Der RLC ist ein Knotenrechner, der ein Mini-Funkmodem zur drahtlosen Datenkommunikation mit einem Hostrechner oder anderen Robotern steuert. Das Funkmodem ist eine kommerzielle fertige Baugruppe, die besonders klein und leicht ausgeführt ist.

Initialisierung

Initialisiert wird der RLC durch Anlegen der Versorgungsspannung oder durch den MCP.

Messages vom MCP ® RLC

Message	Beschreibung
0x08 INIT	Initialisierungsrequest vom MCP nur an den RLC.
0xFF INIT	Initialisierungsrequest vom MCP via broadcasting.
0x08 ID_REQU	Der MCP sendet eine Identification Request Message an den RLC.
0x08 STATUS_REQ	Der MCP fordert eine Rückmeldung über den aktuellen Status des RLC an. Derzeit sollen nur die beiden Stati BUSY und NOT BUSY implementiert werden.

Messages von RLC ® MCP

Message	Beschreibung
0x08 ID_ACK	Der RLC quittiert den Identification Request des MCP durch zurücksenden seiner Knotenadresse.
0x08 STATUS_ACK <i>Status</i>	Statusrückmeldung vom RLC an den MCP. <i>Status:</i> NOT BUSY ... Der RLC ist für Kommunikation mit MCP frei. BUSY ... Der RLC ist momentan mit Datenkommunikation beschäftigt.

3.2.10 LCP(n) (Leg Control Processor #n)

Typ: Level 1 - Knoten
Adresse: 0x21 ... Bein 1
 0x22 ... Bein 2
 0x23 ... Bein 3
 0x24 ... Bein 4
 0x25 ... Bein 5
 0x26 ... Bein 6

Status: Notwendig

Der LCP ist ein Knotenrechner, der die Steuerung jeweils eines Beines mit seinen drei Motoren übernimmt. Die dazu notwendigen Daten für die Beinbewegungen erhält er vom MMC. Der LCP übersetzt diese Vorgaben in entsprechende Steuersequenzen für die Antriebsservos der Beinsegmente. Zusätzlich führt er eine lokale Sensordatenverarbeitung bezüglich Kollision eines Beins mit Objekten und Hindernissen entlang der Trajektorie, Übertemperaturen der Motoren und Bodenkontakt des Beins durch. Das ist zwar ein Bruch mit der Konvention, daß die gesamte Sensordatenverarbeitung von entsprechenden Knoten im SPU-Netz vorgenommen wird, die aber im Sinne einer zwingend notwendigen, möglichst schnellen Reaktion auf taktile Reize im Workspace der Beine in Kauf genommen wird.

Initialisierung

Initialisiert werden die LCP's durch Anlegen der Versorgungsspannung oder durch den MCP.

Messages vom MMC ® LCP

Message	Beschreibung
0x## INIT 0xFF INIT	Initialisierungsrequest vom MCP nur an den LCP mit der Adresse ##. ## Bein 1: 0x21 Bein 2: 0x22 Bein 3: 0x23 Bein 4: 0x24 Bein 5: 0x25 Bein 6: 0x26 Initialisierungsrequest vom MCP via broadcasting.
0x## ID_REQU	Der MMC sendet eine Identification Request Message an den LCP.
0x## STATUS_REQ	Der MMC fordert eine Rückmeldung über den aktuellen Status des LCP an. Derzeit sollen nur die beiden Stati IN_ORDER und OUT_OF_ORDER implementiert werden.

Messages von LCP ® MMC

Message	Beschreibung
0x## ID_ACK	Der adressierte LCP quittiert den Identification Request des MCP durch zurücksenden seiner Knotenadresse.
0x## STATUS_ACK Status	Statusrückmeldung vom LCP an den MMC. <i>Status:</i> IN_ORDER ... Der LCP ist online und kann Daten vom MMC empfangen. OUT_OF_ORDER ... Der LCP kann 'sein' Bein nicht mehr steuern (blockiert, Motor oder JAC ausgefallen).

3.2.11 USRC (Ultrasonic Range Controller)

Typ: Level 1 - Knoten
 Adresse: 0x27
 Status: Notwendig

Der USRC ist in der hier beschriebenen Version von ARACHNAE II der Knotenrechner, der für den Scan der körperfernen Umgebung des Roboters zuständig ist. Dazu steuert er einen 40kHz-Ultrasonic Transducer, der auf einer um 360° in der Horizontalen schwenkbaren Trägerplattform befestigt ist. Die Winkelauflösung beträgt dabei mindestens 1°. Die Verarbeitungsleistung des gesamten Subsystems muß so hoch sein, daß ein kompletter 360°-Scan bei einer maximalen, geade noch meßbaren Objektentfernung, nicht länger als 8 Sekunden dauert.

Initialisierung

Initialisiert wird der USRC durch Anlegen der Versorgungsspannung oder durch den MCP.

Messages vom SPU ® USRC

Message	Beschreibung
0x27 INIT	Initialisierungsrequest vom MCP nur an den USRC.
0xFF INIT	Initialisierungsrequest vom MCP via broadcasting.
0x27 ID_REQU	Der MCP sendet eine Identification Request Message an den USRC.
0x27 STATUS_REQ	Der MCP fordert eine Rückmeldung über den aktuellen Status des USRC an. Derzeit sollen nur die beiden Stati IN_ORDER und OUT_OF_ORDER implementiert werden.
0x27 HEAD_SET_ABS <i>Angle</i>	Kommando an den USRC, den Sensorkopf an die Winkelposition <i>Angle</i> zu drehen
0x27 HEAD_SET_REL <i>Angle</i>	Kommando an den USRC, den Sensorkopf relativ zur aktuellen Position um den Winkel <i>Angle</i> zu drehen.
0x27 HEAD_STEP	Der USRC dreht um 1° an die nächste Position.
0x27 DO_MEASURE	Kommando an den USRC, eine Messung durchzuführen
0x27 FREE_RUN	Der USRC läuft im Free Running Mode.
0x27 XMIT_DATA	Die SPU erwartet Daten vom USRC.

Messages von USRC ® SPU

Message	Beschreibung
0x27 ID_ACK	Der USRC quittiert den Identification Request des MCP durch zurücksenden seiner Knotenadresse.
0x27 STATUS_ACK <i>Status</i>	Der USRC sendet seinen internen Status an die SPU. Derzeit implementierte Statusmeldungen betreffen die Zuverlässigkeit des Meßsystems. Bei einer Winkelposition werden immer drei Messungen durchgeführt und gemittelt. Liegen die drei Meßwerte über einem bestimmten Limit zu weit auseinander, werden nocheinmal drei Messungen durchgeführt, um etwaige Fehlmessungen durch schnell bewegende Objekte im Meßraum zu minimieren. Liegen die Meßwerte erneut außerhalb des

	<p>erlaubten Fehlerbereichs, deklariert sich das Meßsystem selbständig als nicht mehr zuverlässig.</p> <p><i>Status:</i> UNRELIABLE ... Zuverlässigkeit der Meßwerte sind nicht mehr garantiert OUT_OF_ORDER ... Meßsystem setzt sich selbst außer Betrieb IN_ORDER ... Der USRC ist online und kann Daten vom MMC empfangen.</p>
0x27 DATA_RDY	Der USRC sendet eine Kommunikationsaufforderung an die SPU.
0x27 XMIT_DATA Data	<p>Der USRC sendet die zuletzt gemessene Entfernung an die SPU. <i>data</i> ist ein 8 Bit Wort mit folgender Zuordnung der Meßwerte: 00000000 \cong 0 cm 11111111 \cong 10 m Die Auflösung des US-Systems ist zirka 4 cm, was insbesondere für den Einsatz als Weitbereichsscanner völlig ausreichend erscheint.</p>

3.2.12 TPSC (Tactile Proximity Sensor Controller)

Typ: Level 1 - Knoten
 Adresse: 0x28
 Status: Notwendig

Der TPSC ist ein Knotenrechner, der für die Abfrage der beiden im Kopfbereich des Roboters montierten, taktilen Kollisionssensoren zuständig ist. In der hier beschriebenen Version von ARACHNAE II sind diese Sensoren als einfache Endschalter ausgeführt, die bei Kontakt mit einem Objekt ein Signal erzeugen. Der TPSC kann anhand der Signale (rechts, links oder beide) eine grobe Abschätzung der Lage des Objekts oder Hindernisses durchführen.

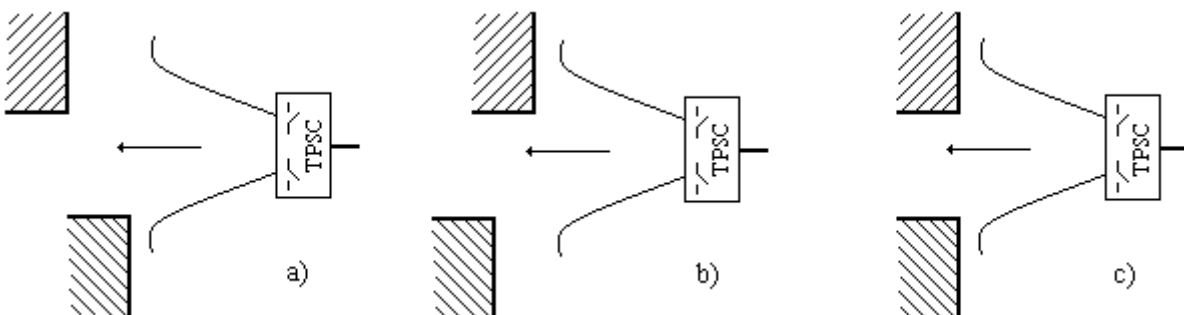


Abb.xx: a) Linker Sensor erfaßt Objekt zuerst und generiert Signal OBJ_LEFT
 b) Rechter Sensor erfaßt Objekt zuerst und generiert Signal OBJ_RIGHT
 c) Beide Sensoren erfassen gleichzeitig das Hinderniss. Der TPSC generiert das Signal OBJ_BOTH

Der Eintritt in eine Kommunikation wird immer vom TPSC initiiert.

Initialisierung

Initialisiert wird der TPSC durch Anlegen der Versorgungsspannung oder durch den MCP.

