

Die neue Motorsteuerung

Einfache Ansteuerung / Top Leistung zum LowCost Preis



RN-MOTOR Version 1.0 + 1.2

Dieses Board gibt es in verschiedenen Varianten. Der Aufbau aller Varianten ist identisch, nur das Betriebssystem ist bei den Varianten unterschiedlich. Alle Varianten zeichnen sich durch ein hervorragendes Preis/Leistungsverhältnis und eine kinderleichte Ansteuerung aus. Durch einfache RS232-Befehle (z.B. Print Befehl) oder I2C-Befehle lassen sich die Motoren völlig unabhängig und exakt steuern. Eine eingebaute PWM-Stromregelung sorgt dafür das auch Schrittmotor kraftvoll arbeiten und zudem bipolare Motoren beliebiger Nennspannung 1 bis 22V angesteuert werden können. Der Strom wird einfach per Befehl dem Board mitgeteilt, alles andere macht das Board selbstständig.

Steuersoftware, Beispiele für PC oder RN-Control, RNBFA-Board , C-Control werden beim Bausatz mitgeliefert.



Ich denke die Leistungsdaten und Features können wieder überzeugen!

Datum der Doku: 06.07.05

Die verschiedenen Ausführungen des Boards:

RN-Motor ST RS232

Diese Variante steuert bis zu zwei Schrittmotoren völlig unabhängig per RS232 an. Es sind TTL-Anschlüsse sowie Standard RS232 Anschlüsse vorhanden. Ein Controller als auch ein PC kann somit direkt angeschlossen werden. Ein Port würde somit für die komplette Steuerung ausreichen (2 Ports sind ideal). Die Übertragung erfolgt mit 9600 Baud.

Die Schritte werden natürlich vom Board automatisch generiert, dennoch kann die Anzahl genau überwacht werden. Die Stromregelung erfolgt automatisch. Der erlaubte Strangstrom kann per Befehl bis zu 2 A eingestellt werden. Das reicht gewöhnlich selbst für sehr starke Schrittmotoren aus.

RN-Motor ST I2C

Wie zuvor, jedoch erfolgt hier die Ansteuerung über den beliebten I2C-Bus. Nahezu alle modernen Controller/Boards verfügen über einen I2C-Bus (z.B. RN-Control, RNBFR4 und viele mehr)

RN-Motor GT RS232

Diese Variante steuert bis zu vier Getriebemotoren. völlig unabhängig per RS232 an. Es sind TTL-Anschlüsse sowie Standard RS232 Anschlüsse vorhanden. Ein Controller als auch ein PC kann somit direkt angeschlossen werden. Ein Port würde somit für die komplette Steuerung ausreichen (2 Ports sind ideal). Die Übertragung erfolgt mit 9600 Baud. Zwei Motoren der vier werden per PWM in der Geschwindigkeit geregelt.

Alle 4 Motoren können bis zu 2A aufnehmen

RN-Motor GT I2C

Wie zuvor, jedoch erfolgt hier die Ansteuerung über den beliebten I2C-Bus. Nahezu alle modernen Controller/Boards verfügen über einen I2C-Bus (z.B. RN-Control, RNBFR4 und viele mehr)

Und das tollste:

Man kann das Board entweder gleich in der gewünschten Variante kaufen oder es später durch Wechsel der Firmware(Betriebssystem) von einer Variante in eine andere umbauen. Die Firmware wird für einen geringen Unkostenbetrag auch einzeln über robotikhardware.de angeboten.

Hier die Leistungsmerkmale der Schrittmotorversionen *RN-Motor ST RS232* und *RN-Motor ST I2C*:

- Deutlich höheres Drehmoment (Motorkraft) durch automatische PWM-Stromregelung
- Voll- und Halbschrittmodus (in I2C Version, in RS232 Version nur Vollschrittmodus)
- Maximal zulässiger Strombegrenzung (bei Schrittmotor Strangstrombegrenzung) kann durch einfachen Befehl von einem Controller oder PC zwischen ca. 100mA und max. 2A festgelegt werden. Kein rumschrauben mehr an Potis!
Auch wenn kurzzeitig höhere Belastungen möglich sind, wird eine maximale Dauerbelastung von ca. 2 – 3 A pro Schrittmotor (Stangstrom 1 bis 1,5 A) empfohlen
- Es können nahezu alle bipolaren Schrittmotoren beliebiger Nennspannung 1V bis zur Versorgungsspannung (7 bis 22V) angesteuert werden. Beispiel: Sie können auch einfach einen 1.9V Motor an ein Board mit 9V oder 12V anschließen, der Motor wird nicht überlastet da der max. Strom eingestellt wird. RN-Motor arbeitet etwas ähnlich wie die L297/L298 Schaltung, bietet jedoch mehr High Level-Funktionen
- Durch die Verwendung eines programmierten Controllers und mehrerer integrierter Schaltkreise benötigt das ganze Board nur wenig Bauteile. Dies reduziert Kosten und erleichtert den Aufbau.
- Ein mitgelieferter Steuerprogramm erlaubt die Ansteuerung der Schrittmotoren bequem per Mausklick. Durch die Möglichkeit unterschiedliche Ströme und Geschwindigkeiten einzustellen, lassen sich Schrittmotoren auch optimal austesten.
- Übermittelt wird nur der Befehl der beschreibt wieviel Schritte ein bestimmter Motor in welcher Geschwindigkeit und Drehrichtung erfolgen soll. Um die Generierung und das zählen der Schritte kümmert sich das Board völlig automatisch. Dadurch wird der steuernde Controller oder PC erheblich entlastet
- 255 exakte Geschwindigkeitstufen möglich
- Drehrichtung und max Strombedarf jederzeit änderbar
- Endlosfunktion, welche Schrittmotoren solange dreht bis ein Stopp Befehl kommt
- Schrittezähler, die ausgeführten Schritte der Motoren können jederzeit abgerufen werden, somit läßt sich exakt die Länge der gefahrenen Strecke berechnen
- Bei ausgeschalteten Motoren benötigt das Board weniger als 40mA, das ist deutlich weniger als bei üblichen L297/L298 Schaltungen.
- Board läßt sich auch per RS232 Befehl in Sleep-Mode versetzen um Stromaufnahme weiter zu verringern (auf ca. 20mA).
- LED signalisiert wenn die Betriebsspannung nicht ausreicht um den Schrittmotor-Nennstrom (also maximale mögliche Leistung) bei der gewählten Geschwindigkeit zu erreichen. Eine recht nützliche Analysefunktion.
- ISP-Programmierschluß –Atmel Programmierer können das Board mit einem eigenen Betriebssystem versehen. Dies ist aber nicht notwendig!
- Zur Steuerung des Board´s würde im Notfall ein RS232 TX PORT reichen. Bei der RN-Control könnte fast jeder Port dafür verwendet werden wenn in Basic programmiert wird. Ideal sind jedoch zwei Ports um auch die Rückmeldungen des Motorboards auswerten zu können.
- Ansteuerung über beliebige Controller z.B. RN-Control, RNBFA, C-Control oder PC

- Neue Firmware (Betriebssystem des Boards) kann einfach neu per PC eingespielt werden (dazu ist lediglich ein ISP-Programmierspater notwendig . z.B. im Shop bei robotikhardware.de)
- Erlaubte Versorgungsspannung 7V bis 22 V
- Sehr kompakt, nur halbes Europaformat nach Roboternetz-Norm (ca. 100x75mm). Dadurch mit anderen Boards nach Roboternetz-Standard huckepack verschraubbar.
- Deutsche Doku mit Programmbeispielen für RN-Control und C-Control und PC-Steuerprogramm

Die Übersicht über RS232-Schrittmotorbefehle:

RN-Motor arbeitet mit einer Baudrate von 9600 Baud. Die einzelnen Befehle können bei den meisten Programmiersprachen durch einfache Print-Anweisungen erfolgen. Wird in Bascom-Basic programmiert, kann z.B. fast jeder Port als RS232 benutzt werden.

Motorstrom einstellen				
Wird kein Motorstrom eingestellt, so ist dieser nach dem Einschalten/Reset auf 100mA festgelegt				
Kennung	Befehls-code	1 Parameter	2. Parameter	3. Parameter
#rm	i	0 oder 1 oder 2 0=linker Motor 1=rechter Motor 2=beide Motoren	0 –bis 200 Stromstärke in mA / 10 (Beispiel:für 1900 mA muss 190 angegeben werden.	
Beispiel: print "#rmi" & chr(0) & chr(200)				

Referenzspannung einstellen				
Gewöhnlich ist die Referenzspannung des Controllers mit 2 Widerständen fest auf 2.5V eingestellt, so das dieser Befehl nicht notwendig ist. Bei gewissen Widerstandstoleranzen kann die Referenzspannung am Pin 21 des Controller etwas abweichen. In einem solchen Fall kann man über diesen Befehl den gemessenen Wert einstellen. Wichtig: Der Wert darf nicht zu niedrig eingestellt werden, dies könnte den Controller beschädigen!				
Kennung	Befehls-code	1 Parameter	2. Parameter	3. Parameter
#rm	v	0 bis 255 Gemessene Referenzspannung in Volt * 100 (241 bedeutet 2,41V)		
Beispiel: print "#rmv" & chr(241)				

Motor einschalten				
Kennung	Befehls-code	1 Parameter	2. Parameter	3. Parameter
#rm	o	0 oder 1 oder 2 0=linker Motor 1=rechter Motor 2=beide Motoren		
Beispiel: print "#rmo" & chr(0)				

Motor Drehrichtung				
Kennung	Befehls-code	1 Parameter	2. Parameter	3. Parameter
#rm	d	0 oder 1 oder 2 0=linker Motor 1=rechter Motor 2=beide Motoren	0 oder 1 0= links rum drehen 1= rechts rum drehen.	
Beispiel: print "#rmd" & chr(0) & chr(1)				

Schrittgeschwindigkeit einstellen

Die Schrittgeschwindigkeit wird in Stufen von 0 bis 255 eingestellt. 0 ist dabei die schnellste und 255 die langsamste Schrittgeschwindigkeit. Die Geschwindigkeit ist von der Firmware (Betriebssystem von RN-Motor) abgängig. In Version 1.0 ist die Einteilung wie unten angegeben (bei Firmware Updates kann das geändert werden)

Wichtig: Anzumerken ist noch, das nicht alle Schrittmotoren die hohen Geschwindigkeitsstufen 0 bis 4 erlauben. Wenn sich der Motor bei diesen Stufen nicht dreht, so ist zu prüfen ob auch der zulässige Maximalstrom eingestellt ist. Ist dies der Fall und der Motor dreht immer noch nicht, dann ist die Geschwindigkeit zu hoch für den Motor.

Kennung	Befehls-code	1 Parameter	2. Parameter	3. Parameter
#rm	g	0 oder 1 oder 2 <i>0=linker Motor</i> <i>1=rechter Motor</i> <i>2=beide Motoren</i>	0 bis 255 <i>0= 1000 Schritte pro Sek.</i> <i>1= 500 Schritte pro Sek.</i> <i>2= 333 Schritte pro Sek.</i> <i>3= 250 Schritte pro Sek.</i> <i>4= 200 Schritte pro Sek.</i> <i>5= 167 Schritte pro Sek</i> <i>6= 142 Schritte pro Sek</i> <i>7= 125 Schritte pro Sek</i> <i>8= 111 Schritte pro Sek</i> <i>9= 100 Schritte pro Sek</i> ... <i>255= 4 Schritte pro Sekunde</i> <i>(geringe Abweichungen sind möglich)</i>	
Beispiel: print "#rmg" & chr(5)				

Motor endlos drehen bis Stopp- oder Ausschalt-Befehl kommt

Die ausgeführten Schritte werden dabei mitgezählt und können über den Befehl p abgerufen werden.

Kennung	Befehls-code	1 Parameter	2. Parameter	3. Parameter
#rm	e	0 oder 1 oder 2 <i>0=linker Motor</i> <i>1=rechter Motor</i> <i>2=beide Motoren</i>		
Beispiel: print "#rme" & chr(0)				

Motor stoppen aber Strom eingeschaltet lassen (Schrittposition wird gehalten)

Die bisher ausgeführten Schritte können auch nach Stop über den Befehl p abgerufen werden.

Kennung	Befehls-code	1 Parameter	2. Parameter	3. Parameter
#rm	s	0 oder 1 oder 2 <i>0=linker Motor</i> <i>1=rechter Motor</i> <i>2=beide Motoren</i>		
Beispiel: print "#rms" & chr(0)				

Motor eine bestimmte Anzahl von Schritten drehen

Kennung	Befehls-code	1 Parameter	2. Parameter	3. Parameter
#rm	z	0 oder 1 oder 2 <i>0=linker Motor</i> <i>1=rechter Motor</i> <i>2=beide Motoren</i>	Low Byte der Schrittzahl <i>also Schrittzahl mod 256</i>	High Byte der Schrittzahl <i>also Schrittzahl / 256</i>
Beispiel: print "#rmz" & chr(0) & chr(1) & chr(4) 'Motor links 300 Schritte machen				

Motor einen einzigen Schritt drehen

Kennung	Befehls-code	1 Parameter	2. Parameter	3. Parameter
#rm	x	0 oder 1 oder 2 <i>0=linker Motor</i> <i>1=rechter Motor</i> <i>2=beide Motoren</i>		
Beispiel: print "#rmx" & chr(0)				

Motor ganz ausschalten				
Kennung	Befehls-code	1 Parameter	2. Parameter	3. Parameter
#rm	a	0 oder 1 oder 2 <i>0=linker Motor</i> <i>1=rechter Motor</i> <i>2=beide Motoren</i>		
Beispiel: print "#rma" & chr(0)				

Gefahrene Schrittzahl abrufen				
Die Anzahl der ausgeführten Schritte nach einem neuem Motorstart oder Drehrichtungswechsel werden über RS232 angezeigt- Dies ist ein long Wert.				
Kennung	Befehls-code	1 Parameter	2. Parameter	3. Parameter
#rm	p	0 oder 1 <i>0=linker Motor</i> <i>1=rechter Motor</i>		
Beispiel: print "#rmp" & chr(0)				

Konfiguration anzeigen				
Einige Konfigurationswerte/Einstellungen werden über RS232 angezeigt				
Kennung	Befehls-code	1 Parameter	2. Parameter	3. Parameter
#rm	t			
Beispiel: print "#rmt				

Sleep Modus				
Das Board wird in den Ruhemodus versetzt. Motoren und Controller werden auf Stromsparmmodus gestellt. Das aufwecken ist nur über RESET möglich . Reset kann über einen Controller-Port erfolgen, wenn dieser mit der Reset-Leitung verbunden wird. Oder über Tastendruck bzw. Unterbrechnung,der Stromversorgung				
Kennung	Befehls-code	1 Parameter	2. Parameter	3. Parameter
#rm	l			
Beispiel: print "#rml				

Die Übersicht über I2C-Schrittmotorbefehle:

RN-Motor abreitet mit I2C-Befehlen die immer 5 Bytes übertragen. Einige Befehle benötigen weniger Parameter, in diesem Fall ist der Inhalt der nachfolgenden Bytes unwichtig. Wichtig ist nur das immer 5 Bytes übertragen werden, dies erlaubt dem Controller eine schnellere und einfachere Auswertung. **Die I2C-Slave-Adresse beträgt normalerweise Hex 56 zum senden von Befehlen und Hex 57 für den Abruf von Daten (Abruf der Schrittzahl).**

Achtung: Ab RN-MOTOR ST Betriebssystem Version 1.1 sind auch andere SlaveID's möglich. Im Zweifel kann sowohl die Betriebssystem Version und die aktuelle Slave ID über RS232 ausgegeben werden. Dazu einfach die RS232 Schnittstelle des Board's über ein geeignetes Kabel mit dem PC verbinden. Ein Terminalprogramm mit 9600 Baud Einstellung laden und RESET am Board klicken.

