R-Tron nRF24L01+ Repeating and Routing Arduino based Nods

Überblick:

Adressierbarkeit von ca. 16 Millionen Nodes
Nodes als Repeater oder Router konfigurierbar
Routing über Haupt und Schutzwege ist möglich
Integrierte Kommando Autorisierung sowohl Master->Node als auch Node->Master
128-Bit Passwort.
Das Passwort wird nur einmal durch das Netz bei der Ersteinrichtung übertragen
Über jeden beliebigen Node kann ein neuer Node ein gebucht werden
Neue Nodes müssen nicht vorkonfiguriert werden

Kleinster Repeater/Router Node läuft auf einem ATMega8
Integrierte IO Funktionen:

- 5x Taster direkt anschließbar
- 26x Taster weitere Taster über Diodenmatrix
- Auf allen Tastern bis zu 7xKlick möglich
- 2x Fixe Outputs, auch als PWM verwendbar
- 6x frei einstellbare I/O
- Frei Einstellbarer Timer1
- Frei Einstellbarer ADC
- Frei Einstellbarer Analog Komparator
- Events für 31 Tasten, ADC und Analog Komparator
- Abfrage aller I/O über Polling vom Master möglich
- Master DDR und PORT force möglich, (High, Low, Toggel)
- Portpreset aus EEPROM möglich
- Direktzuweisung Event zu Output möglich (EEPROM gespeichert)

Serialadapter um das Netzwerk mit einen Steuerserver zu verbinden.

- Interface Serial 115,2 kBit x-on/x-off Adapter->Server
- Format HEX und ASCII Server->Adapter, nur HEX Adapter->Server

Quickstart

Hardwareaufbau

https://arduino-info.wikispaces.com/Nrf24L01-2.4GHz-HowTo
von der Beschreibung abweichend sind die folgenden Verbindungen:

- nRF24L01+ CE → Arduino D8 (ATMega8 PB0)
- nRF24L01+ CSN → Arduino D4 (ATMega8 PD4)
- nRF24L01+ IRQ → Arduino D3 (ATMega8 PD3)

Gaaaanz wichtig der 100µF Kondensator nahe bei der 3,3V Versorgung für den nRF24L01+ nicht vergessen.

Die meisten Arduinobords haben einen Serial2USB Wandler mit drauf. Der wird nur für den Serialadapter benötigt. Ich benutze einen CH340G, der von Linux auch wunderbar erkannt wird. Benutzt man z.B. einen Raspberry-PI als Server kann man dessen Serialport auch direkt mit dem Serialport des ATMega8 verbinden.

Firmware

Dann einfach die Software aufspielen. Entweder die HEX-Files auf je einen ATMega8 Progen, LowFuse 0xDF HighFuse 0xCA. Da muss sonst nix gemacht werden.

- nRF24L01RootingNode.ATMega8.hex (Node Firmware)
- nRF24L01_Seriel.ino.ATMega8.hex (Serialadapter Firmware)

Oder selbst in Arduino erstellen und z.B. auf einen Arduino Nano Progen.

- nRF24L01RootingNode.ino (Node Firmware)
- nRF24L01_Seriel.ino (Serialadapter Firmware)

Es gehen alle Arduinos mit 16MHz und ATMega168 oder ATMega328. Beim Serialadapter muss der Puffer für die Serielle Schnittstelle im Arduinosystem verdoppelt werden.

z.B. C:\Program Files (x86)\Arduino\hardware\arduino\avr\cores\arduino\HardwareSerial.h
#define SERIAL_RX_BUFFER_SIZE 64
ändern in
#define SERIAL_RX_BUFFER_SIZE 128
Arduino muss danach neu gestartet werden.

Linuxsystem als Server

Letztlich benötigt man einen Server, der Tag und Nacht läuft. Ein Windowssystem dafür zu verwenden ist wohl nicht die erste Wahl. Viele Bastler haben eh eine Linuxmaschine laufen, auf dem sich oft ein Fileserver befindet. Da kann man den Manager gut dazu packen.