Messages vom SPU ® TPSC

Message	Beschreibung
0x28 INIT	Initialisierungsrequest vom MCP nur an den TPSC.
0xFF INIT	Initialisierungsrequest vom MCP via broadcasting.
0x28 ID_REQU	Der MCP sendet eine Identification Request Message an den TPSC.
0x28 STATUS_REQ	Der MCP fordert eine Rückmeldung über den aktuellen Status des TPSC an. Derzeit sollen nur die beiden Stati IN_ORDER und OUT_OF_ORDER implementiert werden.
0x28 XMIT_DATA	Die SPU erwartet Daten vom TPSC

Messages von TPSC ® SPU

Message	Beschreibung
0x28 ID_ACK	Der TPSC quittiert den Identification Request des MCP durch zurücksenden seiner Knotenadresse.
0x28 STATUS_ACK <i>Status</i>	Der TPSC sendet seinen internen Status an die SPU. Derzeit sind keine Stati implementiert, das heißt, es wird immer der Status IN_ORDER zurückgesendet.
0x28 OBJ_LEFT	Der TPSC detektiert eine Kollision mit einem Objekt am linken Whisker
0x28 OBJ_RIGHT	Der TPSC detektiert eine Kollision mit einem Objekt am rechten Whisker
0x28 OBJ_BOTH	Der TPSC detektiert eine Kollision mit einem Objekt an beiden Whiskers
0x28 DATA_RDY	Der TPSC sendet eine Kommunikationsanforderung an die SPU
0x28 XMIT data	Der TPSC sendet die Kollisionsdaten an die SPU. <i>data ist:</i> 0001 0000 OBJ_LEFT 0010 0000 OBJ_RIGHT 0100 0000 OBJ_BOTH Die vier niederwertigsten Bits sind für zukünftige Erweiterungen reserviert.

3.2.13 IRRC (Infrared Range Controller)

Typ: Level 1 - Knoten
Adresse: 0x29
Status: Notwendig

Der IRRC ist ein Knotenrechner, der in Ergänzung zum Weitbereichsscanner USRC den körpernahen Bereich bis etwa 10 cm vor dem Roboter abtastet. Da Ultraschall in diesem Bereich keine genügend hohe Auflösung mehr gestattet, ist dieser entfernungsgebende Sensor als Infrarot-Meßsystem ausgeführt.

Es kommen drei Systeme zum Einsatz, wovon eines in den Raum entlang der Längsachse des Roboters strahlt, während die beiden anderen jeweils um 15° zur Längsachse versetzt abstrahlen.

Initialisierung

Initialisiert wird der IRRC durch Anlegen der Versorgungsspannung oder durch den MCP.

Messages vom SPU ® IRRC

Message	Beschreibung
0x29 INIT	Initialisierungsrequest vom MCP nur an den IRRC.
0xFF INIT	Initialisierungsrequest vom MCP via broadcasting.
0x29 ID_REQU	Der MCP sendet eine Identification Request Message an den IRRC.
0x29 STATUS_REQ	Der MCP fordert eine Rückmeldung über den aktuellen Status des IRRC an. Derzeit sollen nur die beiden Stati IN_ORDER und OUT_OF_ORDER implementiert werden.
0x29 DO_MEASURE	Kommando an den IRRC, eine Messung durchzuführen.
0x29 FREE_RUN	Der IRRC läuft im Free Running Mode.
0x29 SET_ALARM limit	Der IRRC soll bei Unterschreiten einer Entfernung <i>limit</i> von sich aus eine Meldung an die SPU absetzen. limit: 0..10 cm
0x29 XMIT_DATA	Die SPU erwartet Daten vom IRRC.

Messages von IRRC ® SPU

Message	Beschreibung
0x29 ID_ACK	Der IRRC quittiert den Identification Request des MCP durch zurücksenden seiner Knotenadresse.
0x29 STATUS_ACK <i>Status</i>	Der IRRC sendet seinen internen Status an die SPU. Derzeit implementierte Statusmeldungen betreffen die Zuverlässigkeit des Meßsystems. Es werden immer drei Messungen durchgeführt und gemittelt. Liegen die drei Meßwerte über einem bestimmten Limit zu weit auseinander, werden noch einmal drei Messungen durchgeführt, um etwaige Fehlmessungen durch schnell bewegende oder IR-permeable Objekte im Meßraum zu minimieren. Liegen die Meßwerte erneut außerhalb des erlaubten Fehlerbereichs, deklariert sich das Meßsystem selbständig als nicht mehr zuverlässig. <i>Status:</i> UNRELIABLE ... Zuverlässigkeit der Meßwerte sind nicht mehr garantiert OUT_OF_ORDER ... Meßsystem setzt sich selbst außer Betrieb IN_ORDER ... Der IRRC ist online und kann Daten vom MMC empfangen.
0x29 DATA_RDY	Der IRRC sendet eine Kommunikationsaufforderung an die SPU.
0x29 XMIT_DATA <i>Data</i>	Der IRRC sendet die zuletzt gemessene Entfernung an die SPU. <i>Data</i> ist ein 8 Bit Wort mit folgender Zuordnung der Meßwerte: 00000000 \cong 0 cm 11111111 \cong 10 cm Die Auflösung des IR-Scanners liegt damit theoretisch bei zirka 0.4 mm, was allerdings in der Praxis sicher nicht erreichbar ist. Aufgrund bisheriger Erfahrungen wird die maximal erreichbare Auflösung bei etwa 0.5 .. 1cm

liegen.

3.2.14 ISC (Inclination Sensor Controller)

Typ: Level 1 - Knoten
 Adresse: 0x2A
 Status: Notwendig

Der ISC ist ein Knotenrechner, der laufen die Neigung des Roboterkörpers entlang der Längs- und der Querachse mißt. Da kommerzielle Inklinationssensoren sowohl sehr teuer als auch größtenteils für den Einsatz in kleinen Robotern zu schwer sind, kommt hier ein selbst gebauter Sensor zum Einsatz. Die Konstruktion beruht auf einem Vorschlag von Daniel Rogantis. Die Auflösung ist mit $\pm 3^\circ$ bei einer maximal erfaßbaren Neigung von 60° zwar sehr gering, genügt aber noch immer den Erfordernissen. Für die Erfassung der Inklination der beiden wichtigsten Roboterachsen werden zwei Sensoren benötigt.

Initialisierung

Initialisiert wird der ISC durch Anlegen der Versorgungsspannung oder durch den MCP.

Messages vom SPU ® ISC

Message	Beschreibung
0x2A INIT	Initialisierungsrequest vom MCP nur an den ISC.
0xFF INIT	Initialisierungsrequest vom MCP via broadcasting.
0x2A ID_REQU	Der MCP sendet eine Identification Request Message an den ISC.
0x2A STATUS_REQ	Der MCP fordert eine Rückmeldung über den aktuellen Status des ISC an. Derzeit sollen nur die beiden Stati IN_ORDER und OUT_OF_ORDER implementiert werden.
0x2A SET_ALARM <i>Channel</i> <i>Limit</i>	Der ISC soll automatisch bei Überschreiten einer bestimmten maximalen Inklination <i>limit</i> am Sensor <i>channel</i> eine Message an die SPU generieren. <i>Channel:</i> 0 = Längsachse 1 = Querachse <i>Limit:</i> -60° .. +60°
0x2A XMIT_DATA	Die SPU erwartet Daten vom ISC

Messages von ISC ® SPU

Message	Beschreibung
0x2A ID_ACK	Der ISC quittiert den Identification Request des MCP durch zurücksenden seiner Knotenadresse.
0x2A STATUS_ACK	Statusrückmeldung vom ISC an die SPU.

<i>Status</i>	<p><i>Status:</i> IN_ORDER ... Der ISC ist online und kann Daten vom MMC empfangen. OUT_OF_ORDER ... Beim Kalibrieren ist ein Fehler aufgetreten; der ISC setzt sich selbst außer Betrieb.</p>
0x2A DATA_RDY	Der IS sendet eine Kommunikationsanforderung an die SPU.
0x2A XMIT_DATA <i>Data</i>	Der ISC sendet den aktuellen Meßwert <i>Data</i> des Inklinationsensors an die SPU.

3.2.15 TSC (Temperature Sensor Controller)

Typ: Level 1 - Knoten
 Adresse: 0x2B
 Status: Notwendig

Der TSC mißt mit Halbleiter-Temperatursensoren (z.Bsp: SAS965) bis zu 8 Umgebungstemperaturen. Die Auflösung des Meßsystems ist 10 mV / °C. Der Meßbereich selber ist auf -20°C .. 100°C beschränkt, was für die allgemeine Betriebspraxis des Roboters allerdings keine wirkliche Behinderung darstellt.

Die gemessene Temperatur wird wegen der (zumindest theoretisch meßbaren) möglichen Minustemperaturen in Zweierkomplementdarstellung auf Aufforderung durch den MCP gesendet. Damit ist der darstellbare Bereich der Werte auf -127 .. 128 eingeschränkt und es ergibt sich folgende Zuordnung der Meßwerte:

Temperatur [°C]	Wert
-20	11101100
0	00000000
100	01100100

Initialisierung

Initialisiert wird der TSC durch Anlegen der Versorgungsspannung oder durch den MCP.

Messages vom SPU ® TSC

Message	Beschreibung
0x2B INIT	Initialisierungsrequest vom MCP nur an den TSC.
0xFF INIT	Initialisierungsrequest vom MCP via broadcasting.
0x2B ID_REQU	Der MCP sendet eine Identification Request Message an den TSC.
0x2B STATUS_REQ	Der MCP fordert eine Rückmeldung über den aktuellen Status des TSC an. Derzeit sollen nur die beiden Stati IN_ORDER und OUT_OF_ORDER implementiert werden.
0x2B SET_ALARM <i>Channel</i> <i>Limit</i>	Der TSC soll automatisch bei Überschreiten einer bestimmten maximalen Temperatur <i>Limit</i> am Meßpunkt <i>Channel</i> eine Message an die SPU generieren.