Motorstrom einstellen				
Wird kein Motorstrom eingestellt, so ist dieser nach dem Einschalten/Reset auf 100mA festgelegt				
Kennung 1 Byte	Befehls- code 2. Byte	3 Byte	4. Byte	5. Byte
10	1	0 oder 1 oder 2 <i>0=linker Motor 1=rechter Motor 2=beide Motoren</i>	0 –bis 200 <i>Stromstärke in mA / 10 (Beispiel:für 1900 mA muss 190 angegeben werden.</i>	
Bascom-Beispiel:				
<pre>Dim I2cdaten(6) As Byte 'Motorstrom auf 1900mA begrenzen I2cdaten(1) = 10 'Kennung muss bei RN-Motor immer 10 sein I2cdaten(2) = 1 'Befehlscode I2cdaten(3) = 0 '1 Parameter I2cdaten(4) = 1900 / 10 I2csend &H56, I2cdaten(1),5 'Befehl wird gesendet</pre>				

Referenzspannung einstellen				
Gewöhnlich ist die Referenzspannung des Controllers mit 2 Widerständen fest auf 2.5V eingestellt, so das dieser Befehl nicht notwendig ist. Bei gewissen Widerstandstoleranzen kann die Referenzspannung am Pin 21 des Controller etwas abweichen. In einem solchen Fall kann man über diesen Befehl den gemessenen Wert einstellen. Wichtig: Der Wert darf nicht zu niedrig eingestellt werden, dies könnte den Controller beschädigen!				
Kennung 1 Byte	Befehls- code 2. Byte	3 Byte	4. Byte	5. Byte
10	2	0 bis 255 <i>Gemessene Referenzspannung in Volt * 100 (241 bedeutet 2,41V)</i>	<i>In der Firmware 1.0 bis 1.5 sollte hier nochmals der gleiche Wert wie in Byte 3 eingetragen werden. In späteren Firmware- Revisionen ist dies nicht notwendig!</i>	
Bascom-Beispiel:				
<pre>Dim I2cdaten(6) As Byte I2cdaten(1) = 10 'Kennung muss bei RN-Motor immer 10 sein I2cdaten(2) = 2 'Befehlscode I2cdaten(3) = 241 '1 Parameter I2csend &H56, I2cdaten(1),5 'Befehl wird gesendet</pre>				

Voll- oder Halbschrittmodus				
Der Voll- und Halbschrittmodus ist immer nur generell für beide Motoren wählbar. Zwei unterschiedliche Modis gleichzeitig sind nicht möglich				

Kennung 1 Byte	Befehls- code 2. Byte	3 Byte	4. Byte	5. Byte
10	14	0 oder 1 <i>0=beide Motoren Vollschrittmodus 1= beide Motoren Halbschrittmodus</i>		

Bascom-Beispiel:

```
Dim I2cdaten(6) As Byte
I2cdaten(1) = 10 'Kennung muss bei RN-Motor immer 10 sein
I2cdaten(2) = 14 'Befehlscode
I2cdaten(3) = 1 '1 Parameter
I2csend &H56, I2cdaten(1),5 'Befehl wird gesendet
```

Motor einschalten

Kennung 1 Byte	Befehls- code 2. Byte	3 Byte	4. Byte	5. Byte
10	10	0 oder 1 oder 2 <i>0=linker Motor 1=rechter Motor 2=beide Motoren</i>		

Bascom-Beispiel:

```
Dim I2cdaten(6) As Byte
'Motorstrom auf 200mA begrenzen
I2cdaten(1) = 10 'Kennung muss bei RN-Motor immer 10 sein
I2cdaten(2) = 10 'Befehlscode
I2cdaten(3) = 0 '1 Parameter
I2csend &H56, I2cdaten(1),5 'Befehl wird gesendet
```

Motor Drehrichtung

Kennung 1 Byte	Befehls- code 2. Byte	3 Byte	4. Byte	5. Byte
10	4	0 oder 1 oder 2 <i>0=linker Motor 1=rechter Motor 2=beide Motoren</i>	0 oder 1 <i>0= links rum drehen 1= rechts rum drehen.</i>	

Bascom-Beispiel:

```
Dim I2cdaten(6) As Byte
I2cdaten(1) = 10 'Kennung muss bei RN-Motor immer 10 sein
I2cdaten(2) = 4 'Befehlscode
I2cdaten(3) = 0 '1 Parameter
I2cdaten(4) = 0 '2 Parameter
I2csend &H56, I2cdaten(1),5 'Befehl wird gesendet
```

Schrittggeschwindigkeit einstellen

Die Schrittggeschwindigkeit wird in Stufen von 0 bis 255 eingestellt. 0 ist dabei die schnellste und 255 die langsamste Schrittggeschwindigkeit. Die Geschwindigkeit ist von der Firmware (Betriebssystem von RN-Motor) abgänglich. In Version 1.0 ist die Einteilung wie unten angegeben (bei Firmware Updates kann das geändert werden)

Wichtig: Anzumerken ist noch, das nicht alle Schrittmotoren die hohen Geschwindigkeitsstufen 0 bis 4 erlauben. Wenn sich der Motor bei diesen Stufen nicht dreht, so ist zu prüfen ob auch der zulässige Maximalstrom eingestellt ist. Ist dies der Fall und der Motor dreht immer noch nicht, dann ist die Geschwindigkeit zu hoch für den Motor.

Kennung 1 Byte	Befehls- code 2. Byte	3 Byte	4. Byte	5. Byte
10	8	0 oder 1 oder 2 <i>0=linker Motor</i> <i>1=rechter Motor</i> <i>2=beide Motoren</i>	0 bis 255 <i>0= 1000 Schritte pro Sek.</i> <i>1= 500 Schritte pro Sek.</i> <i>2= 333 Schritte pro Sek.</i> <i>3= 250 Schritte pro Sek.</i> <i>4= 200 Schritte pro Sek.</i> <i>5= 167 Schritte pro Sek</i> <i>6= 142 Schritte pro Sek</i> <i>7= 125 Schritte pro Sek</i> <i>8= 111 Schritte pro Sek</i> <i>9= 100 Schritte pro Sek</i> ... <i>255= 4 Schritte pro Sekunde</i> <i>(geringe Abweichungen sind möglich)</i>	

Bascom-Beispiel:

```
Dim I2cdaten(6) As Byte
I2cdaten(1) = 10 'Kennung muss bei RN-Motor immer 10 sein
I2cdaten(2) = 8 'Befehlscode
I2cdaten(3) = 0 '1 Parameter
I2cdaten(4) = 5 '2 Parameter
I2csend &H56, I2cdaten(1),5 'Befehl wird gesendet
```

Motor endlos drehen bis Stopp- oder Ausschalt-Befehl kommt

Die ausgeführten Schritte werden dabei mitgezählt und können über den Befehl p abgerufen werden.

Kennung 1 Byte	Befehls- code 2. Byte	3 Byte	4. Byte	5. Byte
10	6	0 oder 1 oder 2 <i>0=linker Motor</i> <i>1=rechter Motor</i> <i>2=beide Motoren</i>		

Bascom-Beispiel:

```
Dim I2cdaten(6) As Byte
I2cdaten(1) = 10 'Kennung muss bei RN-Motor immer 10 sein
I2cdaten(2) = 6 'Befehlscode
I2cdaten(3) = 0 '1 Parameter
I2csend &H56, I2cdaten(1),5 'Befehl wird gesendet
```

Motor stoppen aber Strom eingeschaltet lassen (Schrittposition wird gehalten)

Die bisher ausgeführten Schritte können auch nach Stop über den Befehl p abgerufen werden.

Kennung 1 Byte	Befehls- code 2. Byte	3 Byte	4. Byte	5. Byte
10	3	0 oder 1 oder 2 <i>0=linker Motor</i> <i>1=rechter Motor</i> <i>2=beide Motoren</i>		

Bascom-Beispiel:

```
Dim I2cdaten(6) As Byte
I2cdaten(1) = 10 'Kennung muss bei RN-Motor immer 10 sein
I2cdaten(2) = 3 'Befehlscode
I2cdaten(3) = 0 '1 Parameter
I2csend &H56, I2cdaten(1),5 'Befehl wird gesendet
```

Motor eine bestimmte Anzahl von Schritten drehen

Kennung 1 Byte	Befehls- code 2. Byte	3 Byte	4. Byte	5. Byte
10	5z	0 oder 1 oder 2 <i>0=linker Motor</i> <i>1=rechter Motor</i> <i>2=beide Motoren</i>	Low Byte der Schrittzahl <i>also Schrittzahl mod 256</i>	High Byte der Schrittzahl <i>also Schrittzahl /</i> <i>256</i>

Bascom-Beispiel:

```
Dim I2cdaten(6) As Byte
I2cdaten(1) = 10 'Kennung muss bei RN-Motor immer 10 sein
I2cdaten(2) = 5 'Befehlscode
I2cdaten(3) = low(schritte) '1 Parameter
I2cdaten(4) = high(schritte) '2 Parameter
I2csend &H56, I2cdaten(1),5 'Befehl wird gesendet
```

Motor einen einzigen Schritt drehen

Kennung 1 Byte	Befehls- code 2. Byte	3 Byte	4. Byte	5. Byte
10	7	0 oder 1 oder 2 <i>0=linker Motor</i> <i>1=rechter Motor</i> <i>2=beide Motoren</i>		

Bascom-Beispiel:

```
Dim I2cdaten(6) As Byte
I2cdaten(1) = 10 'Kennung muss bei RN-Motor immer 10 sein
I2cdaten(2) = 7 'Befehlscode
I2cdaten(3) = 0 '1 Parameter
I2csend &H56, I2cdaten(1),5 'Befehl wird gesendet
```

Motor ganz ausschalten

Kennung 1 Byte	Befehls- code 2. Byte	3 Byte	4. Byte	5. Byte
10	9	0 oder 1 oder 2 <i>0=linker Motor</i> <i>1=rechter Motor</i> <i>2=beide Motoren</i>		

Bascom-Beispiel:

```
Dim I2cdaten(6) As Byte
I2cdaten(1) = 10 'Kennung muss bei RN-Motor immer 10 sein
I2cdaten(2) = 9 'Befehlscode
I2cdaten(3) = 0 '1 Parameter
I2csend &H56, I2cdaten(1),5 'Befehl wird gesendet
```

Konfiguration anzeigen

Einige Konfigurationswerte/Einstellungen werden über RS232 angezeigt

Kennung 1 Byte	Befehls- code 2. Byte	3 Byte	4. Byte	5. Byte
10	11t			

Bascom-Beispiel:

```
Dim I2cdaten(6) As Byte
I2cdaten(1) = 10 'Kennung muss bei RN-Motor immer 10 sein
I2cdaten(2) = 11 'Befehlscode
I2csend &H56, I2cdaten(1),5 'Befehl wird gesendet
```

Sleep Modus

Das Board wird in den Ruhemodus versetzt. Motoren und Controller werden auf Stromsparmodes gestellt. Das aufwecken ist nur über RESET möglich. Reset kann über einen Controller-Port erfolgen, wenn dieser mit der Reset-Leitung verbunden wird. Oder über Tastendruck bzw. Unterbrechung der Stromversorgung

Kennung 1 Byte	Befehls- code 2. Byte	3 Byte	4. Byte	5. Byte
10	12			

Bascom-Beispiel:

```
Dim I2cdaten(6) As Byte
I2cdaten(1) = 10 'Kennung muss bei RN-Motor immer 10 sein
I2cdaten(2) = 12 'Befehlscode
I2csend &H56, I2cdaten(1),5 'Befehl wird gesendet
```

Slave ID Verändern

Wenn mehrere RN-MOTOR Board's an einem I2C-Bus betrieben werden sollen, so müssen alle unterschiedliche SlaveID's haben. Über diesen Befehl läßt sich die Slave ID in dem Bereich Hex 50 bis Hex 70 ändern.

Achtung: Dies ist erst ab RN-MOTOR ST Betriebssystem 1.1 möglich. Die Betriebssystem Version und die aktuelle Slave ID kann über RS232 ausgegeben werden. Dazu einfach die RS232 Schnittstelle des Board's über ein geeignetes Kabel mit dem PC verbinden. Ein Terminalprogramm mit 9600 Baud Einstellung laden und RESET am Board klicken.

Kennung 1 Byte	Befehls- code 2. Byte	3 Byte	4. Byte	5. Byte
10	15	99	99	slaveid

Bascom-Beispiel:

```
Dim I2cdaten(6) As Byte
I2cdaten(1) = 10 'Kennung muss bei RN-Motor immer 10 sein
I2cdaten(2) = 15 'Befehlscode
I2cdaten(3) = 99 'Sicherheitskennung
I2cdaten(4) = 99 'Sicherheitskennung
I2cdaten(5) = &H58 'Neue SLAVE ID
I2csend &H56, I2cdaten(1),5 'Befehl wird gesendet
```

Motor selektieren um Schritte abzurufen

Die Schritte werden abgerufen indem von RN-Motor ein Long-Wert als 4 Bytes abgerufen werden. Damit RN-Motor weiss von welchem Motor die Schritte abgerufen werden sollen, muss dieser über diesen Selektionsbefehl angewählt werden. Das Beispiel demonstriert die Anwahl des linken Motors und den Schritte-Abruf..

Kennung 1 Byte	Befehls- code 2. Byte	3 Byte	4. Byte	5. Byte
10	13	0 oder 1 <i>0=linker Motor</i> <i>1=rechter Motor</i>		

Bascom-Beispiel:

```
Dim Schritte As Long
```

```
Dim Temp As Byte
```

```
Dim Ltemp As Long
```

```
Dim Umdrehungen As Single
```

```
Dim Fahrstrecke As Word
```

```
Dim I2cdaten(6) As Byte
```

```
'Erst mal Motor wählen, wo Schritte abgerufen werden sollen
```

```
I2cdaten(1) = 10 'Kennung muss bei RN-Motor immer 10 sein
```

```
I2cdaten(2) = 13 'Befehlscode
```

```
I2cdaten(3) = 0 'Linker Motor
```

```
I2csend &H56, I2cdaten(1),5 'Befehl wird gesendet
```

```
I2cstop
```

```
I2cstart
```

```
I2cwbyte &H57
```

```
I2crbyte Temp , Ack
```

```
Schritte = Temp
```

```
I2crbyte Temp , Ack
```

```
Ltemp = Temp * 256
```

```
Schritte = Schritte + Ltemp
```

```
I2crbyte Temp , Ack
```

```
Ltemp = Temp * 65536
```

```
Schritte = Schritte + Ltemp
```

```
I2crbyte Temp , Nack
```

```
Ltemp = Temp * 16777216
```

```
Schritte = Schritte + Ltemp
```

```
I2cstop
```

```
Print "Schrittzahl:" ; Schritte
```

```
Umdrehungen = Schritte / 200 'Wenn Schrittwinkel 1,8 Grad beträgt
```

```
Print "Umdrehungen: " ; Umdrehungen
```

```
Fahrstrecke = Umdrehungen * 31.4 'bei 10 cm Raddurchmesser
```

```
Print "Der Roboter ist " ; Fahrstrecke ; " cm gefahren "
```



Aufbau und Anwendung

Aufbau

Der Aufbau der Schaltung ist durch die vorgefertigte Platine bzw. den Bausatz (über <http://www.robotikhardware.de> beziehbar) eigentlich völlig problemlos auch von Elektronik-Einsteigern zu bewerkstelligen. Durch den Bestückungsdruck und die Bestückungsliste, etwas weiter hinten in dieser Dokumentation, ist der Aufbau unkritisch.

Aufgrund moderner Bauteile ist es gelungen, das das Board nur relativ wenig Bauteile besitzt. Auf Logikbausteine konnte verzichtet werden, da das Board über einen eigens dafür programmierten Controller verfügt. Dieser spezielle Controller RNST01RS oder RNST01I2C ist im Bausatz enthalten, kann jedoch auch einzeln bestellt werden.