Zuerst brauchen wir einen Symbolischen Link zu der seriellen Schnittstelle an der unser Serialadapter angeschlossen wird. Da ich einen CH340G verwende habe ich das wie folgt gemacht.
Die Datei /etc/udev/rules.d/10-usb-drives.rules

einfach um die Zeile

```
SUBSYSTEMS=="usb-serial",ACTION=="add", KERNELS=="ttyUSB?", MODE="0777",
DRIVERS=="ch341-uart", SYMLINK+="nRF24L01Net", RUN+="/bin/stty -F /dev/nRF24L01Net
400:0:1cf2:0:3:1c:7f:15:4:0:1:0:11:13:1a:0:12:f:17:16:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0"
```

ergänzt. Sobald ich meinen Serialadapter anstecke wird automatisch der benötigte symbolische Link /dev/nRF24L01Net erzeugt. Das funktioniert aber nur wenn kein anderer CH340G an das Linuxsystem angeschlossen wird.

Auf dem Linuxsystem muss natürlich auch die php5-cli installiert sein.

Unter Debian geht das als Root mit dem Kommando apt-get install php5-cli

Da ich noch am Entwickeln bin habe ich noch absolute Pfade in den PHP-Scrips stehen, deshalb bitte den folgenden Verzeichnispfad anlegen.

/srv/steuerung/scripts

Bitte an die Rechtevergabe denken ! (Quick and Dirty 0777 setzen).
In das Verzeichnis dann die Dateien

- SerialMan.php
- nRFcli.php

kopieren. Auch hier bitte die Rechte vergeben !

Wenn dann alles passt, benutzen wir der Einfachheit halber erst mal zwei Konsolefenster
Im Ersten starten wir den SerialMan.php im Zweiten den nRFcli.php.
Konsole 1 (folgend K1 benannt):

```
root@fileserver:~# /srv/steuerung/scripts/SerialMan.php
Find A2M: e
Find M2A: k
Find BPointer: 72474
nRF24L01Net is pluged.
Starting Reader
```

Konsole 2 (folgend K2 benannt):

```
root@fileserver:~# /srv/steuerung/scripts/nRFcli.php
>nRF24L01Net:
```

Der nRFcli.php hat 2 Sonderbefehle.

- exit (beendet den nRFcli.php)
- stop (beendet den SerialMan.php)

Der erste Kontakt

Nach dem alles eingerichtet und wie oben beschrieben angeschlossen wurde kann es los gehen. In K2 geben wir nun das folgende ein und beenden die Eingabe mit der Entertaste.

```
>nRF24L01Net: 0,ff,ff,f4
```

Auf K1 erhalten wir die folgende Antwort:

```
0,ff,ff,f4
CF:00,FF,0D,F4,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF
```

Mit `0,ff,ff,f4` haben wir die Konfigurationsbytes aus den Serialadapter gelesen. `0,ff,ff` ist die Node Adresse für den Serialadapter. Auf die Systematik der Adresse wird später noch eingegangen. `f4` der Kommandocode zum Lesen der Konfiguration des Serialadapter.
Als Antwort bekommen wir `00,FF,0D,F4,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF.` `00,FF` zeigt an, das es eine Antwort an den Server ist. `0D` ist die Anzahl der nun folgenden Bytes `F4` zeigt den Antworttyp, in diesem Fall die Antwort auf F4.
Die nachfolgenden 12xFF zeigen an, dass der Serialadapter noch unkonfiguriert ist.
Also mal konfigurieren:

```
>nRF24L01Net: 0,FF,FF,F8,0,DF,FF,FF,FF,B0,0B,1E,50,42,41,02,03
```

`0,FF,FF` ist wieder die Node Adresse, `F8` das Kommando um den EEPROM des Serialadapter zu beschreiben. `0` die EEPROM-Startadresse, dann kommen die Daten.
`DF` Sendereinstellung
Bit 0-1: RF24_PA; Bit 2:1Router/0Repeater Bit 3-7 Channel in 4-Step + 3
mehr dazu später.
`FF` LSB Adressteil des Serialadapter der Serialadapter am Server muss immer FF haben.
`FF` LSB Adressteil des Nachbarnode der meine Antworten zum Serverbringt (Main)
`FF` LSB Adressteil des Nachbarnode der meine Antworten zum Serverbringt (Spare)
Da unser Serialadapter direkt am Server hängt bleiben die unkonfiguriert.
`B0,0B,1E,50` ist die Adresse des Netzwerks, die darf frei gewählt werden. Nur in ASCII gesehen darf sie nicht „INIT“ also `48,4D,48,54` sein. Alle zum Netz gehörenden Nodes müssen hier die gleiche Adresse haben.
`42,41,02,03` ist die Routingtabelle des Nodes. Dazu später mehr.