	<i>Channel:</i> 0..7 <i>Limit:</i> 0..63°
0x2B XMIT_DATA	Die SPU erwartet Daten vom TSC

Messages von TSC ® SPU

Message	Beschreibung
0x2B ID_ACK	Der TSC quittiert den Identification Request des MCP durch zurücksenden seiner Knotenadresse.
0x2B STATUS_ACK <i>Status</i>	Statusrückmeldung vom TSC an die SPU. <i>Status:</i> IN_ORDER ... Der TSC ist online und kann Daten vom MMC empfangen. OUT_OF_ORDER ... Der TSC setzt sich selbst außer Betrieb.
0x2B DATA_RDY	Der TSC sendet eine Kommunikationsanforderung an die SPU.
0x2B XMIT_DATA channel data	Der TSC sendet die zuletzt gemessenen 8 Temperaturwerte an die SPU. channel: 0..7 data: -20..100°

3.2.16 LSC (Light Sensor Controller)

Typ: Level 1 - Knoten
 Adresse: 0x2C
 Status: Notwendig

Der LSC mißt laufend die Umgebungshelligkeit und sendet bei Aufforderung durch den MCP den Mittelwert der drei letzten Messungen an diesen zurück.
 Damit auch Helligkeitsunterschiede in sehr hellen und in dunklen Räumen erfaßt werden können, wird von einem logarithmischen Verstärker eine Spannung erzeugt, die proportional ist zum Logarithmus der von einer Phototransistorschaltung erzeugten Spannung.
 Die Ansprechschwelle, bei der der LSC von sich aus einen Interrupt generiert, ist in einem Bereich von 0..255 einstellbar. Das entspricht einem Meßbereich von 0.1 (sehr dunkel) bis 1000 (sehr hell).

Lichtstärke	Ausgabe
0.1	0
1	64

10	128
100	192
1000	255

Initialisierung

Initialisiert wird der LSC durch Anlegen der Versorgungsspannung oder durch den MCP.

Messages vom SPU ® LSC

Message	Beschreibung
0x2C INIT	Initialisierungsrequest vom MCP nur an den LSC.
0xFF INIT	Initialisierungsrequest vom MCP via broadcasting.
0x2C ID_REQU	Der MCP sendet eine Identification Request Message an den LSC.
0x2C STATUS_REQ	Der MCP fordert eine Rückmeldung über den aktuellen Status des LSC an. Derzeit sollen nur die beiden Stati IN_ORDER und OUT_OF_ORDER implementiert werden.
0x2C SET_ALARM <i>Limit</i>	Der LSC soll automatisch bei Überschreiten einer bestimmten maximalen Leuchstärke <i>Limit</i> eine Message an die SPU generieren. <i>Limit:</i> 1..1000
0x2C XMIT_DATA	Die SPU erwartet Daten vom LSC

Messages von LSC ® SPU

Message	Beschreibung
0x2C ID_ACK	Der LSC quittiert den Identification Request des MCP durch zurücksenden seiner Knotenadresse.
0x2C STATUS_ACK <i>Status</i>	Statusrückmeldung vom LSC an die SPU. <i>Status:</i> IN_ORDER ... Der LSC ist online und kann Daten vom MMC empfangen. OUT_OF_ORDER ... Der LSC setzt sich selbst außer Betrieb.
0x2C DATA_RDY	Der LSC sendet eine Kommunikationsanforderung an die SPU.
0x2C XMIT_DATA <i>Data</i>	Der LSC sendet den Mittelwert <i>Data</i> der drei zuletzt gemessenen Leuchtstärkewerte an die SPU. <i>Data:</i> 0.1..1000

3.2.17 PSC (Proprioceptive Sensor Controller)

Typ: Level 1 - Knoten
 Adresse: 0x2D

Status: Optional

Der PSC überwacht die wichtigsten inneren Zustände des Roboters. In der hier beschriebenen Ausbaustufe von ARACHNAE II wird nur die Höhe der Batteriespannung gemessen. Fällt während des Betriebs des Roboters die Versorgungsspannung unter einen definierten Grenzwert, wird er angehalten und eine akustische und optische Warnung (intermittierender Piepston mit dem Piezosummer und blinkende LED auf dem MCP) ausgegeben.

Initialisierung

Initialisiert wird der PSC durch Anlegen der Versorgungsspannung oder durch den MCP.

Messages vom SPU ® PSC

Message	Beschreibung
0x2D INIT	Initialisierungsrequest vom MCP nur an den PSC.
0xFF INIT	Initialisierungsrequest vom MCP via broadcasting.
0x2D ID_REQU	Der MCP sendet eine Identification Request Message an den PSC.
0x2D STATUS_REQ	Der MCP fordert eine Rückmeldung über den aktuellen Status des PSC an. Derzeit sollen nur die beiden Stati IN_ORDER und OUT_OF_ORDER implementiert werden.
0x2D SET_ALARM <i>Limit</i>	Der PSC soll automatisch bei Unterschreiten einer bestimmten minimalen Betriebsspannung <i>Limit</i> eine Message an die SPU generieren. <i>Limit:</i> 4.2V .. 5.0V
0x2D XMIT_DATA	Die SPU erwartet Daten vom PSC

Messages von PSC ® SPU

Message	Beschreibung
0x2D ID_ACK	Der PSC quittiert den Identification Request des MCP durch zurücksenden seiner Knotenadresse.
0x2D STATUS_ACK <i>Status</i>	Statusrückmeldung vom PSC an die SPU. <i>Status:</i> IN_ORDER ... Der PSC ist online und kann Daten vom MMC empfangen. OUT_OF_ORDER ... Der PSC setzt sich selbst außer Betrieb.
0x2D DATA_RDY	Der PSC sendet eine Kommunikationsanforderung an die SPU.
0x2D XMIT_DATA <i>Data</i>	Der PSC sendet den Mittelwert <i>Data</i> der drei zuletzt gemessenen Spannungswerte an die SPU. <i>Data:</i> 4.2V .. 5.0V

3.2.18 JAC (Joint and Axis Controller)

Typ:	Level 2 - Knoten
Adresse:	0x41 Bein 1, Gelenk 1
	0x42 Bein 1, Gelenk 2
	0x43 Bein 1, Gelenk 3
	0x44 Bein 2, Gelenk 1
	0x45 Bein 2, Gelenk 2
	0x46 Bein 2, Gelenk 3
	0x47 Bein 3, Gelenk 1
	0x48 Bein 3, Gelenk 2
	0x49 Bein 3, Gelenk 3
	0x4A Bein 4, Gelenk 1
	0x4B Bein 4, Gelenk 2
	0x4C Bein 4, Gelenk 3
	0x4D Bein 5, Gelenk 1
	0x4E Bein 5, Gelenk 2
	0x4F Bein 5, Gelenk 3
	0x50 Bein 6, Gelenk 1
	0x51 Bein 6, Gelenk 2
	0x52 Bein 6, Gelenk 3
Status:	Notwendig

Der JAC ist ein Knotenrechner, der für die Steuerung jeweils eines Antriebsservos in einem Bein zuständig ist. Gleichzeitig mißt er zum einen über einen Positionssensor (ein Potentiometer) die aktuelle Lage des Beinsegments und zum anderen mit einem Tachogenerator die aktuelle Winkelgeschwindigkeit des angetriebenen Gelenks.

Initialisierung

Initialisiert werden die JAC's durch Anlegen der Versorgungsspannung oder durch den MCP.

Messages vom LCP ® JAC

Message	Beschreibung
0x## INIT	Initialisierungsrequest vom MCP nur an den JAC mit der Adresse ##.
0xFF INIT	Initialisierungsrequest vom MCP via broadcasting.
0x## ID_REQU	Der MCP sendet eine Identification Request Message an den JAC.
0x## STATUS_REQ	Der MCP fordert eine Rückmeldung über den aktuellen Status des JAC an. Derzeit sollen nur die beiden Stati IN_ORDER und OUT_OF_ORDER implementiert werden.

Messages von JAC ® LCP

Message	Beschreibung
0x## ID_ACK	Der adressierte JAC quittiert den Identification Request des MCP durch zurücksenden seiner Knotenadresse.

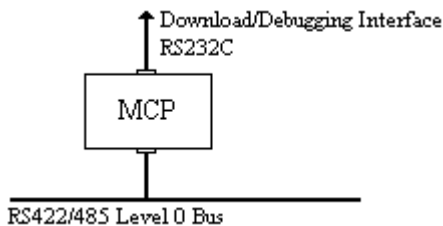
0x## STATUS_ACK <i>Status</i>	Statusrückmeldung vom JAC ## an den MCP. <i>Status:</i> IN_ORDER ... Der PSC ist online und kann Daten vom MMC empfangen. OUT_OF_ORDER ... Der PSC setzt sich selbst außer Betrieb.
----------------------------------	--

3.3 Software

First-Level-Bus

Die Abschnitte 3.3.1 bis 3.3.8 beschreiben alle jene Subsysteme, die am First-Level-Bus arbeiten. Diese implementieren derzeit die oberste Abstraktionsebene innerhalb der realisierten Subsumption Architecture.

3.3.1 MCP (Master Control Processor)



Initialisierung des Netzes

Administration des Netzes

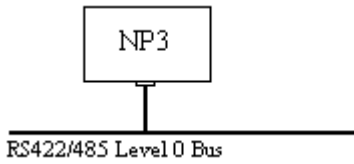
Der MCP verwaltet eine Tabelle NODETAB, die alle im Netz existierenden Knoten enthält und sie als Aktiv oder als Inaktiv markiert. Das heißt, diese Tabelle hat Einträge für alle möglichen Knoten, wie sie in Abschnitt 3.2.1 beschrieben sind. Als Aktiv markiert wird ein Knoten während der Initialisierung des Netzes, wenn er auch tatsächlich physisch vorhanden ist und mit dem MCP über das serielle Bussystem kommunizieren kann. Inaktiv wird er demzufolge genau dann, wenn er entweder physisch nicht vorhanden ist oder er zwar am Bus angeschlossen ist aber aus irgendeinem Grund nicht in eine Kommunikation mit dem MCP eintreten kann.