Der Typ des Controllers bestimmt auch die Ansteuerung des Boards.:

Wird der Typ RNST01RS eingesetzt, so wird das Board über RS232 (PC-Pegel oder TTL-Pegel) angesteuert. Der I2C-Bus wird in diesem Fall nicht ungenutzt.

Wird der Typ RNST01I2C eingesetzt, so wird das Board über den I2C-Bus angesteuert. Die RS232-Schnittstelle wird in diesem Fall nicht genutzt, lediglich einige Infos kann man dort ausgeben anzeigen lassen.

Die Schaltung benötigt in der Regel eine Aufbauzeit von ca. 30 – 60 Minuten.

Dennoch einige Anmerkungen zu kleinen Hürden:

1. Nicht vergessen das in die Fassung wo MEGA8 steht, der oben erwähnte programmierte Spezialchip eingesetzt werden muß. Diesen gibt es fix und fertig programmiert bei robotikhardware.de.
Versierte Programmierer/Experten, können natürlich auch ein eigenes Betriebssystem für das Board entwickeln.
2. Beim Einlöten der Motortreiber IC´s L298 ist zu bedenken das diese eine genaue Höhe einnehmen müssen, damit diese später bequem an den montierten Kühlkörper geschraubt werden können.
Daher ist es empfehlenswert zuerst den Kühlkörper zu montieren. Der Kühlkörper wird durch zwei kurze 3mm Schrauben von der Unterseite der Platine angeschraubt. Er paßt genau auf die vordefinierten Löcher.
Wichtig: Zwischen Kühlkörper und Platine müssen zwei Plastikunterlegscheiben untergelegt werden, damit der Kühlkörper ca. 1mm Abstand zur Platine hat. Dies ist wichtig damit er keine Verbindung zu den darunterliegenden Leiterbahnen hat.
Hat man den Kühlkörper angeschraubt, kann man gleich die beiden L298 ebenfalls mit kleinen 3mm Schrauben anschrauben. Auch hier sind die notwendigen Gewinde vorhanden.
Wenn beides montiert ist, kann man die L298 einlöten und gegebenenfalls den Kühlkörper wieder entfernen und die restlichen Bauelemente bestücken.
3. Es muss in jedem Fall der mitgelieferte 16 Mhz Quarze bestückt werden. Andere Taktraten sind nicht möglich.
4. Achten Sie darauf das die Taster richtig herum eingelötet ist. Achten Sie genau auf den Bestückungsplan in dieser Anleitung.
5. Beim Einlöten der LED, sollte der lange Anschlußdraht auf der Seite wo ANODE steht sein.

Das waren eigentlich schon die besonderen Punkte die zu beachten sind. Ansonsten natürlich sauber mit einem 15 – 25 W LötKolben alles auf der Unterseite verlöten. Grundkenntnisse beim Löten werden empfohlen.

nach dem Aufbau sollten Sie nochmals alle Lötunkte kontrollieren. Wenn Sie dann Spannung anlegen, dann sollten deutlich weniger als 100mA Strom fließen. Ist der Strom höher, dann deutet das auf ein Lötfehler hin.

Wenn Sie die RS232 Version aufgebaut haben, dann ist auf der CD ein PC-Programm zum steuern des Boards enthalten. Verbinden Sie das Board über ein RS232-Kabel (passende gibt es bei Robotikhardware.de) und installieren Sie das PC-Programm RN-Motor. Damit können Sie dann alle Funktionen austesten.

Haben Sie die I2C-Version aufgebaut, dann benötigen Sie ein I2C-Master mit einem Testprogramm. Ein solches Testprogramm läßt sich mit Hilfe der Befehlstabelle der Seiten zuvor recht einfach schreiben.

Für das inzwischen schon recht weit verbreitete Board RN-Control wird ein Beispielprogramm mitgeliefert. Über die Taster können Sie damit verschiedene Funktionen des RN-Motor-Boards aktivieren.

Nochmal ganz wichtig:

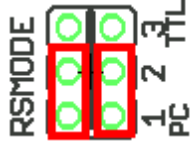
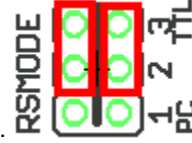
Niemals den I2C-Bus Stecker versehendlich in die vordere ISP-Buchse stecken. Dadurch wird fast immer Ihr Motorboard als auch Controllerboard beschädigt.

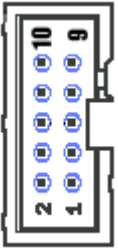
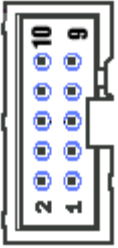
Immer genau darauf achten das der I2C-Stecker auch in die I2C-Bus Buchse gesteckt wird. Die I2C-Buchse befindet sich am linken Rand des Motorboards.

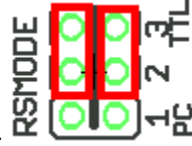
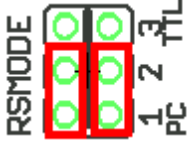
**Beachten Sie die Nachträge auf den letzten Seiten dieser Anleitung.
Dies betrifft auch Käufer des Fertigboards mit
Platinenrevision 1.0 oder 1.2.**

Erläuterung der Anschlüsse, Regler und Kurzschlussbrücken

Anschluss- Bezeichnung	Erläuterung
Motor Links	<p>Anschluss des linken Schrittmotors Über diese 4 polige Schraubklemme wird der linke Schrittmotor angeschlossen. Dabei ist darauf zu achten das eine Wicklung an die beiden linken und eine Wicklung an die beiden rechten Schraubklemmen angeschlossen wird. Dreht sich später ein Motor nicht, so muß eine Wicklung umgepolt werden. Dreht der Motor falsch herum, dann Wicklungen umtauschen.</p> <p>Die Motoren werden bipolar angesteuert. Das bedeutet das eine eventuelle Mittelanzapfung an der Motorwicklung nicht benötigt wird.</p> <p>z.B. Anschluß eines 7V Motors Typ: 17PS-C058 (wie er von robotikhardware.de angeboten wird): Klemmenbezeichnung 10: braun Klemmenbezeichnung 20: orange Klemmenbezeichnung 30: rot Klemmenbezeichnung 40: gelb</p> <p>z.B. Anschluß eines 1,9V Motors Typ: 23LQ-C202-01H (wie er von robotikhardware.de angeboten wird): Klemmenbezeichnung 10: gelb Klemmenbezeichnung 20: blau Klemmenbezeichnung 30: braun Klemmenbezeichnung 40: rot</p> <p>Es können Motoren von Nennspannungen ab 1V bis zur Betriebsspannung (max. 22V angeschlossen werden). Dies ist deshalb möglich, da später vor dem einschalten zuerst mit einem Befehl der maximale Nennstrom übermittelt wird. Dadurch überwacht das Board automatisch die richtige Motorspannung, ein großer Vorteil gegenüber einfachen Schrittmotoransteuerungen.</p> <p>Wird der Nennstrom nicht eingestellt, so ist dieser auf maximal 100mA begrenzt. Dies kann bis zu 2A je Motor erhöht werden.</p>
Motor Rechts	<p>Anschluss des rechten Schrittmotors Über diese 4 polige Schraubklemme wird der rechte Schrittmotor angeschlossen. Dabei ist darauf zu achten das eine Wicklung an die beiden linken und eine Wicklung an die beiden rechten Schraubklemmen angeschlossen wird. Dreht sich später ein Motor nicht, so muß eine Wicklung umgepolt werden.</p> <p>Die Motoren werden bipolar angesteuert. Das bedeutet das eine eventuelle Mittelanzapfung an der Motorwicklung nicht benötigt wird.</p> <p>z.B. Anschluß eines 7V Motors Typ: 17PS-C058 (wie er von robotikhardware.de angeboten wird): Klemmenbezeichnung 10: braun Klemmenbezeichnung 20: orange Klemmenbezeichnung 30: rot Klemmenbezeichnung 40: gelb</p> <p>z.B. Anschluß eines 1,9V Motors Typ: 23LQ-C202-01H (wie er von robotikhardware.de angeboten wird): Klemmenbezeichnung 10: gelb Klemmenbezeichnung 20: blau Klemmenbezeichnung 30: braun Klemmenbezeichnung 40: rot</p> <p>Es können Motoren von Nennspannungen ab 1V bis zur Betriebsspannung (max. 22V angeschlossen werden). Dies ist deshalb möglich, da später vor dem einschalten zuerst mit einem Befehl der maximale Nennstrom übermittelt wird. Dadurch überwacht das Board automatisch die richtige Motorspannung, ein großer Vorteil gegenüber einfachen Schrittmotoransteuerungen.</p> <p>Wird der Nennstrom nicht eingestellt, so ist dieser auf maximal 100mA begrenzt. Dies kann bis zu 2A je Motor erhöht werden.</p>

<p>Power</p>	<p>Spannungsversorgung Über diese Schraubklemme wird das Board mit Spannung versorgt. Es reicht eine ungestabilisierte Gleichspannung von 7 bis 22V aus (ab ca. 17V sollte man Spannungsregler kühlen, z.B. mit Wärmeleitkleber am oberen Kühlkörper ankleben. + und – sind auf der Platine markiert.</p>
<p>RS232</p>	<p>PC kompatible RS232 Schnittstelle Über ein Adapterkabel kann die serielle Schnittstelle des PC direkt mit dem Board verbunden werden. Dadurch können bei der RS232-Version des Boards alle Motorfunktionen per PC gesteuert werden. Ein entsprechendes Steuerprogramm ist auf der CD enthalten. Hat man ein Controllerboard wie RN-Control, so kann man dieses ebenfalls über eine dreipolige Leitung miteinander verbinden.</p> <p>Die Belegung entspricht der Roboternetz-Norm und ist kompatibel zum Conrad Roboter CCRP5:</p> <p>Pin 1 RX Pin 2 GND Pin 3 TX</p> <p>Ein geeignetes Anschlußkabel kann schnell selbst angefertigt werden oder gibt es bei robotikhardware.de bereits fertig</p> <p>Wichtig: Wenn diese RS232-Buchse benutzt wird, dann müssen die zwei Kurzschlussbrücken bei Jumper RSMODE wie folgt gesteckt sein:</p>  <p>Hat ihr Board (z.B. Version 1.0) keinen solchen Jumper RSMODE, dann reicht es wenn Sie darauf achten das das IC MAX232 eingesteckt ist.</p>
<p>TTL RS232</p>	<p>RS232 Schnittstelle mit TTL Pegel Dies ist die RS232 Schnittstelle mit TTL-Pegel, also max. 5V Pegel. Hier lassen sich Controllerboards, die keinen Treiberbaustein wie Max232 besitzen, direkt anschließen</p> <p>Die TTL- Stiftleiste wird nach Roboternetz-Definition immer 4 polig ausgestattet um Verwechslungen mit der Standard RS232 Stiftleiste zu vermeiden. Verbinden Sie niemals eine 4 polige TTL-Stiftleiste mit einer 3 poligen, dies würde den Controller oder PC beschädigen.</p> <p>Die Belegung entspricht der Roboternetz-Norm:</p> <p>Pin 1 RX Pin 2 TX Pin 3 GND Pin 4 +5V</p> <p>Ein geeignetes Anschlußkabel kann schnell selbst angefertigt werden.</p> <p>Wichtig: Wenn die TTL-Buchse benutzt wird, dann müssen die zwei Kurzschlussbrücken bei Jumper RSMODE wie folgt gesteckt sein</p>  <p>Hat ihr Board (z.B. Version 1.0) keinen solchen Jumper RSMODE, dann reicht es wenn Sie das IC MAX232 einfach aus der Fassung nehmen.</p>

<p>I2C-Bus</p> 	<p>I2C-Bus Wenn Sie die I2C-version des Boards gebaut haben, also den entsprechenden Controller eingesteckt haben, dann ist diese I2C-Buchse für die Steuerung zuständig. Der I2C-Bus benötigt nur 2 Leitungen für alle Funktionen. Entsprechend der Roboternetz-Norm wird hier ein 2x5 poliger Stecker angeschlossen. Die Belegung entspricht exakt der von allen Roboternetz-Boards:</p> <p>Pin 1 SCL (Taktleitung) Pin 3 SDA (Datenleitung) Pin 5 +5V (Kann über Jumper I2C5V getrennt werden) Pin 7 +5V (Kann über Jumper I2C5V getrennt werden) Pin 9 Batteriespannung Diese Leitung wird jedoch bei RN-Motor nicht benutzt Pin 2,4,6,8 GND Pin 10 INT Diese Leitung wird jedoch bei RN-Motor nicht benutzt</p>
<p>ISP</p> 	<p>ISP – IN SYSTEM PROGRAMMING Über diesen Anschluß kann der Controller auf dem Motorboard mit einem Standard ISP-Kabel direkt an einen Parallelport des PC´s angeschlossen und programmiert werden.</p> <p>Achtung: Dies ist in der Regel nicht notwendig, da ja der Controller bereits programmiert ist. Dieser Anschluß sollte deshalb nur von erfahreneren AVR-Programmierern benutzt werden.</p> <p>Die Belegung des ISP-Anschlusses ist zu dem weit verbreitetet STK200 Programmier Dongle kompatibel. Ein entsprechender Dongle kann man sich entweder selber basteln (siehe Artikel „ISP-Programmieradapter“ unter www.roboternetz.de) oder fertig bestellen (z.B. www.robotikhardware.de).</p> <p>Pin 1 MOSI Pin 2 VCC Pin 3 Nicht belegt Pin 4 GND Pin 5 RESET Pin 6 GND Pin 7 SCK Pin 8 GND Pin 9 MISO Pin 10 GND</p>
<p>Wecken</p>	<p>Dieser Anschluß wird derzeit vom Betriebssystem nicht unterstützt und sollte unbelegt bleiben.</p>
<p>JRESET</p>	<p>RESET Über diesen Anschluß kann ein RESET, also ein zurücksetzen des Bords erfolgen. Man kann beispielsweise die Reset-Leitung über diesen Anschluß mit einem Port eines Steuerboards (z.B. RN-Control / C-Control) verbinden. Wenn das Motorboard in den Sleep-Mode versetzt wurde (Stromsparmodus), könnte man es so wieder aufwecken.</p> <p>Pin1: GND Pin2 Reset-Leitung</p>

RSMODE	<p>Jumper zum Einstellen der RS232 Schnittstelle Über diesen Jumper wird festgelegt ob die normale RS232 Stiftleiste oder die 4 polige TTL-RS232 Stiftleiste aktiv ist.</p> <p>Wenn die TTL-Buchse benutzt wird, dann müssen die zwei Kurzschlussbrücken bei Jumper RSMODE wie folgt gesteckt sein</p>  <p>Hat ihr Board (z.B. Version 1.0) keinen solchen Jumper RSMODE, dann reicht es wenn Sie das IC MAX232 einfach aus der Fassung nehmen.</p> <p>Wenn diese RS232-Buchse benutzt wird, dann müssen die zwei Kurzschlussbrücken bei Jumper RSMODE wie folgt gesteckt sein:</p>  <p>Hat ihr Board (z.B. Version 1.0) keinen solchen Jumper RSMODE, dann reicht es wenn Sie darauf achten das das IC MAX232 eingesteckt ist.</p>
JP1	<p>Getriebemotor Parallelmodus für Motor 1 und 2 Dieser Jumper ist für ein Getriebemodus-Betriebssystem vorgesehen. Der Erläuterung wird später in der Anleitung für die Getriebemotorsteuerung folgen. Wichtig: Im Schrittmotor-Betrieb muss immer auf der Innenseite wo PWM steht, ein Kurzschlussstecker aufgesteckt werden.</p>
JP2	<p>Getriebemotor Parallelmodus für Motor 3 und 4 Dieser Jumper ist für ein Getriebemodus-Betriebssystem vorgesehen. Der Erläuterung wird später in der Anleitung für die Getriebemotorsteuerung folgen. Wichtig: Im Schrittmotor-Betrieb muss immer auf der Innenseite wo PWM steht, ein Kurzschlussstecker aufgesteckt werden.</p>
I2C5V	<p>5V vom I2C Bus verwenden Gewöhnlich legt das Masterboard bereits 5V auf den I2C-Bus. Wichtig: In diesem fall darf dieser Jumper nicht eingesteckt sein.</p> <p>Der Jumper ist nur dann einzustecken, wenn kein anderes Board 5V auf den I2C-Bus legt und man andere Boards somit über den I2C-Bus mit Strom versorgen möchte.</p> <p>In Platinenversion 1.0 von RN-Motor heißt dieser Jumper JP4.</p>
JP5	<p>Motorstrom messen Diesen Anschluß gibt es nur noch in der Platinenversion 1.0. Er diente der Entwicklung der Betriebssystemsoftware. Hier konnte der abfallende Strom an den Zementwiderständen gemessen werden. Für die praktische Anwendung ist dieser unbedeutend.</p>
TASTER RESET	<p>Motorboard wird in Grundstellung versetzt Motoren werden ausgeschaltet und Motorstrom wird auf 100mA begrenzt</p>
LED1	<p>Die Leuchtdiode signalisiert verschiedene Dinge</p> <ol style="list-style-type: none"> 1. Ein RESET wird durch kurzes Blinken der LED angezeigt 2. Programmübertragungen per ISP werden durch Flackern signalisiert 3. Wenn ein Motor läuft und die Betriebsspannung nicht ausreicht um den Nennstrom zur erreichen, leuchtet die LED auf.