Aufbau der Tabelle NODETAB

Knotenname:	String(4)	NAME
Adresse:	Byte	ADDRESS
Status:	Boolean	STATUS

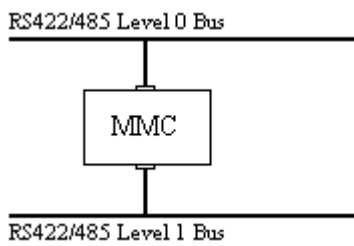
Message Scheduling

3.3.2 NP3 (Navigation and Path Planning Processor)

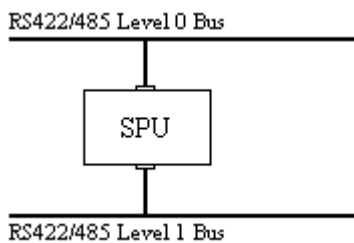


Generieren, modifizieren und adaptieren einer Topologischen Map

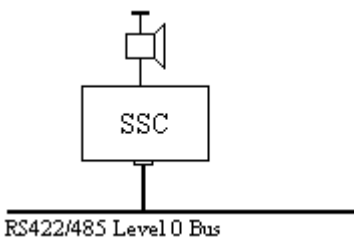
3.3.3 MMC (Master Motion Controller)



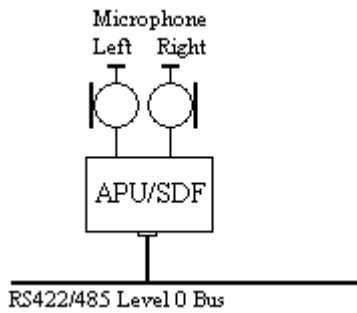
3.3.4 SPU (Sensor Processing Unit)



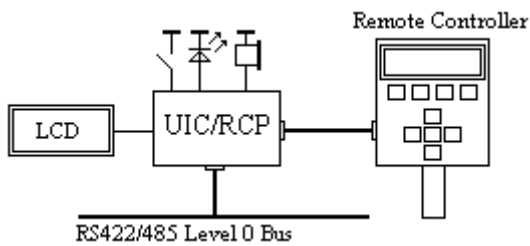
3.3.5 SSC (Speech Synthesizer Controller)



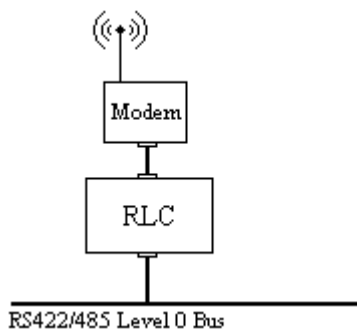
3.3.6 APU/SDF (Audio Processing Unit / Sound Direction Finder)



3.3.7 UIC/RCP (User Interface Controller / Remote Control Processor)



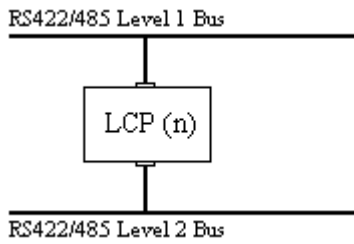
3.3.8 RLC (Radio Link Controller)



MMC Second-Level-Bus

Der nächste Abschnitt beschreibt das LCP-Subsystem, das als Slave am Second-Level-Bus des MMC angeschlossen ist. Als unabhängig arbeitendes Rechnernetz ist es für alle Bewegungsabläufe, Beinsteuerungen und lokale Sensordatenverarbeitung zuständig.

3.3.9 LCP(n) (Leg Control Processor)

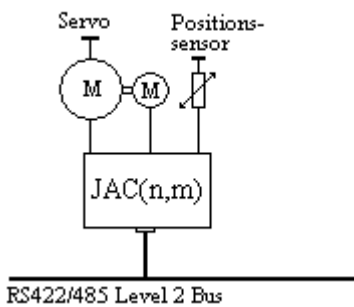


Berechnung von Trajektorien,
Steuerung der JACs,
lokale Sensordatenverarbeitung

LCP Third-Level-Bus

Abschnitt 3.3.10 beschreibt das JAC-Subsystem, von denen jeweils 3 als Slaves am Third-Level-Bus eines LCP arbeiten.

3.3.10 JAC(n,m) (Joint and Axis Controller)

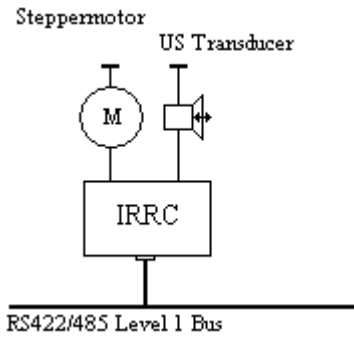


Steuerung der Servos,
Abfrage der Positionsgeber und Tachogeneratoren

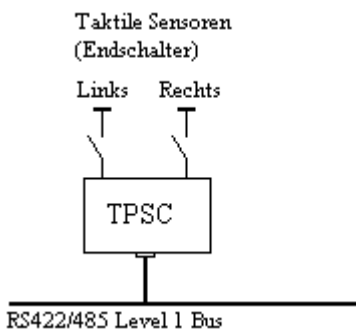
SPU Second-Level-Bus

Die SPU ist das zweite Subsystem, das einerseits als Slave am First-Level-Bus angeschlossen ist und andererseits als Master für einen untergeordneten Second-Level-Bus arbeitet. Dieses Subnetz ist für die Sensordatenverarbeitung und Sensorfusion innerhalb des Gesamtsystems zuständig.

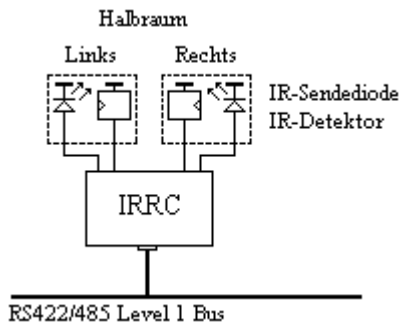
3.3.11 USRC (Ultrasonic Range Controller)



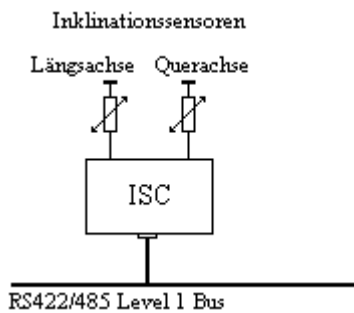
3.3.12 TPSC (Tactile Proximity Sensor Controller)



3.3.13 IRRC (Infrared Range Controller)

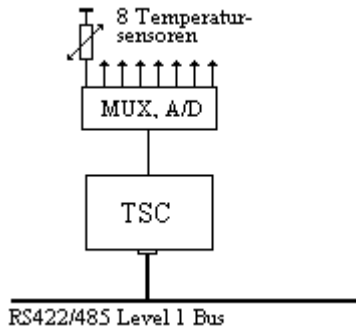


3.3.14 ISC (Inclination Sensor Controller)

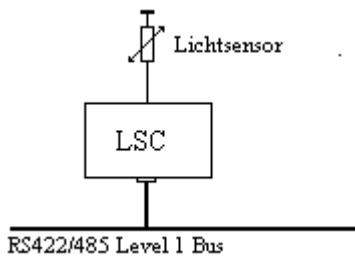


Der ISC mißt laufend die Neigung der Längsachse und der Querachse des Roboterkörpers gegen den Boden. Das heißt, bei jedem Meßzyklus fallen zwei Meßwerte im Bereich $-60^\circ..+60^\circ$ an.

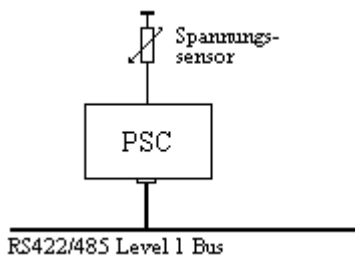
3.3.15 TSC (Temperature Sensor Controller)



3.3.16 LSC (Light Sensor Controller)



3.3.17 PSC (Proprioceptive Sensor Controller)



4 Benutzerschnittstellen

Da ein mobiler Roboter üblicherweise autonom agieren soll, also meistens fern von jedem menschlichen Operator, ist es nicht sinnvoll, ein aufwendiges grafisches Userinterface zu implementieren. Trotzdem muß er, sofern er innerhalb kooperativer Prozesse mit Menschen in Kontakt kommt, mit diesen in deren vertrauten Kommunikationsformen interagieren können. Das heißt, er muß, zumindest ansatzweise, sprechen und ebenso Sprache verstehen können. Da diese beiden Forderungen Gegenstand intensiver Forschung sind, ist es evident, daß sie nicht in befriedigendem Umfang in das Betriebssystem von ARACHNAE II integriert werden können. Deshalb wird, um wenigstens eine rudimentäre Interaktionsbasis zu gewährleisten, ein relativ simples Userinterface implementiert.

Dieses Interface besteht in der hier beschriebenen Version von ARACHNAE II aus drei voneinander unabhängigen Baugruppen:

- Dem User Interface Controller und Remote Control Processor UIC/RCP für die textuelle Ausgabe von Fehlermeldungen auf ein LCD, Statusanzeigen mit LEDs und Tasten zur manuellen Eingabe von Kommandos.
- Speech Synthesizer Controller SSC zur Ausgabe gesprochener Texte über einen Lautsprecher.
- Audio Processing Unit und Sound Direction Finder APU/SDF zur Auswertung akustischer Ereignisse wie Klatschen, Sprache und Detektion der Richtung von Schallereignissen.

4.1 User Interface Controller und Remote Control Processor UIC/RCP

Dieses besteht im wesentlichen aus einem zweizeiligen LCD und LEDs zur textuellen und optischen Anzeige von detaillierten Fehler-, Alarm- und Statusmeldungen, einigen Tasten zur manuellen Eingabe von Kommandos und schließlich einem Piezosummer für akustische Warnungen.

Der RCP-Teil dieser Subprozessoreinheit ist im weitesten Sinne auch als Benutzerschnittstelle zu verstehen, da mit ihm der Roboter über ein serielles Terminal oder ein anderes serielles VT100 kompatibles Endgerät mit einfachen Befehlen (F=Forward, B=Backward, H=Halt, usw) gesteuert werden kann.

Ein detaillierte Beschreibung der einzelnen Funktionen findet sich in Abschnitt 3.3.7.

4.2 Speech Synthesizer Controller SSC

Der SSC gibt einen empfangenen Textstring als gesprochene Sprache über einen Lautsprecher aus und realisiert damit wenigstens einen Teil der erforderlichen Basisfunktionalität für natürliche Mensch-Maschine-Kommunikation.

Die Funktionalität des SSC ist in Abschnitt 3.3.5 im Detail beschrieben.

4.3 Audio Processing Unit und Sound Direction Finder APU/SDF

Die Funktionalität dieser Baugruppe ist in der aktuellen Version von ARACHNAE II auf Detektion einfacher Schallereignisse wie Klatschen oder Pfeifen und zusätzliche Erkennung der Richtung, aus der diese Ereignisse ankommen, beschränkt. Der Grund ist der immense Aufwand, der für die

Erkennung und Interpretation gesprochener Sprache erforderlich ist und den Rahmen des Praktikums bei weitem sprengen würde.

Die einzelnen Funktionen sind im Abschnitt 3.3.6 beschrieben.

5 Nichtfunktionale Anforderungen

5.1 Qualitätsanforderungen

Die zentralen Qualitätsaspekte bei ARACHNAE II sind

- ◆ Wie gut findet sich der Roboter in einer unbekanntem Umgebung zurecht ?
- ◆ Wie sicher funktioniert die Kollisionsvermeidung und Objekterkennung ?
- ◆ Wie gut ist das System bezüglich Wartbarkeit und Erweiterbarkeit ?
- ◆ Wie stabil ist das System hinsichtlich des mechanischen Aufbaus ?

5.1.1 Wie gut findet sich der Roboter in einer unbekanntem Umgebung zurecht ?

Über die zeitlichen Restriktionen hinaus, die in Punkt 5.2 beschrieben sind, ist ein wesentliches Gütekriterium für einen autonomen, mobilen Roboter jenes, das beschreibt, wie er sich in einer ihm zunächst unbekanntem Umwelt zurechtfindet und agiert.

Eines der Designziele von ARACHNAE II ist es, die Navigationsalgorithmen so leistungsfähig zu implementieren, daß

5.1.2 Wie sicher funktioniert die Kollisionsvermeidung und Objekterkennung ?

Damit ARACHNAE II Aufgaben in einer realen Umwelt durchführen kann, müssen Kollisionen mit stationären und, mit gewissen Einschränkungen, mobilen Hindernissen zuverlässig erkannt beziehungsweise vermieden werden. Stationäre Objekte werden gleichzeitig von taktilen Sensoren (Endschalter) und von Ultraschall- und Infrarotsensoren erkannt. Mobile Objekte, wie zum Beispiel Menschen oder andere Roboter im Arbeitsbereich der Laufmaschine, werden ebenfalls von diesen Sensorsystemen erfaßt, sofern ihre Eigengeschwindigkeit nicht die Verarbeitungsleistung und damit die Reaktionszeit des Systems überfordert.

5.1.3 Wie gut ist das System bezüglich Wartbarkeit und Erweiterbarkeit ?

Da das System im geplanten Ausbau lediglich eine Plattform für weitere Entwicklungen und Experimente darstellt, ist es von nahezu existentieller Bedeutung, zukünftige Erweiterungen bezüglich neuer Funktionalitäten und Leistungsmerkmale bereits im jetzigen Design genügend zu berücksichtigen. Das gilt sowohl für den mechanischen Aufbau des Roboters selbst, als auch für die Steuerungshardware, die Sensorik und im besonderen für die Software.

Beim mechanischen Design ist ein modulartiger Aufbau anzustreben, der es erlaubt, einzelne Teile der Konstruktion gegen neue auszutauschen, ohne das Gesamtkonzept zu beeinträchtigen oder verwerfen zu müssen (zum Beispiel: Eine neue Bein konstruktion muß sich ohne massive Um- oder Ausbauten in den Korpus integrieren lassen. Der Idealfall wäre: Zwei Schrauben lösen, das alte Bein ausbauen und das neue Bein mit den beiden Schrauben an den gleichen Platz befestigen).

Hinsichtlich des hardwaremäßigen Aufbaus der Steuerungselektronik und der Sensorik gelten prinzipiell die gleichen Forderungen wie für die Mechanik. Eine neue Funktionalität oder ein neues Leistungsmerkmal benötigt lediglich das Anstecken eines neuen Prozessors an den Bus beziehungsweise den Austausch oder die Neuprogrammierung eines bereits bestehenden, ohne die jeweils anderen in irgendeiner Weise zu beeinträchtigen.

Die Software ist so modular aufgebaut, dass Funktionalitäten auf einer höheren Systemebene immer auf bereits bestehende, niederwertigere Funktionen zugreifen (zum Beispiel werden Ergebnisse der Navigation und Pfadplanung nicht direkt vom zuständigen Subsystem in Motorsteuerbefehl umgesetzt, sondern an den dafür zuständigen Prozess beziehungsweise Subsystem weitergeleitet).

5.1.4 Wie stabil ist das System hinsichtlich des mechanischen Aufbaus ?

Wie bereits mehrfach erwähnt, muß gerade eine mobile Laufmaschine ein möglichst geringes Eigengewicht aufweisen, da ein Roboter mit Beinen im Gegensatz zu radgetriebenen Maschinen ein besonders empfindliches kinematisches System darstellt. Deshalb kommen als Materialien nur extrem leichte und trotzdem hochfeste Leichtmetalle und Kunststoffe in Frage. Gleichzeitig muß aber eine hinreichende mechanische Stabilität gewährleistet sein, damit bei eventuellen Kollisionen oder bei einer Exploration im rauhen und ebenen Terrain der Roboter nicht irreparabel beschädigt oder sogar nur funktionsunfähig wird.

5.2 Leistungsanforderungen

Um ein annähernd 'natürliches' Verhalten des Roboters in seiner Umwelt zu ermöglichen, müssen bestimmte zeitliche Rahmenbedingungen erfüllt werden. So darf eine Kollisionserkennung nicht so lange dauern, daß bei Ende der Verarbeitung der Roboter bereits beschädigt ist, oder eine Reaktion auf einen Lichtreiz so lange, daß das relevante Ereignis, auf das reagiert werden sollte, ebenfalls bereits obsolet ist. Aufgrund mangelnder Erfahrung lassen sich derzeit keine exakten Zeitangaben treffen, doch läßt sich sagen, daß die Reaktion auf visuelle oder taktile Reize nicht länger als etwa 1 Sekunde dauern darf.

Navigation und Pfadplanung werden aufgrund der verwendeten, relativ einfachen Prozessoren und wegen der inhärenten Komplexität des Problems durchaus 10 bis 30 Sekunden benötigen (je nach geforderter Genauigkeit und Qualität der Ergebnisse).

Diese zeitlichen Anforderungen spiegeln sich auch in der Struktur des Systems. Das heißt, Funktionen, die eine sehr schnelle Reaktion auf ein Ereignis erfordern (Kollision, etc), sind eher auf niedrigen Systemebenen anzutreffen, wo sie direkt auf Aktuatoren und Sensoren wirken können. Mit anderen Worten, der eher unkritische Prozeß der Navigation und Pfadplanung kann beziehungsweise muß durch den höherpriorären Prozeß der Kollisionserkennung unterbrechbar sein.

Die Genauigkeit des Weitbereichscanners (Ultraschallsensor) ist eher unkritisch, da er ausschließlich für die Abtastung der Umwelt im Entfernungsbereich von 30 cm bis zu zirka 10 m eingesetzt wird.

Der für den Nahbereich verwendete IR-Sensor benötigt dagegen eine Auflösung mindestens im cm-Bereich, um aussagekräftige und verwendbare Meßergebnisse für die körpernahe Objekt- und Hinderniserkennung zu generieren.

6 Fehlerverhalten

6.1 Mechanische Probleme und Fehler

An mechanischen Problemen, die während des laufenden Betriebs erkennbar beziehungsweise meßbar sind, kommen mit vertretbarem Aufwand derzeit nur das Blockieren der Beine und deren Antriebe in Frage. Stößt ein Bein während seiner Schwungphase an ein Hindernis, steigt die Stromaufnahme des Motors unverhältnismäßig stark an. Dieser Stromfluß wird gemessen und in eine geeignete Reaktion umgesetzt (Zurückziehen des Beins und Berechnung einer neuen Trajektorie).

6.2 Abweichungen und Fehler in den Sensoren

Die bei ARACHNAE II verwendeten Sensoren sind aus Kostengründen relativ billige oder selbstgebaute Ausführungen und deshalb mit einer hohen Ungenauigkeit behaftet. Besonders bei den entfernungsmessenden Sensoren (Ultraschall und Infrarot) werden daher zur Ermittlung eines Meßergebnisses mehrere Messungen durchgeführt und gemittelt. Die nachfolgende Sensordatenverarbeitung korrigiert die Ergebnisse anhand der durch einen einmaligen Kalibrationsvorgang ermittelten systematischen Abweichungen. Treten unverhältnismäßig viele 'Ausreißer' bei den Meßergebnissen auf, wird das Sensorsystem als nicht mehr 'vertrauenswürdig' eingestuft. Je nach Sensortyp wird darauf verschieden reagiert.

IR-Sensor: Dieser Sensor detektiert Objekte und Hindernisse im Nahbereich des Roboters. Fällt er aus irgendeinem Grund aus oder ist er nicht mehr 'vertrauenswürdig', kann die Laufmaschine durch mechanische Einwirkungen ebendieser 'übersehenen' Objekte schwer beschädigt oder gar zerstört werden. Deshalb erscheint es sinnvoll, den Roboter in so einem Fall anzuhalten und gegebenenfalls eine entsprechende Message an einen Operator zu senden.

US-Sensor: Falls der Weitbereichsscanner gestört ist oder ausfällt, ist die Navigations- und Pfadplanung beeinträchtigt beziehungsweise im Extremfall nicht mehr funktionsfähig. In beiden Fällen wird das System abgeschaltet, das heißt, es wird innerhalb des Rechnernetzes als nicht mehr 'vertrauenswürdig' deklariert und seine gesendeten Meßdaten werden ignoriert. Die weitere Bewegung in der Umwelt wird ab dann nur mehr vom IR-Sensor und den taktilen Sensoren gesteuert. Gegebenenfalls wird eine entsprechende Message an einen Supervisor gesendet.

Die bei ARACHNAE II verwendeten, einfachen taktilen Sensoren mit ihrem digitalen Charakter der Meßergebnisse (Kontakt vs. Kein Kontakt) gestatten keine Überprüfung und werden deshalb auch nicht überwacht.