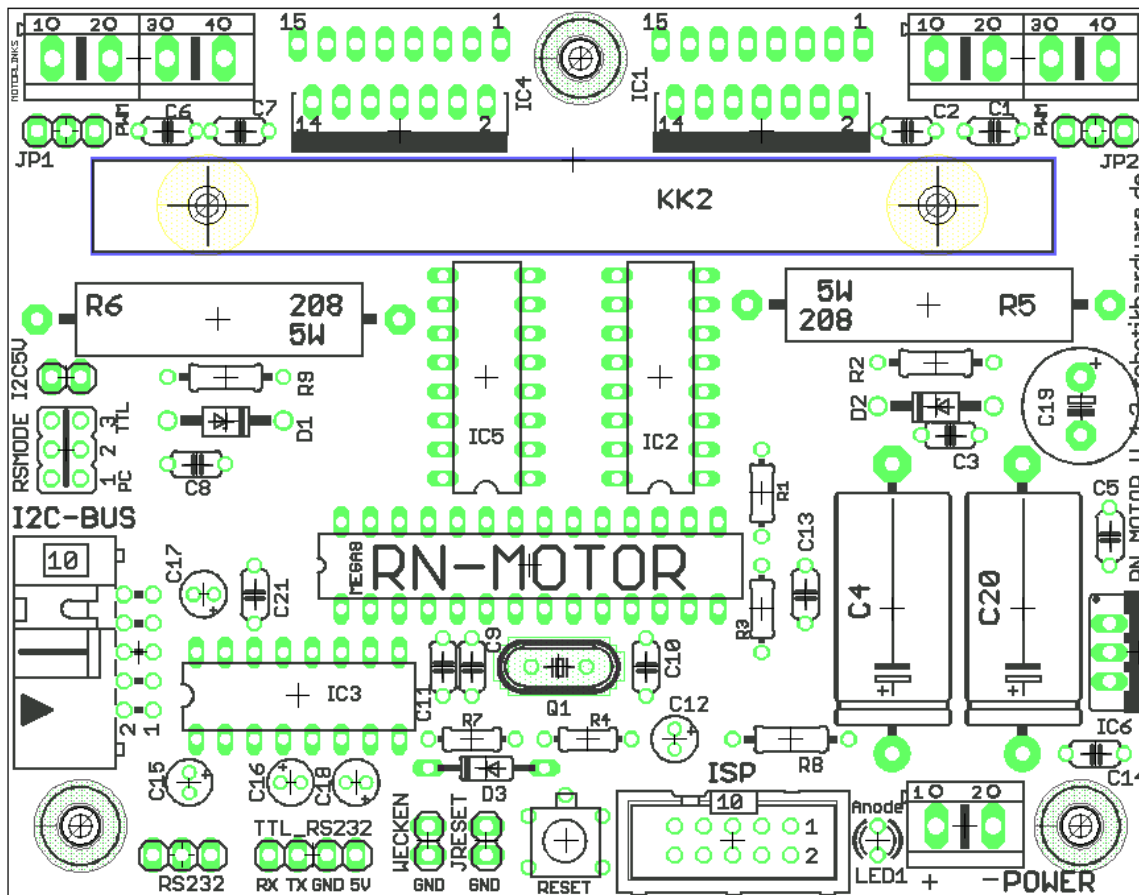
Bauteile Bestell- und Bestückungsliste für RN-Motor Platine Version 1.2

Angaben ohne Gewähr

Bauteil	Wert	Beschreibung	Reichelt Best.Nr.
C1	100n	Keramik Kondensator	KERKO100N
C2	100n	Keramik Kondensator	KERKO100N
C3	100n	Keramik Kondensator	KERKO100N
C4	470uF 25V	Elko 470uF max 12mm hoch	ax 470/25
C5	100n	Keramik Kondensator	KERKO100N
C6	100n	Keramik Kondensator	KERKO100N
C7	100n	Keramik Kondensator	KERKO100N
C8	100n	Keramik Kondensator	KERKO100N
C9	22pf	Keramik Kondensator	KERKO-500 22p
C10	22pf	Keramik Kondensator	KERKO-500 22p
C11	100n	Keramik Kondensator	KERKO100N
C12	1uF	Elko	SM 1,0/63RAD
C13	100n	Keramik Kondensator	KERKO100N
C14	100n	Keramik Kondensator	KERKO100N
C15	4,7uF	Elko	SM 4,7/50RAD
C16	4,7uF	Elko	SM 4,7/50RAD
C17	4,7uF	Elko	SM 4,7/50RAD
C18	4,7uF	Elko	SM 4,7/50RAD
C19	220uF	Elko	RAD 220/35
C20	470uF 25V	Elko 470uF max 12mm hoch	ax 470/25
C21	100n	Keramik Kondensator	KERKO100N
D1	ZPY	2,7V Zehnerdiode 2 Watt	ZD 2,7
D2	ZPY	2,7V Zehnerdiode 2 Watt	ZD 2,7
D3	1N4148	Diode	1n 4148
I2C-BUS		Wannebuchse Gewinkelt	WSL 10W
I2C5V		Stiftleiste 2polig	LU 2,5 MS2
IC1	L298	Motortreiber L298	L 298
IC2	L6210	Freilaufdiodenbrücke	L 6210
IC3	MAX232	RS232 Treiber	MAX 232 CPE
IC4	L298	Motortreiber L298	L 298
IC5	L6210	Freilaufdiodenbrücke	L 6210
IC6	78S05	Spannungsregler	78S05
ISP	ML10	Wannenbuchse	WSL 10G
JP1	PINHD-1X3	Stiftleiste 3 polig	LU 2,5 MS3
JP2	PINHD-1X3	Stiftleiste 3 polig	LU 2,5 MS3
JRESET	PINHD-1X2	Stiftleiste 2 polig	LU 2,5 MS2
KK2	SK96/84	Kühlkörper	v 6716z
LED1	LED3MM	Leuchdiode Low	LED 3MM 2MA GN
MEGA8	MEGA8-P	Programmierter Spezialcontroller von Robotikhardware.de	
MOTORLINKS	AK500/4	Schraublemme 4 polig	AKL 101-04
MOTORRECHTS	AK500/4	Schraublemme 4 polig	AKL 101-04
POWER	AK500/2	Schraubklemme 2 polig	AKL 101-02
Q1	16Mhz	Quarz 16 Mhz	16-HC18
R1	10k	Widerstand 10k Toleranz max. 1%	METALL 10,0K
R2	1000	Widerstand 1k Toleranz max. 1%	METALL 1,00K
R3	10k	Widerstand 10k Toleranz max. 1%	METALL 10,0K
R4	1k	Widerstand 1k	1/4W 1k
R5	0,5	Drahtwiderstand 0,5 Ohm	5W Axial 0,51
R6	0,5	Drahtwiderstand 0,5 Ohm	5W Axial 0,51
R7	100k	Widerstand 100k	1/4W 100k
R8	1k	Widerstand 1K	1/4W 1,0K
R9	1000	Widerstand 1k Toleranz max. 1%	METALL 1,00K
RESET	TASTER330	Minitaster liegend	TASTER 3301
RS232		Stiftleiste 3 polig	LU 2,5 MS3
RSMODE	JP3Q	Jumper Stiftleiste 2x3	Stiftl. 2x50g (teilen)
TTL_RS232		Stiftleiste 4polig	LU 2,5 MS4
WECKEN		Stiftleiste 2 polig	LU 2,5 MS2

Bestückungsplan

Achtung, dieser Bestückungsplan gilt für die Version 1.2 der Platine. Es werden auch einige Fertigboards mit etwas anderer Platine ausgeliefert, dort weicht die Bestückung geringfügig ab. Die Funktion der Boards ist jedoch identisch.

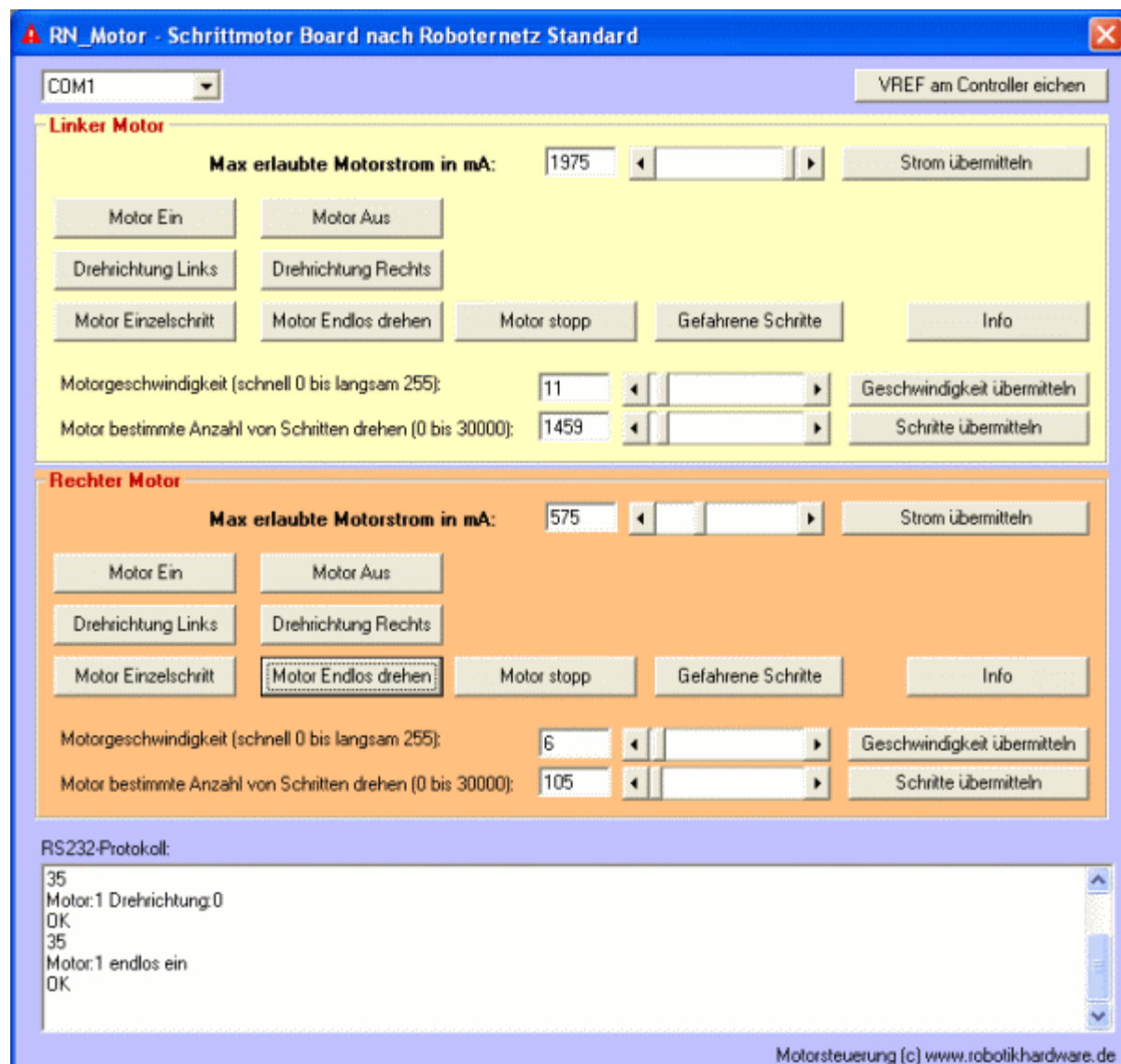


Inbetriebnahme des Motor-Boards

Nachdem Board aufgebaut bzw. ein Fertigboard erworben haben, wird hier kurz erläutert wie Sie das Board in Betrieb nehmen.

Beispiel bei der RS232-Version des Boards:

1. Zunächst vergewissern Sie sich das alle Jumper korrekt entsprechend der vorhergehenden Beschreibung gesteckt sind. Bei der RS-232-Version achten Sie besonders auf Jumper RSMODE.
2. Dann sollten Sie das Board über die 3 polige Stiftleiste mit der RS232-Schnittstelle des PC's verbinden. Ein passendes Kabel kann selbst angefertigt oder bei Robotikhardware bestellt werden.
3. Versorgen Sie nun das Board mit einer Gleichspannung zwischen 7 und 22V über den POWER Eingang. Achten Sie darauf das Sie auf richtige Polung!
4. Schließen Sie die Motoren an
5. Anschließend installieren Sie auf ihrem PC das PC-Programm RN-Motor (befindet sich auf der mitgelieferten CD).
6. Rufen Sie das installierte Steuerprogramm auf



Das Steuerprogramm ist sehr einfach aufgebaut, eigentlich erklären sich alle Funktionen von selbst. Wichtig ist, das Sie als erstes den Motorstrom korrekt einstellen. Achten Sie daher auf die Angaben auf Ihrem Motor und stellen Sie den Phasenstrom dann mit dem oberen Schieberegler ein. Es ist nicht schlimm wenn Sie weniger Strom einstellen, der Motor hat dann nur weniger Kraft und schafft eventuell nicht mehr die höchsten Drehzahlen. Sie sollten jedoch den zulässigen Nennstrom nicht überschreiten.

Ist auf dem Motor keine Stromangabe vorhanden, so kann dieser auch über den Wicklungswiderstand und die Nennspannung berechnet werden ($I=U/R$).

Ist der Strom richtig eingestellt, dann klicken Sie auf die Schaltfläche "Strom übermitteln". Erst durch diesen Button wird die Einstellung an das Motorboard übertragen. Im unteren Protokollfenster sollte das Motorboard ein "OK" zurückmelden. Dies bedeutet das es die Einstellung korrekt vorgenommen hat.

Nun können Sie beispielsweise über die Schaltfläche "Motor Ein" den Motor aktivieren. Ein eingeschalteter Schrittmotor dreht sich natürlich noch nicht, aber er hält die Position. Sie können dies einfach nachvollziehen indem Sie mal versuchen die Welle bei eingeschalteten und bei ausgeschalteten Motor mit der Hand zu drehen

Nachdem Sie den Motor eingeschaltet haben, können Sie beispielsweise über die Schaltfläche "Motor endlos drehen" den Motor endlos in eine Richtung drehen lassen. Die Richtung kann dabei jederzeit über die Schaltfläche Drehrichtung Links / Rechts geändert werden.
Sollte der Motor nicht richtig oder garnicht anlaufen, dann ist entweder der Motorstrom zu gering eingestellt oder aber die Geschwindigkeit ist zu hoch.
Die höchsten Geschwindigkeitsstufen (beim Motorboard 0 und 1) schaffen viele Schrittmotoren nicht. Höhere geschwindigkeiten kann man auch dadurch erreichen das man die Motoren mit langsameren Geschwindigkeitsstufen anlaufen läßt.