6.3 Selbstdiagnose

Das System gestattet eine *POST* Prozedur (Power On Self Test). Wird der Roboter eingeschaltet, führt er eine relativ simple Selbstdiagnose durch. Diese besteht im wesentlichen aus der Abfrage aller im Netz befindlichen, aktiven Knoten und Sensorsystemen und deren momentanen Stati. Jedes dieser Subsysteme führt nun seinerseits einen Selbsttest durch und sendet das oder die Ergebnisse an die jeweils übergeordnete Instanz. Einige dieser Subsysteme sind zum rudimentären Betrieb des Roboters unbedingt nötig. Dazu gehören im Einzelnen der Master

Control Processor MCP als Koordinationsprozessor, der Master Motion Controller MMC für die Beisteuerung, wenigstens fünf Leg Control Processors LCP(n) und deren lokale Subsysteme JAC(n,m) (Joint and Axis Controller) sowie der Ultraschallsensor USRC (Ultrasonic Controller) und die beiden taktilen Sensoren TPSC (Tactile Proximity Sensor Controller). Fehlt einer dieser Prozessoren oder deklariert sich als nicht aktiv, ist der Roboter nicht betriebsbereit und gibt eine entsprechende Message an den Supervisor oder Operator aus.

Die Selbstdiagnose kann auch jederzeit über den optionalen User Interface Controller/Remote Control Processor UIC/RCP initiiert werden.

7 Dokumentationsanforderungen

7.1 Modulebene

Im Sourcecode sind bei jeder Procedure und bei jedem Modul Kommentare so einzufügen, sodaß jederzeit nachvollziehbar ist, welche Anforderungen beziehungsweise Leistungen dieser Code erbringt und ob damit letztlich alle Anforderungen der Spezifikation in die Implementierung eingeflossen sind.

7.2 Sourcecode

Die Sources zu den einzelnen Knoten sind sowohl auf einem Datenträger (Diskette) und als Printout aufzubewahren.

7.3 Endbericht

Als abschließendes Dokument wird ein Projektendbericht als HTML-File erstellt.

Anhang A Schaltbilder der Knotenrechner

A.1 MCP (Master Control Processor)

A.2 NP3 (Navigation and Path Planning Processor)

A.3 MMC (Master Motion Controller)

A.4 SPU (Sensor Processing Unit)

A.5 SSC (Speech Synthesizer Controller)

A.6 APU/SDF (Audio Processing Unit / Sound Direction Finder)

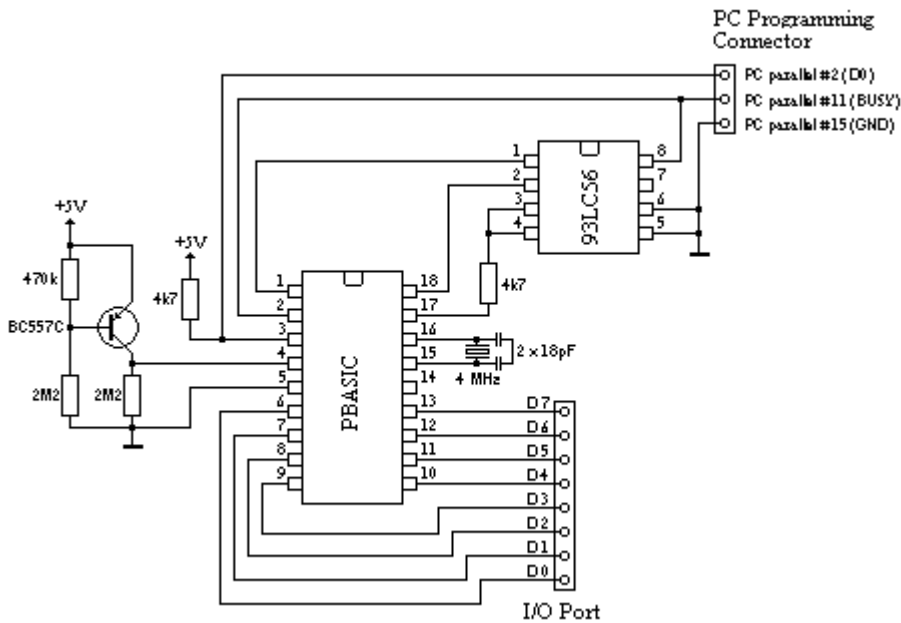
A.7 UIC/RCP (User Interface Controller / Remote Control Processor)

A.8 RLC (Radio Link Controller)

A.9 LCP(n) (Leg Control Processor)

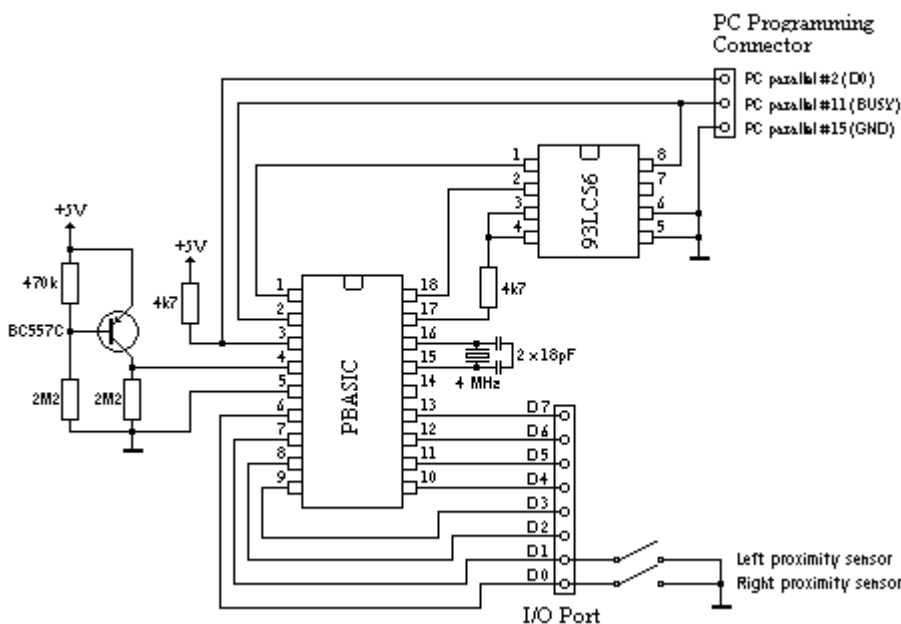
A.10 USRC (Ultrasonic Range Finder)

Der Microcontroller Basic STAMP I ist hier in der von Parallax Inc. beschriebenen Standard-schaltung wiedergegeben.



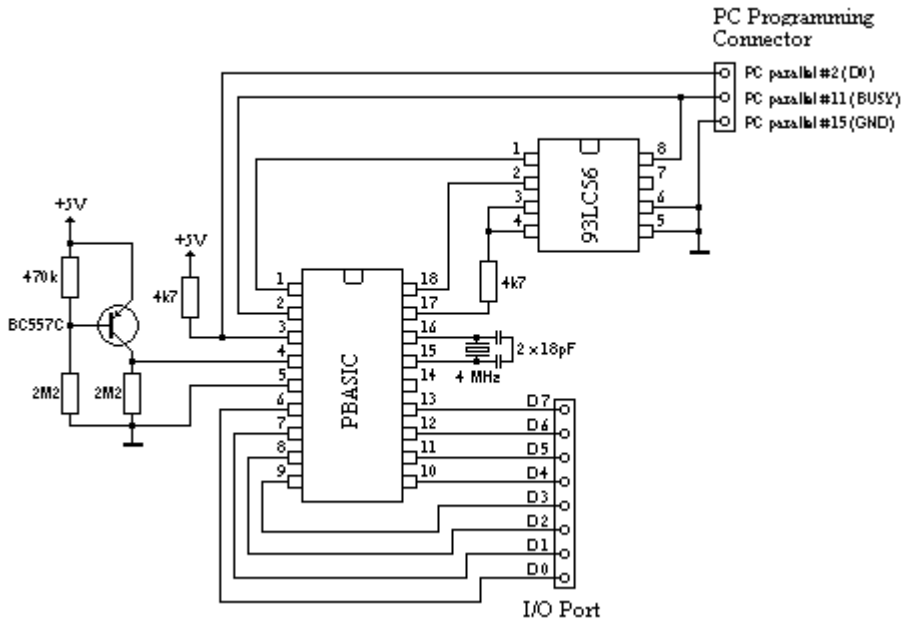
A.11 TPSC (Tactile Proximity Sensor Controller)

Der Microcontroller Basic STAMP I ist hier in der von Parallax Inc. beschriebenen Standard-schaltung wiedergegeben.



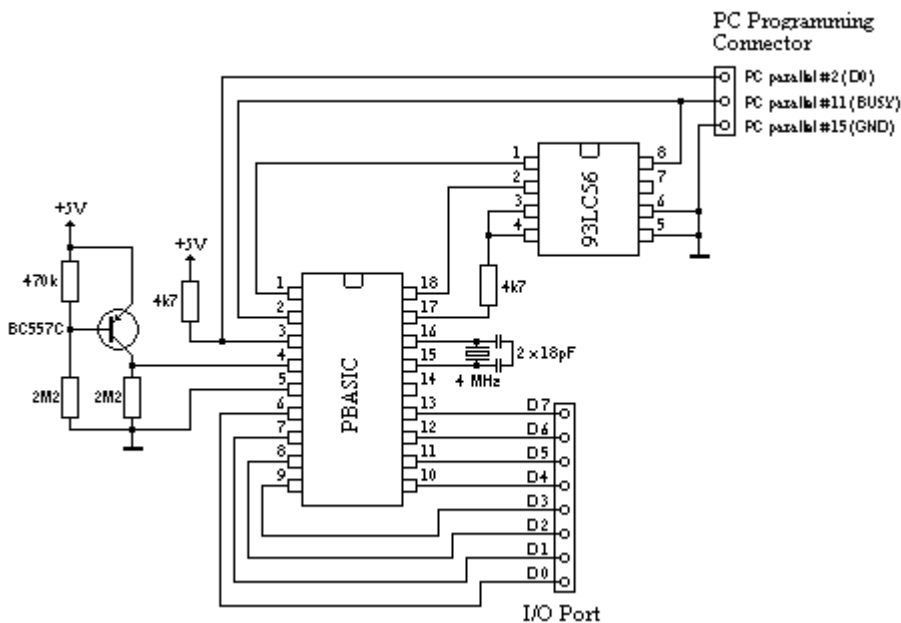
A.12 IRRC (Infrared Range Controller)

Der Microcontroller Basic STAMP I ist hier in der von Parallax Inc. beschriebenen Standard-schaltung wiedergegeben.