Die Geschwindigkeit wird ebenfalls über einen Schieberegler eingestellt. Nicht vergessen das die Geschwindigkeit erst dann geändert wird, wenn die Schaltfläche "Geschwindigkeit übermitteln" eingestellt wird.

Klicken Sie auch ab und zu mal auf die Schaltfläche "Gefahrene Schritte". Dadurch wird die Anzahl der gefahrenen Schritte genau ausgegeben. Dies ist auch nach einem Stopp-Befehl noch möglich. Daraus läßt sich genau berechnen wie weit ihr Roboter gefahren ist.

Der Stop-Befehl stoppt im übrigen den Motor nur, er schaltet nicht den Haltestrom aus. Möchte man Energie sparen, sollte lieber der Motor ganz ausgeschaltet werden oder nach einem Stop der maximale Strom verringert werden.

Alle weiteren Funktionen erklären sich wohl von selbst. Anzumerken ist das mit der Software beide Motoren völlig unabhängig voneinander angesteuert werden können. Über die RS232-Befehle lassen sich aber auch immer Anweisungen gleichzeitig an beide Motoren geben (siehe Befehlsauflistung weiter vorne).

Die RS232-Befehle sind sehr einfach. Wenn etwas programmieren können, dann können Sie sich selbst eine eigene Steuersoftware für den PC oder einen Controller basteln. Als Hilfe ist auch der Quellcode der Motorsteuerung in Visual Basic auf CD.

Hier der komplette Quellcode. Sie sehen wie einfach die RS232-Befehle sind. Hier wird dafür der Befehl `SendBefehl` benutzt. Dies ist eine Routine die gleichzeitig prüft ob OK zurück kommt.

```
Option Explicit

Dim mybuffer As Variant
Dim low As Byte
Dim high As Byte

Private Sub buttLDrehLinks_Click()
    SendBefehl "#rmd" & Chr(0) & Chr(0), "OK"
End Sub

Private Sub buttLDrehRechts_Click()
    SendBefehl "#rmd" & Chr(0) & Chr(1), "OK"
End Sub

Private Sub buttLEndlosdrehen_Click()
    SendBefehl "#rme" & Chr(0), "OK"
End Sub

Private Sub buttLGeschwindigkeituebermitteln_Click()
    SendBefehl "#rmg" & Chr(0) & Chr(HScrolllgeschwindigkeit.Value), "OK"
End Sub

Private Sub buttLMotorAus_Click()
    SendBefehl "#rma" & Chr(0), "OK"
End Sub

Private Sub buttLMotorEin_Click()
    SendBefehl "#rmo" & Chr(0), "OK"
End Sub

Private Sub buttlmotorstop_Click()
    SendBefehl "#rms" & Chr(0), "OK"
End Sub
```

```

Private Sub buttLSchritt_Click()
    SendBefehl "#rmm" & Chr(0), "OK"
End Sub

Private Sub butttlSchritteuebermitteln_Click()
    high = Int(HScrolllschritte.Value / 256)
    low = HScrolllschritte.Value - (high * 256)
    SendBefehl "#rmz" & Chr(0) & Chr(low) & Chr(high), "OK"
End Sub

Private Sub buttlSchritteZeigen_Click()
    SendBefehl "#rmp" & Chr(0), "OK"
End Sub

Private Sub buttlStatus_Click()
    SendBefehl "#rmt", "OK"
End Sub

Private Sub buttlStrom_Click()
    SendBefehl "#rmi" & Chr(0) & Chr(lmaxstrom.Value / 10), "OK"
End Sub

Private Sub buttrDrehLinks_Click()
    SendBefehl "#rmd" & Chr(1) & Chr(0), "OK"
End Sub

Private Sub buttrDrehRechts_Click()
    SendBefehl "#rmd" & Chr(1) & Chr(1), "OK"
End Sub

Private Sub buttrEndlosdrehen_Click()
    SendBefehl "#rme" & Chr(1), "OK"
End Sub

Private Sub buttrGeschwindigkeituebermitteln_Click()
    SendBefehl "#rmg" & Chr(1) & Chr(rHScrolllgeschwindigkeit.Value), "OK"
End Sub

Private Sub buttrMotorAus_Click()
    SendBefehl "#rma" & Chr(1), "OK"
End Sub

Private Sub buttrMotorEin_Click()
    SendBefehl "#rmo" & Chr(1), "OK"
End Sub

Private Sub buttrmotorstop_Click()
    SendBefehl "#rms" & Chr(1), "OK"
End Sub

Private Sub buttrSchritt_Click()
    SendBefehl "#rmm" & Chr(1), "OK"
End Sub

Private Sub buttrSchritteuebermitteln_Click()
    high = Int(RHScrolllschritte.Value / 256)
    low = RHScrolllschritte.Value - (high * 256)
    SendBefehl "#rmz" & Chr(1) & Chr(low) & Chr(high), "OK"
End Sub

Private Sub buttrschritteZeigen_Click()
    SendBefehl "#rmp" & Chr(1), "OK"
End Sub

Private Sub buttrStatus_Click()
    SendBefehl "#rmt", "OK"
End Sub

Private Sub buttrStrom_Click()
    SendBefehl "#rmi" & Chr(1) & Chr(rmaxstrom.Value / 10), "OK"
End Sub

Private Sub buttvref_Click()
    frmVref.Show 1
End Sub

Private Sub Form_Load()
    txtStrom = ""
    txtLSchritte = ""
    txtlGeschwindigkeit = ""

    txtStromR = ""

```



```

txtRSchritte = ""
txtrGeschwindigkeit = ""

'txtStrom = lmaxstrom.Value
'txtLSchritte = HScrolllschritte.Value
'txtlGeschwindigkeit = HScrolllgeschwindigkeit.Value
txtlog = ""

ComboComPort.ListIndex = 0
MSComm1.CommPort = ComboComPort.ListIndex + 1
MSComm1.Settings = "9600,N,8,1"
MSComm1.InputLen = 0
MSComm1.PortOpen = True
End Sub

Private Sub Form_Unload(Cancel As Integer)
    MSComm1.PortOpen = False
End Sub

Private Sub HScrolllgeschwindigkeit_Change()
    txtlGeschwindigkeit = HScrolllgeschwindigkeit.Value
End Sub

Private Sub HScrolllschritte_Change()
    txtLSchritte = HScrolllschritte.Value
End Sub

Private Sub lmaxstrom_Change()
    txtStrom = lmaxstrom.Value
End Sub

Function SendBefehl(befehl As Variant, warteauf As String) As Boolean
    Dim buffer As String
    Dim rueckgabe As String
    Dim n As Long
    Dim bytefeld() As Byte
    Dim anz As Integer
    Dim i As Integer

    anz = Len(befehl)
    ReDim bytefeld(anz)
    For i = 1 To anz
        bytefeld(i - 1) = Asc(Mid(befehl, i, 1))
    Next i

    SendBefehl = False
    buffer = ""
    MSComm1.Output = befehl

    n = 0
    Do
        If n > 100000 Then
            Beep
            Exit Function
        End If
        rueckgabe = MSComm1.Input
        txtlog = txtlog & rueckgabe
        DoEvents
        buffer = buffer & rueckgabe
        n = n + 1
    Loop Until InStr(buffer, warteauf)

    txtlog.SelStart = Len(txtlog.Text)
    'txtlog.SelLength = 1
    DoEvents
    SendBefehl = True
End Function

Private Sub rHScrolllgeschwindigkeit_Change()
    txtrGeschwindigkeit = rHScrolllgeschwindigkeit.Value
End Sub

Private Sub RHScrolllschritte_Change()
    txtRSchritte = RHScrolllschritte.Value
End Sub

Private Sub rmaxstrom_Change()
    txtStromR = rmaxstrom.Value
End Sub

```

Ansteuerung per I2C

Wenn Sie das Board über den I2C-Bus ansteuern möchten, dann ist dies ohne Adapter nicht über den PC möglich. Die RS232 Schnittstelle müssen Sie auch nicht mit dem Board verbinden. Diese Betriebsart ist in erster Linie für Controller wie RN-Control, RNFRA, C-Control usw. gedacht. Fast alle modernen Controller verfügen über einen I2C-Bus und sind somit in der Lage das Board anzusteuern.

Ich möchte es daher am Beispiel des recht beliebten Boards RN-Control demonstrieren.

1. Zunächst sollten aus Sicherheitsgründen als erstes bei der I2C-Version mal der Kurzschlussstecker I2C5V (bzw. bei älteren Platinen JP4) entfernt werden, da gewöhnlich schon 5V auf dem Bus liegt.
2. Verbinden Sie nun die I2C-Buchse von RN-Motor mit der I2C-Buchse von RN-Control über eine 10polige Flachbandleitung. Es ist nicht schlimm wenn an dieser Flachbandleitung bereits einige andere Boards (wie RN-Speak, RN-Relais usw.) angeschlossen sind, da der I2C-Bus ja bis zu 128 Boards erlaubt. Eine Flachbandleitung mit 2 Steckern lässt sich selbst recht einfach konfektionieren oder fertig über robotikhardware.de beziehen.
3. Versorgen Sie nun das RN-Control und RN-Motor mit einer Gleichspannung zwischen 7 und 22V über den POWER Eingang. Achten Sie darauf das Sie auf richtige Polung!
4. Schließen Sie die Motoren an
5. Nun können Sie alle weiter vorn aufgelisteten i2C-Befehle in ihrem Bascom Programm verwenden. Die Handhabung ist ähnlich einfach wie bei den RS232-Befehlen. Als erstes sollte immer die Einstellung des maximalen Motorstroms und des Schrittmodus (Vollschritt- oder Halbschritt) erfolgen.

Damit das ganze anschaulicher wird, befindet sich auf der CD ein kleines Bascom-Programm für RN-Control. Dieses Programm legt verschiedene Funktionen auf die 5 Tasten, siehe Beschreibung im Quelltext:

```
#####  
'Schrittmotoransteuerung mit RN-Control und  
'Ansteuerungsboard RN-MOTOR über I2C  
'  
'  
'Aufgabe:  
' Dieses Testprogramm beschreibt die Motorboard I2C-Funktionen  
' und legt einige der Befehle auf die 5 Tasten  
  
' Den verschiedenen Tasten sind bestimmte Funktionen zugeordnet  
' Taste 1: Stellt für den linken Motor den Motorstrom auf 200mA ein und schaltet Motor danach  
ein  
' Taste 2: Lässt den linken Motor endlos drehen  
' Taste 3: Fragt der bereits ausgeführten Motorschritte des linken Motors vom Board ab und  
' berechnet daraus Umdrehungen und Fahrtstrecke  
' Daten werden über RS232 angezeigt  
' Taste 4: Ändert die Geschwindigkeit des linken Motors mit jedem Tastendruck  
' Taste 5: Schaltet den linken Motor wieder aus  
  
' Achtung:  
' In der Bascom Version 1.11.7.4 ist noch ein kleienr Bug in der I2C  
' Library. Dadurch funktioniert eventuell der I2CSEND Befehl nicht korrekt  
' Zum korrekten Ablauf des Demos sollte daher die aktuelle Library  
' in Bascom installiert werden. Kann man hier Downloaden  
' Infos dazu im Roboternetz-Thread  
' http://www.roboternetz.de/phpBB2/viewtopic.php?t=694&start=22  
' http://www.roboternetz.de/download/bascomi2clib.zip  
'  
'Autor: Frank Brall  
'Weitere Beispiele und Beschreibung der Hardware  
'in der Anleitung zu RN-Motor  
'Anleitung findet man unter http://www.Roboternetz.de im Download Bereich  
'oder www.robotikhardware.de  
'Weitere Beispiele sind im Roboternetz gerne willkommen!  
'#####  
  
Declare Function Tastenabfrage() As Byte  
Declare Sub Aendereslaveid()
```

```

$regfile = "m16def.dat"

'Befehle im Schrittmotorbetrieb
'K ist eine Kennungsnummer die bei RN-Motor immer 10 ist, danach folgt Befehlscode und
notwendige Parameter
'Es müssen immer 5 Bytes übertragen werden, auch wenn weniger Parameter ausreichen

'k 1 x y      Motor maximaler Strom x=0 Linker Motor  x=1 Rechte Motor x2= Beide Motoren Y =
(Milliampere / 10)
'
'k 2 y      Beispiel : Y = 150 Bedeutet 1,5A Oder Y = 20 Bedeutet 200ma
Referenzspannung des Controllers (normal 2.5 und muss nicht geändert werden)
'
'k 3 x      Beispiel: y=241 bedeutet 2,41 V
Motor stopp (Haltespannung bleibt ein) x=0 Linker Motor  x=1 Rechte Motor  x2=
Beide Motoren Stopp
'k 4 x y      Motor Drehrichtung x=0 Linker Motor  x=1 Rechte Motor x2= Beide Motoren y=0=Links
rum y=1 rechts rum
'k 5 x y1 yh Motor genau y Schritte bewegen x=0 Linker Motor  x=1 Rechte Motor  x2= Beide
Motoren
'y wird in High und Low Byte übertragen
'k 6 x      Motor endlose Schritte bewegen bis Stop Befehl kommt x=0 Linker Motor  x=1 Rechte
Motor  x2= Beide Motoren

'k 7 x      Motor einen einzigen Schritt bewegen x=0 Linker Motor  x=1 Rechte Motor  x2=
Beide Motoren
'k 8 x y      Motor Schrittgeschwindigkeit x=0 Linker Motor  x=1 Rechte Motor x2= Beide Motoren
Y = 1 bis 255
'k 9 x      Motor ganz ausschalten  x=0 Linker Motor  x=1 Rechte Motor  x2= Alle Motoren
Stopp

'k 10 x      Motor ein und init  x=0 Linker Motor  x=1 Rechte Motor

'k 11      Konfiguration und Testwerte werden über RS232 ausgegeben

'k 12      Sleep Mode

'k 13 x      Motor selektieren

'k 14 x      x=0 Vollschrittmodus  x=1 Halbschrittmodus
'k 15 99 99 x Slave ID zwischen Hex 50 und Hex 70 ändern (erst ab RN-Motor Firmware 1.1
möglich)

Const Befehl_maxstrom = 1
Const Befehl_referenzspannung = 2
Const Befehl_stop = 3
Const Befehl_drehrichtung = 4
Const Befehl_schrittanzahl_drehen = 5
Const Befehl_endlos_drehen = 6
Const Befehl_einziges_schritt = 7
Const Befehl_geschwindigkeit = 8
Const Befehl_ausschalten = 9
Const Befehl_einschalten = 10
Const Befehl_konfiguration_zeigen = 11
Const Befehl_sleep_modus = 12
Const Befehl_motor_selektieren = 13
Const Befehl_schrittmotormodus = 14

Const Motor_links = 0
Const Motor_rechts = 1

Const I2crnmotorslaveadr = &H56          'I2C SlaveAdresse von RN-Motor
Erweiterung
Const I2crnmotorsreadlaveadr = &H58      'I2C SlaveAdresse von RN-Motor
Erweiterung

Dim I2cdaten(6) As Byte                   'Array um Befehlsfolge auszunehmen
Dim Lowbyte As Byte
Dim Highbyte As Byte
Dim Schritte As Long
Dim Temp As Byte
Dim Ltemp As Long
Dim Umdrehungen As Single
Dim Fahrstrecke As Word

Dim Geschwindigkeit As Byte

Dim I As Integer
Dim N As Integer
Dim Ton As Integer

$crystal = 16000000                       'Quarzfrequenz
$baud = 9600

```

```

Config Scl = Portc.0           'Ports fuer IIC-Bus
Config Sda = Portc.1

Geschwindigkeit = 250
I2cinit

Config Adc = Single , Prescaler = Auto           'Für Tastenabfrage und
Spannungsmessung
Config Pina.7 = Input           'Für Tastenabfrage
Porta.7 = 1                     'Pullup Widerstand ein
Dim Taste As Byte

I = 0
Sound Portd.7 , 400 , 450       'BEEP
Sound Portd.7 , 400 , 250       'BEEP
Sound Portd.7 , 400 , 450       'BEEP
Print
Print "**** RN-CONTROL V1.4 ****"
Print "Demoprogramm um Zusatzkarte RN-Motor im Schrittmotormodus zu testen"
Print
Do
  Taste = Tastenabfrage()
  If Taste <> 0 Then

    Select Case Taste
      Case 1
        'Motorstrom auf 200mA begrenzen
        I2cdaten(1) = 10           'Kennung muss bei RN-Motor immer
10 sein
        I2cdaten(2) = Befehl_maxstrom           'Befehlscode
        I2cdaten(3) = Motor_links           '1 Parameter
        I2cdaten(4) = 200 / 10           '2 Parameter (hier Milliampere /10
in diesem Fall wird 200mA eingestellt)
        I2csend I2crnmotorslaveadr , I2cdaten(1) , 5           'Befehl wird gesendet (es müssen
immer 5 Bytes gesendet werden, auch wenn weniger Parameter notwendig sind)
        I2cstop

        'Halbschrittmodus
        I2cdaten(1) = 10           'Kennung muss bei RN-Motor immer
10 sein
        I2cdaten(2) = Befehl_schrittmotormodus           'Befehlscode
        I2cdaten(3) = 1           'Beide Motoren Halbschrittmodus
        I2csend I2crnmotorslaveadr , I2cdaten(1) , 5           'Befehl wird gesendet (es müssen
immer 5 Bytes gesendet werden, auch wenn weniger Parameter notwendig sind)

        'Motor einschalten
        I2cdaten(1) = 10           'Kennung muss bei RN-Motor immer
10 sein
        I2cdaten(2) = Befehl_einschalten           'Befehlscode
        I2cdaten(3) = Motor_links           '1 Parameter
        I2csend I2crnmotorslaveadr , I2cdaten(1) , 5           'Befehl wird gesendet (es müssen
immer 5 Bytes gesendet werden, auch wenn weniger Parameter notwendig sind)

      Case 2
        'Motor endlos drehen
        I2cdaten(1) = 10           'Kennung muss bei RN-Motor immer
10 sein
        I2cdaten(2) = Befehl_endlos_drehen           'Befehlscode
        I2cdaten(3) = Motor_links           '1 Parameter
        I2csend I2crnmotorslaveadr , I2cdaten(1) , 5           'Befehl wird gesendet (es müssen
immer 5 Bytes gesendet werden, auch wenn weniger Parameter notwendig sind)

      Case 3
        Aendereslaveid
        'Motor für Datenabruf anwählen
        I2cdaten(1) = 10           'Kennung muss bei RN-Motor immer
10 sein
        I2cdaten(2) = Befehl_motor_selektieren           'Befehlscode
        I2cdaten(3) = 0           '1 Parameter

```

```

        I2csend I2crnmotorslaveadr , I2cdaten(1) , 5      'Befehl wird gesendet (es müssen
immer 5 Bytes gesendet werden, auch wenn weniger Parameter notwendig sind)
        I2cstop

        'Schritte abrufen und Umdrehungen und Fahrtstrecke berechnen und über RS232
anzeigen
        I2cstart
        I2cwbyte I2crnmotorsreadlaveadr
        I2crbyte Temp , Ack
        Schritte = Temp
        I2crbyte Temp , Ack
        Ltemp = Temp * 256
        Schritte = Schritte + Ltemp
        I2crbyte Temp , Ack
        Ltemp = Temp * 65536
        Schritte = Schritte + Ltemp
        I2crbyte Temp , Nack
        Ltemp = Temp * 16777216
        Schritte = Schritte + Ltemp
        I2cstop
        Print "Schrittzahl:" ; Schritte
        Umdrehungen = Schritte / 200                                'Wenn Schrittwinkel 1,8 Grad
beträgt dann durch 200 teilen
        Print "Umdrehungen: " ; Umdrehungen
        'Umfang des rades mal Umdrehungen ergibt Fahrtstrecke
        'Umfang berechnet sich auch U=3.14 * Durchmesser
        'In dem Beispiel nehmen wir ein 10 cm Rad am Roboter an
        Fahrtstrecke = Umdrehungen * 31.4
        Print "Der Roboter ist " ; Fahrtstrecke ; " cm gefahren "

        Case 4
        'Geschwindigkeit ändern
        If Geschwindigkeit > 40 Then
            Geschwindigkeit = Geschwindigkeit - 20
        Else
            Geschwindigkeit = Geschwindigkeit - 1
        End If
        Print "Geschwindigkeitsstufe: " ; Geschwindigkeit
        I2cdaten(1) = 10                                            'Kennung muss bei RN-Motor immer
10 sein
        I2cdaten(2) = Befehl_geschwindigkeit                       'Befehlscode
        I2cdaten(3) = Motor_links                                  '1 Parameter
        I2cdaten(4) = Geschwindigkeit                             '2 Parameter (hier Milliampere /10
in diesem Fall wird 200mA eingestellt)
        I2csend I2crnmotorslaveadr , I2cdaten(1) , 5      'Befehl wird gesendet (es müssen
immer 5 Bytes gesendet werden, auch wenn weniger Parameter notwendig sind)
        I2cstop

        Case 5
        'Motor ausschalten
        I2cdaten(1) = 10                                            'Kennung muss bei RN-Motor immer
10 sein
        I2cdaten(2) = Befehl_ausschalten                           'Befehlscode
        I2cdaten(3) = Motor_links                                  '1 Parameter
        I2csend I2crnmotorslaveadr , I2cdaten(1) , 5      'Befehl wird gesendet (es müssen
immer 5 Bytes gesendet werden, auch wenn weniger Parameter notwendig sind)
        I2cstop

        End Select
        Sound Portd.7 , 400 , 500                                'BEEP
    End If

    Waitms 100
Loop
End

' Diese Unterfunktion fragt die Tastatur am analogen Port ab
' Sollte beim betätigen einer Taste kein Quittungston kommen, dann
' muss die die Tastenabfrage (Select Case Anweisung in Funktion )
' an ihr Board angepasst werden. Widerstandstoleranzen sorgen in
' Einzelfällen manchmal dafür das die Werte etwas anders ausfallen
' Am besten dann den WS wert mit Print für jede Taste ausgegeben lassen

Function Tastenabfrage() As Byte
Local Ws As Word

Tastenabfrage = 0

```

```

Ton = 600
Start Adc
Ws = Getadc(7)
Print "ws= " ; Ws
If Ws < 1010 Then
  Select Case Ws
    Case 410 To 455
      Tastenabfrage = 1
      Ton = 550
    Case 340 To 380
      Tastenabfrage = 2
      Ton = 500
    Case 250 To 305
      Tastenabfrage = 3
      Ton = 450
    Case 180 To 220
      Tastenabfrage = 4
      Ton = 400
    Case 100 To 130
      Tastenabfrage = 5
      Ton = 350
  End Select
  Sound Portd.7 , 400 , Ton
End If

End Function

'Diese Funktion kann genutzt werden um die SlaveID von RN-Motor
'zu aendern. Dies ist allerdings erst in der neusten Firmware
'von RN-Motor möglich. Bei älteren Versionen müsste Controller gewechselt
'werden. Die Aktuelle Version und Slave ID wird über
'RS232 von RN-Motor bei RESET ausgegeben
Sub Aendereslaveid()
  I2cdaten(1) = 10
  I2cdaten(2) = 15
  I2cdaten(3) = 99
  I2cdaten(4) = 99
  I2cdaten(5) = &H58
  I2csend I2crnmotorslaveadr , I2cdaten(1) , 5
End sub

```

Taste 1	Linken Motor einschalten
Taste 2	Linken Motor endlos
Taste 3	Schrittzahl abrufen und Fahrtstrecke berechnen und über RS232 ausgeben
Taste 4	Geschwindigkeit ändern
Taste 5	Linken Motor ausschalten

Damit haben Sie den ersten Einstieg erfolgreich abgeschlossen.

Wenn Sie das Demoprogram gründlich studieren werden Sie viele Sachen davon ableiten und in eigenen Programmen verwenden können.

Beachten Sie das die derzeitige I2C-Firmware von dem sogenannten Clock-Stretching (ein Begriff aus dem I2C-Standard) Gebrauch macht. Bei modernen Controllern und Programmiersprachen wie z.B. Bascom brauchen sie sich damit nicht näher zu befassen. Es gibt jedoch in anderen Programmiersprachen oder bei anderen Controllern I2C Routinen, die diesen Mode nicht berücksichtigen. In dem Fall sollten Sie entweder eigene I2C-Routinen verwenden, oder aber nach aktuellen Librarys Ausschau halten. Zum Stichwort Clock Stretching finden Sie hier Infos: <http://www.roboternetz.de/wiki/pmwiki.php?n=Main.ClockStretching>

Hier noch ein Pascal-Beispiel :

```

{
Pascal Beispiel von Roboternetz.de Mitglied Nickname Johannes
Autor-Webseite: http://algorithms.mindrobots.de
Da ich zwei Boards ansteuere, habe ich das ganze so programmiert,
dass ich als Nummer für die Motoren eine Zahl von 1 bis 4 übergebe.
Die Funktionen pStepperAdress() und pStepperMotor() ermitteln daraus
die Adresse und die Motornummer, die gesendet werden soll.
Das soll nicht irritieren.
Bei der StepperSteps()-Funktion muss man daran denken, dass die Funktion nicht so lange
wartet, bis die Schritte ausgeführt wurden.
}

program ApplicationBoardTest;

```

```

{ $BOOTRST $00C00}          {Reset Jump to $00C00}
{$NOSHADOW}
{ $W+ Warnings}           {Warnings off}

Device = mega8, VCC = 5;
Import SysTick, SerPort, PWMport1, PWMport2, ADCPort, I2Cexpand, TWImaster, LCDmultiPort;

From System Import;

Define
  ProcClock      = 16000000;      {Hertz}
  SysTick        = 10;           {msec}
  StackSize      = $003C, iData;
  FrameSize      = $003C, iData;
  SerPort        = 19200, Stop1;  {Baud, StopBits|Parity}
  RxBuffer       = 8, iData;
  TxBuffer       = 8, iData;
  ADCchans       = 2, iData;
  ADCpresc       = 128;
  PWMres         = 9;            {bits}
  PWMpresc       = 64;
  TWIpresc       = TWI_BR400;
  LCDmultiPort   = I2C_TWI;
  LCDrows_M      = 2;           {rows}
  LCDcolumns_M   = 16;          {columns per line}
  LCDtype_M      = 44780;
  I2Cexpand      = I2C_TWI, $38; {Port0 = PCA9554 an Adresse $38}
  I2CexpPorts    = Port0; // I2Cexpand ports
                  {Port0 ist die IO Port Erweiterung (I2Cexpand) über den PCA9554 Baustein.
                   Durch Anschluß weiterer PCA9554 kann man bis Port7 erweitern}

Implementation

{$IDATA}

{-----}
{ Type Declarations }

type

{-----}
{ Const Declarations }
const
  STEPPER_ADRESS1:byte = 43;
  STEPPER_ADRESS2:byte = 44;
  STEPPER_MOTOR1:byte = 0;
  STEPPER_MOTOR2:byte = 1;
  STEPPER_MOTOR3:byte = 2;
  STEPPER_MOTOR4:byte = 3;
  STEPPER_DIRECTION_RIGHT:byte = 0;
  STEPPER_DIRECTION_LEFT:byte = 1;
  STEPPER_MODUS_VS:byte = 0;
  STEPPER_MODUS_HS:byte = 1;

{-----}
{ Var Declarations }
{$IDATA}

var
{Portdefinition für Taster und LED's. Kein Unterschied zwischen normalen IO-Ports und
I2Cexpand Ports}
  LED1[@Port0, 4]      : bit;
  LED2[@Port0, 5]      : bit;
  LED3[@Port0, 6]      : bit;
  LED4[@Port0, 7]      : bit;
  Taster1[@Pin0, 3]    : bit;
  Taster2[@Pin0, 2]    : bit;
  Taster3[@Pin0, 1]    : bit;
  Taster4[@Pin0, 0]    : bit;
  M1F[@PortD, 6]       : bit;
  M1R[@PortD, 7]       : bit;
  M2F[@PortD, 5]       : bit;
  M2R[@PortD, 4]       : bit;
  temp:word;

{-----}
{ functions }

procedure InitPorts;
begin
  PortD:= %00001100;

```

```

    DDRD:= %11110000;
    PWMPort1:= 0;
    PWMPort2:= 0;
end InitPorts;

procedure Init_I2C;
begin
    LCDsetup_M(LCD_m1);
    //Initialisiere I2C LCD an Adresse $20
    LCDcursor_M(LCD_m1, false, false); //Setze Cursor
    Blink off und Cursor visible off
    DDR0:= $F0; //Port0.4 bis
    7 als Ausgänge (LED's)
    Port0:= $F0; //Port0.4 bis
    7 auf 1 (LED's aus)
    INP_POL0:= $00; //Polarität
    für Taster positiv
end Init_I2C;

function pStepperAddress( Motor:byte ) : byte;
begin
    if (Motor = 0) or (Motor = 1) then return(STEPPER_ADRESS1); endif;
    if (Motor = 2) or (Motor = 3) then return(STEPPER_ADRESS2); endif;
end pStepperAddress;

function pStepperMotor( Motor:byte ) : byte;
begin
    if (Motor = 2) then return(0);
    elsif (Motor = 3) then return(1);
    else return(Motor); endif;
end pStepperMotor;

procedure StepperCurrency( Motor : byte; Currency : integer );
var data : array[1..4] of byte;
begin
    data[1] := 1;
    data[2] := pStepperMotor( Motor );
    data[3] := byte( Currency DIV 10 );
    data[4] := 0;
    mDelay( 100 );
    TWIout( pStepperAddress( Motor ), 10, data );
    mDelay( 100 );
end;

procedure StepperModus( Motor, Modus : byte );
var data : array[1..4] of byte;
begin
    data[1] := 14;
    data[2] := Modus;
    data[3] := 0;
    data[4] := 0;
    mDelay( 100 );
    TWIout( pStepperAddress( Motor ), 10, data );
    mDelay( 100 );
end;

procedure StepperOn( Motor : byte );
var data : array[1..4] of byte;
begin
    data[1] := 10;
    data[2] := pStepperMotor( Motor );
    data[3] := 0;
    data[4] := 0;
    mDelay( 200 );
    TWIout( pStepperAddress( Motor ), 10, data );
    mDelay( 200 );
end;

procedure StepperStop( Motor : byte );
var data : array[1..4] of byte;
begin
    data[1] := 3;
    data[2] := pStepperMotor( Motor );
    data[3] := 0;
    data[4] := 0;
    mDelay( 200 );
    TWIout( pStepperAddress( Motor ), 10, data );
    mDelay( 200 );
end;

procedure StepperOff( Motor : byte );
var data : array[1..4] of byte;
begin