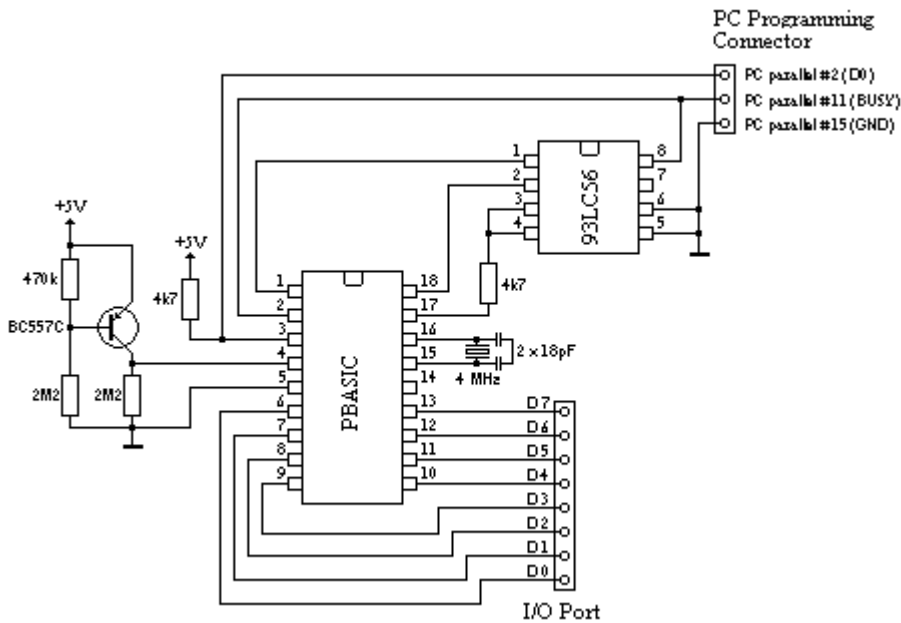
A.13 ISC (Inclination Sensor Controller)

Der Microcontroller Basic STAMP I ist hier in der von Parallax Inc. beschriebenen Standard-schaltung wiedergegeben.



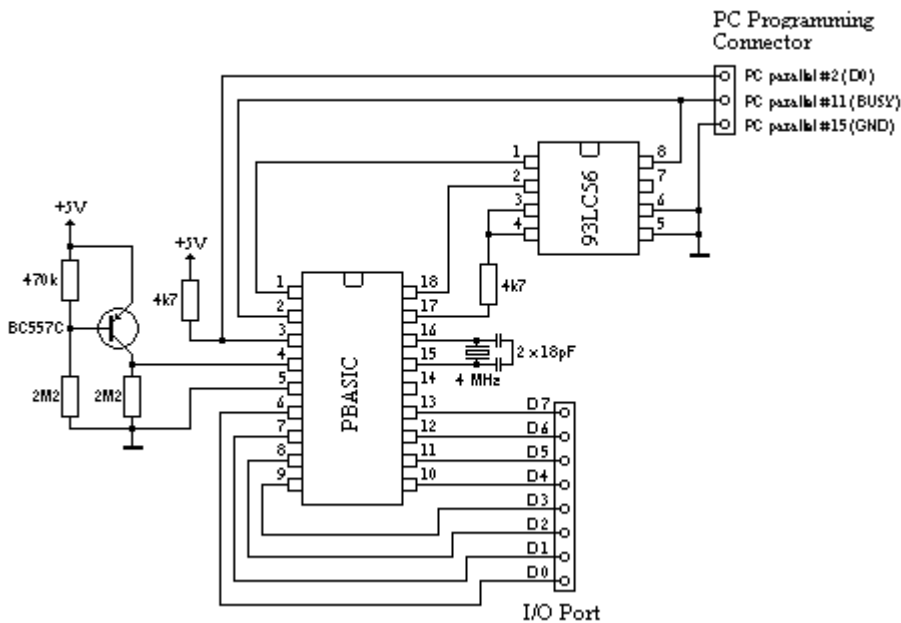
A.14 TSC (Temperature Sensor Controller)

Der Microcontroller Basic STAMP I ist hier in der von Parallax Inc. beschriebenen Standard-schaltung wiedergegeben.



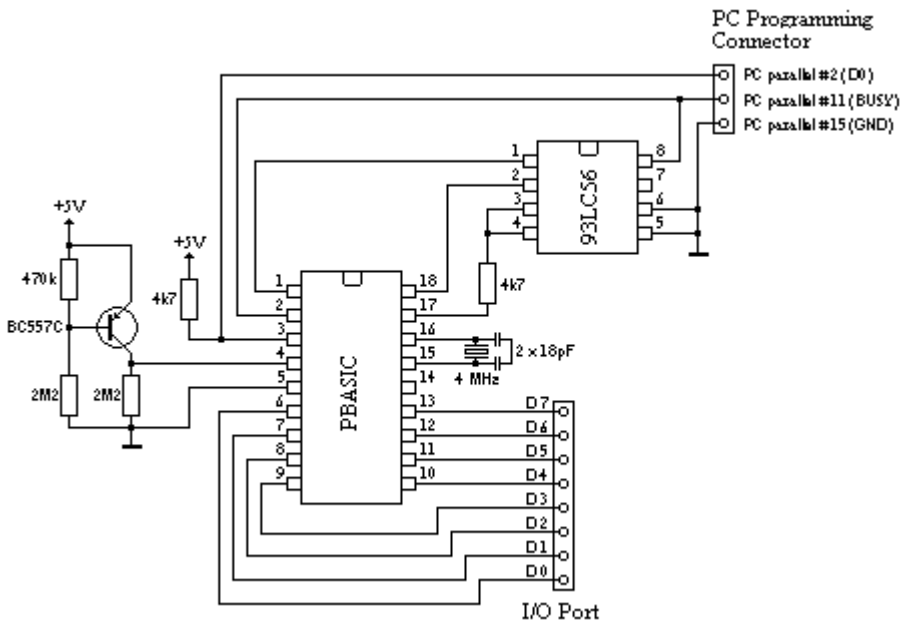
A.15 LSC (Light Sensor Controller)

Der Microcontroller Basic STAMP I ist hier in der von Parallax Inc. beschriebenen Standard-schaltung wiedergegeben.



A.16 PSC (Proprioceptive Sensor Controller)

Der Microcontroller Basic STAMP I ist hier in der von Parallax Inc. beschriebenen Standard-schaltung wiedergegeben.



A.17 JAC(n,m) (Joint and Axis Controller)

Anhang B Verwendete Abkürzungen und Glossar

- APU/SDF: Audio Processing Unit / Sound Direction Finder. Die APU ist ein Knotenrechner, der akustische Reize (Klatschen, Pfeifen, etc.) verarbeitet. Der SDF detektiert die Richtung von Schallereignissen.
- DOF: Degree of Freedom. Die Anzahl der rotatorischen und/oder translatorischen Freiheitsgrade eines kinematischen Systems. Bei den Beinen eines Roboters sind das im allgemeinen die Anzahl der Achsen beziehungsweise Gelenke.
- IRRC: Infrared Range Controller. Ein Knotenrechner, der eine Infrarot Sende/Empfängereinheit zur Abtastung des körpernahen Bereichs 10..20cm vor dem Roboter steuert.
- ISC: Inclination Sensor Controller. Dieser Subprozessor verarbeitet Daten von zwei Sensoren, die die Neigung des Roboters in der Ebene messen.
- JAC(n,m): Joint and Axis Controller. Diese Einheit ist ein lokaler Subprozessor zu den LCP's und ist jeweils für die Steuerung eines Gelenkantriebs zuständig.
- LCP(n): Leg Control Processor. Dieser Knotenrechner ist für die Steuerung jeweils eines Beines zuständig. Er berechnet Trajektorien für kollisionsfreie Beinbewegungen und führt eine lokale Sensordatenverarbeitung für Gelenkpositionen und Geschwindigkeiten sowie Temperatur des tragenden Servos (α -Servo) durch.
- LSC: Light Sensor Controller. Ein Subprozessor, der programmgesteuert auf Helligkeitsunterschiede beziehungsweise Lichtreize reagiert.
- MCP: Master Control Processor. Ein Knotenrechner im verteilten Rechnernetz von ARACHNAE II, zuständig für Initialisierung und Administration des Netzes sowie Message Scheduling.
- MMC: Master Motion Controller. Ein Knotenrechner, der Vorgaben hinsichtlich Positionierung des Roboters in dazu erforderliche Schrittfolgen umsetzt.
- POST: Power On Self Test. Bei ARACHNAE II eine Prozedur, die nach Anlegen der Versorgungsspannung beziehungsweise nach einem systemweiten Reset durchlaufen wird. Zur Detektion aktiver/inaktiver Subsysteme und Erfassung von Kalibrationsdaten für Sensoren und Aktuatoren.
- PSC: Proprioceptive Sensor Controller. Ein Subprozessor, der den inneren Zustand des Roboters (Batteriespannung, Motortemperaturen, usw.) überwacht.
- Ripple Gait:
- RLC: Radio Link Controller. Dieser Subprozessor bedient ein einfaches Funkmodem zur drahtlosen Datenübertragung von/zu einem Hostrechner beziehungsweise zur Kommunikation zwischen mehreren Robotern.
- SPU: Sensor Processing Unit. Der Knotenrechner im ARACHNAE II - Netz, der die Subprozessoren für die einzelnen Sensorsysteme administriert.
- SSC: Speech Synthesizer Controller. Ein Prozessorknoten, der einen einfachen Speechsynthesizer zur Ausgabe von Texten steuert.
- TPSC: Tactile Proximity Sensor Controller. Ein Knotenrechner, der zwei taktile Berührungssensoren zur Kollisions- und Objekterkennung steuert.
- Tripod Gait: Eine sechsbeinige Gangart, bei der sich immer jeweils drei Beine am Boden befinden, während sich die drei anderen an die nächste Stützposition bewegen.
- TSC: Temperature Sensor Controller. Ein Knotenrechner, der Daten von 8 Temperatursensoren beziehungsweise Meßstellen verarbeitet.