```



```

data[1] := 9;
data[2] := pStepperMotor( Motor );
data[3] := 0;
data[4] := 0;
mDelay( 10 );
TWIout( pStepperAdress( Motor ), 10, data );
mDelay( 10 );
end;

procedure StepperRotate( Motor, Speed, Direction : byte );
var data : array[1..4] of byte;
begin
data[1] := 4;
data[2] := pStepperMotor( Motor );
data[3] := Direction;
data[4] := 0;
TWIout( pStepperAdress( Motor ), 10, data );
mDelay( 100 );
data[1] := 8;
data[2] := pStepperMotor( Motor );
data[3] := Speed;
data[4] := 0;
TWIout( pStepperAdress( Motor ), 10, data );
mDelay( 100 );
data[1] := 6;
data[2] := pStepperMotor( Motor );
data[3] := 0;
data[4] := 0;
TWIout( pStepperAdress( Motor ), 10, data );
mDelay( 100 );
end;

procedure StepperStep( Motor, Direction : byte );
var data : array[1..4] of byte;
begin
data[1] := 4;
data[2] := pStepperMotor( Motor );
data[3] := Direction;
data[4] := 0;
TWIout( pStepperAdress( Motor ), 10, data );
mDelay( 30 );
data[1] := 7;
data[2] := pStepperMotor( Motor );
data[3] := 0;
data[4] := 0;
TWIout( pStepperAdress( Motor ), 10, data );
end;

procedure StepperSteps( Motor, Speed, Direction : byte; Steps : integer );
var data : array[1..4] of byte;
begin
data[1] := 4;
data[2] := pStepperMotor( Motor );
data[3] := Direction;
data[4] := 0;
TWIout( pStepperAdress( Motor ), 10, data );
mDelay( 30 );
data[1] := 8;
data[2] := pStepperMotor( Motor );
data[3] := Speed;
data[4] := 0;
TWIout( pStepperAdress( Motor ), 10, data );
mDelay( 100 );
data[1] := 5;
data[2] := pStepperMotor( Motor );
data[3] := LO( Steps );
data[4] := HI( Steps );
TWIout( pStepperAdress( Motor ), 10, data );
end;

{-----}
{ Main Program }
{$IDATA}

begin
InitPorts;
Init_I2C;
EnableInts;
LCDclr_m(LCD_m1);
M1F:= true;
M1R:= false;
M2F:= true;

```

```

    M2R:= false;
    PWMport1:= 400;
    PWMport2:= 400;
    temp:=GetADC(1);
write(LCDout_m, '   mindrobots   ');
LCDxy_m(LCD_m1, 0, 1);

//Einstellen des Motorstroms
StepperCurrency( STEPPER_MOTOR1, 200 );
//Vollschritt oder Halbschritt
StepperModus( STEPPER_MOTOR1, STEPPER_MODUS_VS );
//Einschalten
StepperOn( STEPPER_MOTOR1 );
//15 Schritte mit langsamer Geschwindigkeit, links herum
StepperSteps( STEPPER_MOTOR1, 200, STEPPER_DIRECTION_LEFT, 15 );
mDelay( 5000 );
//15 Schritte mit höherer Geschwindigkeit, links herum
StepperSteps( STEPPER_MOTOR1, 50, STEPPER_DIRECTION_LEFT, 15 );
mDelay( 1000 );

end ApplicationBoardTest.

```

Und hier noch ein Beispiel wie die C-Control RN-Motor per RS232 ansteuert:

```

'DEMO das aufzeigt
'wie einfach RN-Motor über RS232
'auch mit einer C-Control angesteuert werden kann

'Motorstrom festlegen

Print "#rmi" ; : Put 2 : Put 58
Pause 20

'beide Motoren ein
Print "#rmo" ; : Put 2
Pause 20

'beide motoren rechts
Print "#rmd" ; : Put 2 : Put 0

Pause 20
'geschwindigkeit
Print "#rmg" ; : Put 2 : Put 25
Pause 20

'endlos drehen
Print "#rme" ; : Put 2

#hinundher
pause 1000

'Linker motor nach links
Print "#rmd" ; : Put 0 : Put 0

'rechter motor nach rechts
Print "#rmd" ; : Put 1 : Put 1

pause 1000

'Linker motor nach rechts
Print "#rmd" ; : Put 0 : Put 1

'rechter motor nach links
'Print "#rmd" ; : Put 1 : Put 0

Goto hinundher

```

Noch ein Beispiel das zeigt wie man mit einer C-Control die I2C-Version ansteuert.

```
'*****'
'* Demo bereitgestellt von WINDT SYSTEMS 2005 / H.J. WINDT
'* für RN-MOTOR ST I2C (I2C Schrittmotoransteuerung)
'* EXAMPLE VIA EMULATED I2C BUS v1.1 for CCIUM2.02 and CCIMainU1.1*'
'* Zur besseren Übersicht gering modifiziert und übersetzt bei Frank
'******'

'***** I/O PORTS *****'
define sda port[4]
define scl port[5]

'***** VARIABLES *****'
define i2c_nack bit[1]
define i2c_last bit[2]
define i2c_nack_count byte[2]
define i2c_byte byte[3]
define i2c_out_bit bit[24]
define i2c_in_bit bit[17]

define loop byte[4]

define motor_selection byte[5]
define control byte[6]
define motor_direction byte[7]
define motor_speed byte[8]
define motor_continuous byte[9]

define motor_steps word[6]

define low_word word[7]
define high_word word[8]

'***** SETUP *****'
print"#ON_CONFIG#"; : put &b1000 : print"#OFF#"; '<--TAKE THIS LINE AWAY FOR C-CONTROL MAIN
UNIT V1.1!!'
gosub initialize_rn_motor

'***** PROGRAMM *****'
#start
print"Wähle Motor/ 0 = links/ 1 = rechts/ 2 = beide"
input motor_selection
print"Enter control/ 0 = Stoppe Motor(en)/ 1 = Parameter Motor(en)/ 2 = Gefahrene Schritte
anzeigen"
input control
on control goto stop_motor, control_motor, read_motor_steps_taken

#stop_motor
gosub start_i2c
i2c_byte = &h56 : gosub write_byte_i2c
i2c_byte = 10 : gosub write_byte_i2c
i2c_byte = 3 : gosub write_byte_i2c
i2c_byte = motor_selection : gosub write_byte_i2c
i2c_byte = 255 : gosub write_byte_i2c
i2c_byte = 255 : gosub write_byte_i2c
gosub stop_i2c

if i2c_nack_count > 18 then goto no_ack_from_i2c_device
if i2c_nack then goto stop_motor
goto start

#control_motor
print"Wähle Drehrichtung/ 0 = links/ 1 = rechts"
input motor_direction
print"Wähle Motorgeschwindigkeit (0 to 255) 255 = sehr langsam"
input motor_speed
print"Motoren endlos drehen?/ 0 = Nein/ 1 = Ja"
input motor_continuous
if motor_continuous = 1 then gosub rn_motor_control
if motor_continuous = 1 then goto start
print"Wieviele Schritte (0 to 32767)?"
input motor_steps
gosub rn_motor_control
goto start

#read_motor_steps_taken
if motor_selection > 1 then print "Ich kann nicht beide Schrittzahlen gleichzeitig lesen!!"
if motor_selection > 1 then goto start
gosub start_i2c
```

```

i2c_byte = &h56 : gsub write_byte_i2c
i2c_byte = 10 : gsub write_byte_i2c
i2c_byte = 13 : gsub write_byte_i2c
i2c_byte = motor_selection : gsub write_byte_i2c
gsub stop_i2c
if i2c_nack then goto pass_read
gsub start_i2c
i2c_byte = &h57 : gsub write_byte_i2c
gsub read_byte_i2c : low_word = i2c_byte
gsub read_byte_i2c : low_word = i2c_byte * 256 + low_word
gsub read_byte_i2c : high_word = i2c_byte
gsub read_last_byte_i2c : high_word = i2c_byte * 256 + high_word
gsub stop_i2c

#pass_read
if i2c_nack_count > 18 then goto no_ack_from_i2c_device
if i2c_nack then goto read_motor_steps_taken
print"HIGH word = ";high_word;" Low word = ";low_word
goto start

end

'***** Unterrountinen *****'
#rn_motor_control
#set_motor_direction
gsub start_i2c
i2c_byte = &h56 : gsub write_byte_i2c
i2c_byte = 10 : gsub write_byte_i2c
i2c_byte = 4 : gsub write_byte_i2c
i2c_byte = motor_selection : gsub write_byte_i2c
i2c_byte = motor_direction : gsub write_byte_i2c
i2c_byte = 255 : gsub write_byte_i2c
gsub stop_i2c
if i2c_nack_count > 18 then goto no_ack_from_i2c_device
if i2c_nack then goto set_motor_direction

#set_motor_speed
gsub start_i2c
i2c_byte = &h56 : gsub write_byte_i2c
i2c_byte = 10 : gsub write_byte_i2c
i2c_byte = 8 : gsub write_byte_i2c
i2c_byte = motor_selection : gsub write_byte_i2c
i2c_byte = motor_speed : gsub write_byte_i2c
i2c_byte = 255 : gsub write_byte_i2c
gsub stop_i2c
if i2c_nack_count > 18 then goto no_ack_from_i2c_device
if i2c_nack then goto set_motor_speed

if motor_continuous = 1 then goto turn_motor_continuous

#turn_motor_number_of_steps
gsub start_i2c
i2c_byte = &h56 : gsub write_byte_i2c
i2c_byte = 10 : gsub write_byte_i2c
i2c_byte = 5 : gsub write_byte_i2c
i2c_byte = motor_selection : gsub write_byte_i2c
i2c_byte = motor_steps mod 256 : gsub write_byte_i2c
i2c_byte = motor_steps / 256 : gsub write_byte_i2c
gsub stop_i2c
if i2c_nack_count > 18 then goto no_ack_from_i2c_device
if i2c_nack then goto turn_motor_number_of_steps
return

#turn_motor_continuous
gsub start_i2c
i2c_byte = &h56 : gsub write_byte_i2c
i2c_byte = 10 : gsub write_byte_i2c
i2c_byte = 6 : gsub write_byte_i2c
i2c_byte = motor_selection : gsub write_byte_i2c
i2c_byte = 255 : gsub write_byte_i2c
i2c_byte = 255 : gsub write_byte_i2c
gsub stop_i2c
if i2c_nack_count > 18 then goto no_ack_from_i2c_device
if i2c_nack then goto turn_motor_continuous
return

#start_i2c
sda = 0 : scl = 0
return

#stop_i2c

```

```

sda = 0 : deact scl : deact sda
return

#write_byte_i2c
for loop = 1 to 8
sda = i2c_out_bit
deact scl
#write_byte_i2c_clock_stretch
if not scl then goto write_byte_i2c_clock_stretch
scl = 0
i2c_byte = i2c_byte shl 1
next
deact sda
deact scl

#write_byte_i2c_ack_clock_stretch
if not scl then goto write_byte_i2c_ack_clock_stretch
i2c_nack = sda
scl = 0
if i2c_nack then goto i2c_error
i2c_nack_count = 0
return

#i2c_error
i2c_nack_count = i2c_nack_count + 1
goto stop_i2c

#read_byte_i2c
i2c_last = 0
#get_i2c_byte
deact sda
for loop = 1 to 8
deact scl
#read_byte_i2c_clock_stretch
if not scl then goto read_byte_i2c_clock_stretch
i2c_in_bit = sda
scl = 0
if loop < 8 then i2c_byte = i2c_byte shl 1
next
if i2c_last then deact sda else sda = 0
deact scl
#read_byte_i2c_ack_clock_stretch
if not scl then goto read_byte_i2c_ack_clock_stretch
scl = 0
return

#read_last_byte_i2c
i2c_last = 1
goto get_i2c_byte

'***** INITIALISIERE UNTERROUTINEN *****'
#initialize_rn_motor
#set_motor_amps
gosub start_i2c
i2c_byte = &h56 : gosub write_byte_i2c
i2c_byte = 10 : gosub write_byte_i2c
i2c_byte = 1 : gosub write_byte_i2c
i2c_byte = 2 : gosub write_byte_i2c
i2c_byte = 190 : gosub write_byte_i2c
i2c_byte = 255 : gosub write_byte_i2c
gosub stop_i2c
if i2c_nack_count > 18 then goto no_ack_from_i2c_device
if i2c_nack then goto set_motor_amps

#set_mode
gosub start_i2c
i2c_byte = &h56 : gosub write_byte_i2c
i2c_byte = 10 : gosub write_byte_i2c
i2c_byte = 14 : gosub write_byte_i2c
i2c_byte = 0 : gosub write_byte_i2c
i2c_byte = 255 : gosub write_byte_i2c
i2c_byte = 255 : gosub write_byte_i2c
gosub stop_i2c
if i2c_nack_count > 18 then goto no_ack_from_i2c_device
if i2c_nack then goto set_mode

#switch_motors_on
gosub start_i2c
i2c_byte = &h56 : gosub write_byte_i2c
i2c_byte = 10 : gosub write_byte_i2c
i2c_byte = 10 : gosub write_byte_i2c
i2c_byte = 2 : gosub write_byte_i2c

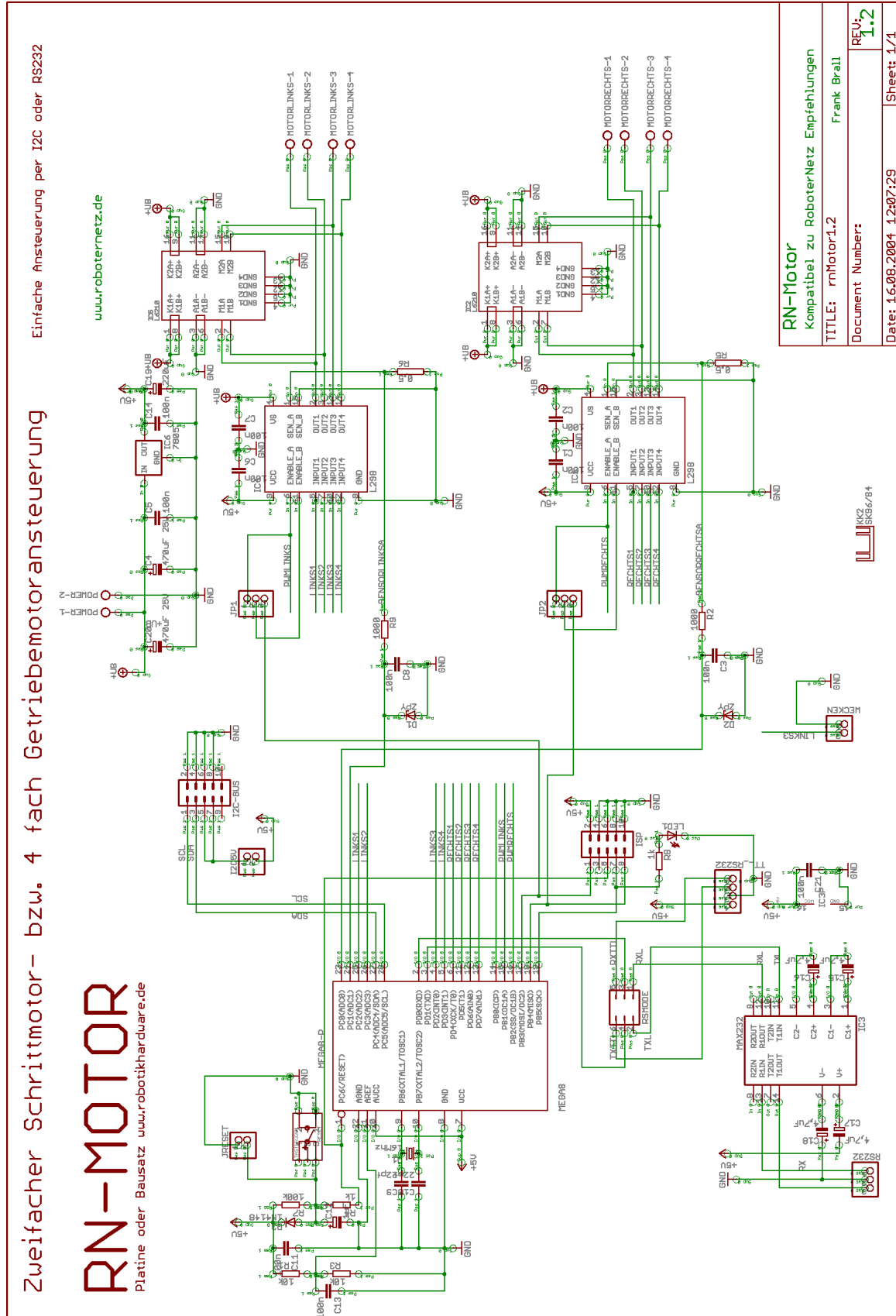
```

```
i2c_byte = 255 : gosub write_byte_i2c
i2c_byte = 255 : gosub write_byte_i2c
gosub stop_i2c
if i2c_nack_count > 18 then goto no_ack_from_i2c_device
if i2c_nack then goto switch_motors_on

#show_configuration
gosub start_i2c
i2c_byte = &h56 : gosub write_byte_i2c
i2c_byte = 10 : gosub write_byte_i2c
i2c_byte = 11 : gosub write_byte_i2c
i2c_byte = 255 : gosub write_byte_i2c
i2c_byte = 255 : gosub write_byte_i2c
i2c_byte = 255 : gosub write_byte_i2c
gosub stop_i2c
if i2c_nack_count > 18 then goto no_ack_from_i2c_device
if i2c_nack then goto show_configuration
return

'***** ERROR MESSAGES *****'
#no_ack_from_i2c_device
print"*****"
print"* No ACK from I2C device *"
print"*****"
pause 50
goto no_ack_from_i2c_device
'*****'
```

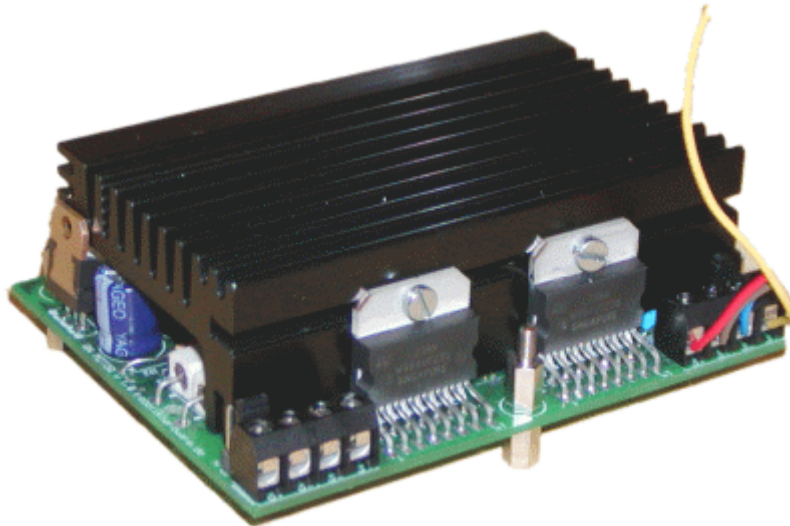
Schaltplan



Achtung, dieser Schaltplan gilt für die Version 1.2 des Boards. Es werden auch einige Fertigboards mit etwas anderer Platine ausgeliefert, dort weicht die Bestückung geringfügig ab. Die Funktion der Boards ist jedoch identisch.

Sollte in dieser Doku noch der ein oder andere Fehler drin stecken, so bitte ich um Nachsicht und Hinweise per Mail an den Entwickler frank@roboternetz.de Also immer mal im Download Bereich nach der Versionsnummer der Doku schauen, Ergänzungen sind denkbar!

**Der Nachbau dieses Boards ist ausdrücklich gestattet,
jedoch nur für den privaten Einsatz!**
Die Kommerzielle bzw. Gewerbliche Verwertungen bedürfen der schriftlichen
Einwilligung des Entwicklers www.robotikhardware.de



**Online-Bestellung von Platinen oder Erweiterungen über
<http://www.robotikhardware.de>**

Haftung, EMV-Konformität

Alle Teile der Schaltung wurden sorgfältigst geprüft und getestet. Trotzdem kann ich natürlich keine Garantie dafür übernehmen, daß alles einwandfrei funktioniert. Insbesondere übernehme ich keine Haftung für Schäden, die durch Nachbau, Inbetriebnahme etc. der hier vorgestellten Schaltungen entstehen. Derjenige, der den Bausatz zusammenbaut, gilt als Hersteller und ist damit selbst für die Einhaltung der geltenden Sicherheits- und EMV-Vorschriften verantwortlich.

Wenn nicht anders angegeben handelt es sich generell bei allen Bausätzen, Modulen und Boards um "nicht CE-geprüfte" Komponenten und sind konzipiert für den Einbau in Geräte oder Gehäuse. Bei der Anwendung müssen die CE-Normen eingehalten werden. Hierfür ist der Käufer verantwortlich.

Für Schäden die durch fehlerhaften Aufbau entstanden sind, direkt oder indirekt, ist die Haftung generell ausgeschlossen. Schadensersatzansprüche, gleich aus welchem Rechtsgrund, sind ausgeschlossen, soweit nicht vorsätzliches oder grob fahrlässiges Handeln vorliegt. Sofern wir haften, umfaßt unsere Haftung nicht solche Schäden, die nicht typischerweise erwartet werden konnten. Haftung und Schadensersatzansprüche sind auf den Auftragswert / Bauteilwert beschränkt. Bei der Lieferung von Fremdprodukten als auch Software gelten über diese Bedingungen hinaus die besonderen Lizenz- oder sonstigen Bedingungen des Herstellers.

Sicherheitshinweise

Beim Umgang mit Produkten, die mit elektrischer Spannung in Berührung kommen, müssen die gültigen VDE-Vorschriften beachtet werden, insbesondere VDE 0100, VDE 0550/0551, VDE 0700, VDE 0711 und VDE 0860. Vor Öffnen eines Gerätes stets den Netzstecker ziehen oder sicherstellen, daß das Gerät stromlos ist. Bauteile, Baugruppen oder Geräte dürfen nur in Betrieb genommen werden, wenn sie vorher berührungssicher in ein Gehäuse eingebaut wurden. Während des Einbaus müssen sie stromlos sein. Werkzeuge dürfen an Geräten, Bauteilen oder Baugruppen nur benutzt werden, wenn sichergestellt ist, daß die Geräte von der Versorgungsspannung getrennt sind und elektrische Ladungen, die in den im Gerät befindlichen Bauteilen gespeichert sind, vorher entladen wurden.

Spannungsführende Kabel oder Leitungen, mit denen das Gerät, das Bauteil oder die Baugruppe verbunden ist, müssen stets auf Isolationsfehler oder Bruchstellen untersucht werden. Bei Feststellen eines Fehlers in der Zuleitung muß das Gerät unverzüglich aus dem Betrieb genommen werden, bis die defekte Leitung ausgewechselt worden ist. Bei Einsatz von Bauelementen oder Baugruppen muß stets auf die strikte Einhaltung der in der zugehörigen Beschreibung genannten Kenndaten für elektrische Größen hingewiesen werden. Wenn aus einer vorliegenden Beschreibung für den nichtgewerblichen Endverbraucher nicht eindeutig hervorgeht, welche elektrischen Kennwerte für ein Bauteil oder eine Baugruppe gelten, wie eine externe Beschaltung durchzuführen ist oder welche externen Bauteile oder Zusatzgeräte angeschlossen werden dürfen und welche Anschlußwerte diese externen Komponenten haben dürfen, so muß stets ein Fachmann um Auskunft ersucht werden. • Es ist vor der Inbetriebnahme eines Gerätes generell zu prüfen, ob dieses Gerät oder Baugruppe grundsätzlich für den Anwendungsfall, für den es verwendet werden soll, geeignet ist!

Im Zweifelsfalle sind unbedingt Rückfragen bei Fachleuten, Sachverständigen oder den Herstellern der verwendeten Baugruppen notwendig! Bitte beachten Sie, daß Bedien- und Anschlußfehler außerhalb unseres Einflusses liegen. Verständlicherweise können wir für Schäden, die daraus entstehen, keinerlei Haftung übernehmen. Bausätze sollten bei Nichtfunktion mit einer genauen Fehlerbeschreibung (Angabe dessen, was nicht funktioniert...denn nur eine exakte Fehlerbeschreibung ermöglicht eine einwandfreie Reparatur!) und der zugehörigen Bauanleitung sowie ohne Gehäuse zurückgesandt werden. Zeitaufwendige Montagen oder Demontagen von Gehäusen müssen wir aus verständlichen Gründen zusätzlich berechnen. Bereits aufgebaute Bausätze sind vom Umtausch ausgeschlossen. Bei Installationen und beim Umgang mit Netzspannung sind unbedingt die VDE-Vorschriften zu beachten. Geräte, die an einer Spannung ≤ 35 V betrieben werden, dürfen nur vom Fachmann angeschlossen werden. In jedem Fall ist zu prüfen, ob der Bausatz für den jeweiligen Anwendungsfall und Einsatzort geeignet ist bzw. eingesetzt werden kann.

Die Inbetriebnahme darf grundsätzlich nur erfolgen, wenn die Schaltung absolut berührungssicher in ein Gehäuse eingebaut ist. Sind Messungen bei geöffnetem Gehäuse unumgänglich, so muß aus Sicherheitsgründen ein Trenntrafo zwischengeschaltet werden, oder, wie bereits erwähnt, die Spannung über ein geeignetes Netzteil, (das den Sicherheitsbestimmungen entspricht) zugeführt werden. Alle Verdrahtungsarbeiten dürfen nur im spannungslosen Zustand ausgeführt werden.

Derjenige, der einen Bausatz fertigt oder eine Baugruppe durch Erweiterung bzw. Gehäuseeinbau betriebsbereit macht, gilt nach DIN VDE 0869 als Hersteller und ist verpflichtet, bei der Weitergabe des Gerätes alle Begleitpapiere mitzuliefern und auch seinen Namen und Anschrift anzugeben. Geräte, die aus Bausätzen selbst zusammengestellt werden, sind sicherheitstechnisch wie ein industrielles Produkt zu betrachten.

Betriebsbedingungen

Der Betrieb der Baugruppe darf nur an der dafür vorgeschriebenen Spannung erfolgen.

Bei Geräten mit einer Betriebsspannung 35 Volt darf die Endmontage nur vom Fachmann unter Einhaltung der VDEBestimmungen vorgenommen werden.

Die Betriebslage des Gerätes ist beliebig.

Bei der Installation des Gerätes ist auf ausreichenden Kabelquerschnitt der Anschlußleitungen zu achten!

Die angeschlossenen Verbraucher sind entsprechend den VDEVorschriften mit dem Schutzleiter zu verbinden bzw. zu erden.

Die zulässige Umgebungstemperatur (Raumtemperatur) darf während des Betriebes 0°C und 40°C nicht unter-, bzw. überschreiten.

Das Gerät ist für den Gebrauch in trockenen und sauberen Räumen bestimmt.

Bei Bildung von Kondenswasser muß eine Akklimatisierungszeit von bis zu 2 Stunden abgewartet werden.

In gewerblichen Einrichtungen sind die Unfallverhütungsvorschriften des Verbandes der gewerblichen Berufsgenossenschaften für elektrische Anlagen und Betriebsmittel zu beachten.

In Schulen, Ausbildungseinrichtungen, Hobby- und Selbsthilfwerkstätten ist das Betreiben von Baugruppen durch geschultes Personal verantwortlich zu überwachen.

Betreiben Sie die Baugruppe nicht in einer Umgebung in welcher brennbare Gase, Dämpfe oder Stäube vorhanden sind oder vorhanden sein können.

Falls das Gerät einmal repariert werden muß, dürfen nur Original-Ersatzteile verwendet werden! Die Verwendung abweichender Ersatzteile kann zu ernsthaften Sach- und Personenschäden führen!

Dringt irgendeine Flüssigkeit in das Gerät ein, so könnte es dadurch beschädigt werden.

Das Board darf nur unter Aufsicht betrieben werden!

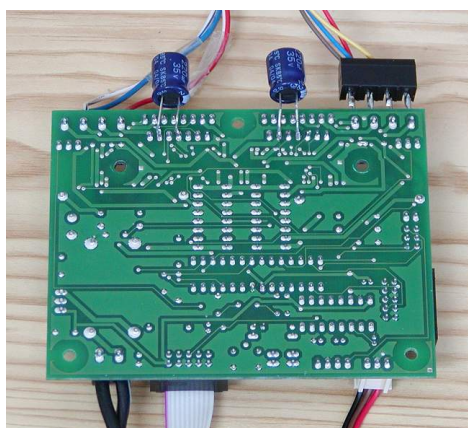
Wichtige Ergänzung für Revision 1.0 und 1.2 der RN-Motor-Platine bzw. des RN-Motor-Boards

Bitte diese Änderung nur durchführen wenn Sie diese Version besitzen. Am oberen Platinenrand können Sie die Revisionsnummer ersehen.

Es hat sich herausgestellt das es beim Motorboard unter Umständen an einigen Spannungsquellen zu Störungen kommen kann, wenn die Motoren höheren Strom benötigen (ab ca. 400 bis 500 mA). Dies kann unter Umständen zu einem RESET oder Absturz des Boards führen, die Motoren bleiben dann automatisch stehen.

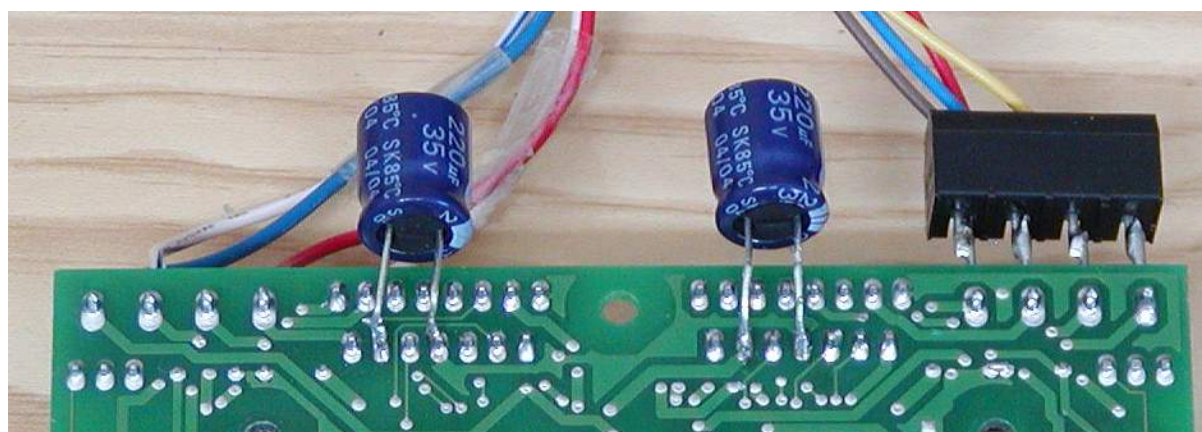
Dieser Fehler tritt nicht immer auf, sollten Sie jedoch Motoren mit mehr als ca. 300mA nutzen, wird dennoch folgende kleine Manipulation am Board empfohlen:

Liegen Sie das Board falsch herum auf den Tisch, so das die Motorklemmen oben sind.



Löten Sie zwei Elkos (220uF oder 470uF) wie auf dem Bild zu sehen an zwei bestimmte PIN's des Motortreibers (siehe Bilder). Achten Sie darauf das die Polarität stimmt, die Minus Kennzeichnung auf dem Elko muss rechts sein..

Hier die Vergrößerung:



Natürlich sollten Sie darauf achten das die Elkos nur die beiden Pins berühren an denen sie angelötet sind. Bei Bedarf können die Drähte auch isoliert werden.

Durch diese Modifikation gibt's auch bei hohen Strömen keinerlei Probleme mehr. In nachfolgenden Platinen Revisionen werden diese Elkos schon eingebaut sein.

Wer über die Seite robotikhardware.de das Board/Bausatz bestellt, erhält automatisch diese beiden Elkos mit der Lieferung. Anwender die das Board vor dem 20.12 erhalten haben, können eine kurze Mail an den Support senden (dort unbedingt Rechnungsnummer und Anschrift nennen). Sie erhalten dann kostenlos 2 Kondensatoren (Elkos) nachgeliefert.