-
- UIC/RCP:** User Interface Controller / Remote Control Processor. Diese Subprozessoreinheit stellt ein einfaches User Interface aus einigen Schaltern und einem LCD zur Verfügung. Mit dem RCP kann der Roboter über einen RS232C-Downlink von einer dedizierten Konsole oder einem PC manuell gesteuert werden.
- USRC:** Ultrasonic Range Controller. Ein Subprozessor, der eine Ultraschall Transducer-Einheit zur Entfernungsmessung steuert.
- Wave Gait:** Jeweils ein Bein bewegt sich auf seinen neuen Aufsetzpunkt. Die Beinbewegungen laufen von hinten nach vorne an einer Körperseite entlang und setzen sich an der anderen fort.

Anhang C Literatur

- [1] Colin M.Angle, Rodney A.Brooks, "*Small Planetary Rovers*", MIT Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139
- [2] John S.Bay, "*Design of the "Army-Ant" Cooperative Lifting Robot*", Bradley Department of Electrical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061-0111
- [3] Michael B.Binnard, "*Design of a Small Pneumatic Walking Robot*", Department of Mechanical Engineering, Massachusetts Institute of Technology
- [4] Michael B.Binnard, "*Leg Design for a Small Walking Robot*", Department of MEchanical Engineering, Massachusetts Institute of Technology
- [5] Johann Borenstein, Yoram Koren, "*Real-Time Obstacle Avoidance for Mobile Robots*", Advanced Technologies Lab, University of Michigan, 1101 Beal Avenue, Ann Arbor, MI 48109-2110
- [6] Johann Borenstein, Yoram Koren, "*Error Eliminating Rapid Ultrasonic Firing for Mobile Robot Obstacle Avoidance*", Advanced Technologies Lab, University of Michigan, 1101 Beal Avenue, Ann Arbor, MI 48109-2110
- [7] Johann Borenstein, "*Control and Kinematic Design for Multi-Degree-of-Freedom Mobile Robots with Compliant Linkage*", Advanced Technologies Lab, University of Michigan, 1101 Beal Avenue, Ann Arbor, MI 48109-2110
- [8] J.Borenstein, H.R.Everett, L.Feng, "*Where Am I? Sensors and Methods for Mobile Robot Positioning.*", University of Michigan, Department of Mechanical Engineering and Applied Mechanics, Mobile Robotics Laboratory, 1101 Beal Avenue, Ann Arbor, MI 48109
- [9] Johann Borenstein, "*Compliant-linkage Kinematic Design for Multi-Degree-of-Freedom Mobile Robots*", Advanced Technologies Lab, University of Michigan, 1101 Beal Avenue, Ann Arbor, MI 48109-2110
- [10] Johann Borenstein, "*Multi-layered Control of a Four-Degree-of-Freedom Mobile Robot with Compliant Linkage*", Advanced Technologies Lab, University of Michigan, 1101 Beal Avenue, Ann Arbor, MI 48109-2110
- [11] J.Borenstein, H.R.Everett, L.Feng, D.Wehe, "*Mobile Robot Positioning - Sensors and Techniques*", Advanced Technologies Lab, University of Michigan, 1101 Beal Avenue, Ann Arbor, MI 48109-2110
- [12] Rodney A.Brooks, "*A Robust Layered Control System for a Mobile Robot*", AI-Memo 864, Massachusetts Institute of Technology, 545 Technology Square, Room 741, Cambridge, MA 02139
- [13] Thomas Burg, "A Six-legged Walking Robot", <http://cccsrv.trevano.ch/~tburg/Hexapod.html>
- [14] K.S.Chong, L.Kleeman, "*Sonar Feature Map Building for a Mobile Robot*", Intelligent Robotics Research Centre, Department of Electrical and Computer Systems Engineering, Monash University, Australia
- [15] Howie Choset, Ilhan Konukseven, Joel Burdick, "*Mobile Robot Navigation: Issues in Implementing the Generalized Voronoi Graph in the Plane*", Department of Mechanical Engineering, California Institute of Technology, Pasadena, CA 91125
- [16] Andrew Dean Christian, "*Design and Implementation of a Flexible Robot*", AI Technical Report 1153, Massachusetts Institute of Technology, 545 Technology Square, Room 741, Cambridge, MA 02139
- [17] Kynan Eng, Alec Robertson, "*Robbie the Running Robot*", Monash University, <http://netspace.net.au/~kynan/robot/robot.html>

-
- [18] Cynthia Ferrell, "*Failure Recognition and Fault Tolerance of an Autonomous Robot*", Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 545 Technology Square, Room 741, Cambridge, MA 02139
 - [19] Cynthia Ferrell, "*A Comparison of Three Insect Inspired Locomotion Controllers*", Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 545 Technology Square, Room 741, Cambridge, MA 02139
 - [20] Cynthia Ferrell, "*Robust Agent Control of an Autonomous Robot with many Sensors and Actuators*", AI Technical Report 1443, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 545 Technology Square, Room 741, Cambridge, MA 02139
 - [21] Cynthia Ferrell, "*Orientation Behaviour Using Registered Topographic Maps*", Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 545 Technology Square, Room 741, Cambridge, MA 02139
 - [22] John Goldie, "*Summary of Well Known Interface Standards*", National Semiconductor, Application Note 216, January 1996
 - [23] John Goldie, Greg Krikorian, "*Increasing System ESD Tolerance for Line Drivers and Receivers Used in RS-232 Interfaces*", National Semiconductor, Application Note 878, January 1993
 - [24] John Goldie, "*Inter-Operation of Interface Standards*", National Semiconductor, Application Note 972, November 1994
 - [25] John Goldie, "*Ten Ways to Bullet-Proof RS-485 Interfaces*", National Semiconductor, Application Note 1057, October 1996
 - [26] B.Holt, J.Borenstein, Y.Koren, D.Wehe, "*OmniNav: Obstacle Avoidance for Large, Non-Circular, Omnidirectional Robots*", Advanced Technologies Lab, University of Michigan, 1101 Beal Avenue, Ann Arbor, MI 48109-2110
 - [27] Dylan Horvath, Jeff Lee, Stefan Williams, "*Hexotica - The Design and Implementation of a Small Walking Robot*", <http://real.uwaterloo.ca/~robot/>
 - [28] Manfred Huber, Willard S.MacDonald, Roderic A.Grupen, "*A Control Basis for Multilegged Walking*", Laboratory for Perceptual Robotics, Department of Computer Science, University of Massachusetts, Amhurst, MA 01003
 - [29] Manfred Huber, Roderic A.Grupen, "*A Control Structure for Learning Locomotion Gaits*", Laboratory for Perceptual Robotics, Department of Computer Science, University of Massachusetts, Amhurst, MA 01003
 - [30] Joseph L.Jones, Anita M.Flynn, "*Mobile Roboter - Von der Idee zur Implementierung*", Addison Wesley Publishing Company, 1996
 - [31] Avinash C.Kak, Akio Kosaka, "*Multisensor Fusion for Sensory Intelligence in Robotics*", School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 17907-1285
 - [32] L.Kleeman, R.Kuc, "*Mobile Robot Sonar for Target Localization and Classification*", Intelligent Robotics Research Centre, Department of Electrical and Computer Systems Engineering, Monash University, Australia
 - [33] Lindsay Kleeman, "*A Three Dimensional Localiser for Autonomous Robot Vehicles*", Intelligent Robotics Research Centre, Department of Electrical and Computer Systems Engineering, Monash University, Australia
 - [34] Jon M.Kleinberg, "*The Localization Problem for Mobile Robots*", Laboratory for Computer Science, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139
 - [35] Willard S.MacDonald, "*Thing. A Four-Legged Walking Robot*", Laboratory for Perceptual Robotics, Department of Computer Science, University of Massachusetts, Amherst MA 01003, <http://piglet.cs.umass.edu:4321/thing/thing.html>

-
- [36] Willard S. MacDonald, "*Design and Implementation of a Multilegged Walking Robot*", Laboratory for Perceptual Robotics, Department of Computer Science, University of Massachusetts, Amhurst, MA 01003
 - [37] Fred G. Martin, Pankaj Oberoi, Randy Sargent, "*The 6.270 Robot Builder's Guide*", (1992), The Media Lab, Massachusetts Institute of Technology, 20 Ames Street, Room E15-309, Cambridge, MA 02139
 - [38] Fred G. Martin, "*The MIT Sensor Robot: User's Guide and Technical Reference*", (1991), The Media Lab, Massachusetts Institute of Technology, 20 Ames Street, Room E15-309, Cambridge, MA 02139
 - [39] Jonathan W. Mills, "*Stiquito: A Small, Simple, Inexpensive Hexapod Robot*", Computer Science Department, Indiana University, Bloomington, Indiana 47405
 - [40] Sundar Narasimhan, David M. Siegel, John M. Hollerbach, "*A Standard Architecture for Controlling Robots*", MIT Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139
 - [41] Ahmet ONAT, "The six-legged walker GOKIBURI", <http://turbine.kuee.kyoto-u.ac.jp/staff/onat.html>
 - [42] Gary B. Parker, David W. Braun, Ingo Cyliax, "*Evolving Hexapod Gait Generation Using a Cyclic Genetic Algorithm*", Department of Computer Science, Indiana University, Bloomington, IN 47405
 - [43] Frank Scott, "Rodney, A Six-Legged Walking Robot", <http://www.frasco.ac.uk/>
 - [44] D. Wehe, Johann Borenstein, L. Feng, Y. Koren, "*Mobile Robot Navigation in Narrow Aisles with Ultrasonic Sensors*", Advanced Technologies Lab, University of Michigan, 1101 Beal Avenue, Ann Arbor, MI 48109-2110
 - [45] Sivakumar Sivasothy, "*Transceivers and Repeaters Meeting the EIA RS-485 Interface Standard*", National Semiconductor, Application Note 409, July 1986
 - [46] David Wettergreen, Chuck Thorpe, "*Gait Generation for Legged Robots*", The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213-3891 USA