Die Antwort im K1 ist:
`UD`
Geben wir nun in K2 wieder
`>nRF24L01Net: 0,ff,ff,f4`
ein, so erhalten wir in K1 nun

```
0,ff,ff,f4
CF:00,FF,0D,F4,DF,FF,FF,FF,B0,0B,1E,50,42,41,02,03
```

Jetzt buchen wir unseren ersten jungfräulichen Node dazu
Aufgebauten Node mit Strom versorgen. A0 bis A5 am Mega müssen nun High sein, da die PullUp aktiviert werden. Hält man den A4 wären eines Resets gegen GND so geht der Node auch in den Initialisierungsmodus selbst wenn er schon mal Initialisiert wurde.

Nun geben wir das Einbuchkommando 9E an den Serialmanager, Unser neuer Node soll die Adresse 40 bekommen. Das Kommando sieht dann so aus
`>nRF24L01Net: 0,FF,FF,9E,40`

Die Antwort im K1 ist:
`FFX:00,FF,0A,F8,00,DF,40,FF,FF,B0,0B,1E,50`
FFX: Besagt das an einen benachbarten Node mit der LSB Adresse FF (unkonfigurierter Node) das Nachfolgende erfolgreich (X) gesendet wurde. FFL: würde bedeuten der Sendeversuch war nicht erfolgreich.
Wie oben taucht auch in der Datensequenz `F8` auf, sprich der EEPROM des neuen Nodes wurde beschrieben. Damit ist der Knoten ins Netz mit der LSB Adresse 40 ein gebucht.

Jetzt kommt die Security ins Spiel. Das Systempasswort eines frisch initialisierten Knotens ist immer 16xFF. Das Systempasswort steht im EEPROM ab Adresse 0C. Im Serialmanager ist das Systempasswort auf 4x3+12xFF voreingestellt. Das kann man gerne jetzt noch schnell mit einem Editor ändern. Dazu aber ihn erst mit stop anhalten. Ich benutze aber im folgenden das genannte Passwort.
Ich sende nun an meinen neuen Knoten 40 das Kommando F1. Mit dem Kommando F1 macht der neue Node gar nichts aber der Serialmanager kennt damit den neuen Knoten und bekommt auch gesagt, dass das Passwort noch auf 16xFF steht. Warum die LSB Node Adresse 40 zweimal in der Nodeadresse auf taucht erkläre ich später.
`>nRF24L01Net: 0,40,40,F1`
Die Antwort im K1 ist dann:
```
0,40,40,F1
0,40,40,FF,8
T40X:00,40,01,F1
T40X:00,40,02,FF,08
RX:00,FF,12,FE,08,A7,49,ED,C9,05,C4,04,6E,FF,47,B6,42,23,95,0E,FF
Key,46,62,E4,9A,ED,70,C8,2E,9C,D0,CA,D0,F9,5B,90,46
0,40,40,AE,FD,AE,94,89,8E,A4,39,74,F3,D7,B0,74,D0,22,EB,ED,9,46,62,E4,9A
Key: 7EEBC4829D30CF0B046AA5D4BE79D03F
T40X:00,40,16,AE,FD,AE,94,89,8E,A4,39,74,F3,D7,B0,74,D0,22,EB,ED,09,46,62,E4,9
A
T40X:00,40,02,FF,0A
RX:00,FF,12,FE,0A,D7,D5,85,52,2C,FF,B0,8E,34,F5,66,A6,38,02,CA,C8
Key,55,E5,8E,D0,F9,C4,BF,1C,B0,05,2F,08,BE,5E,E8,91
```

Autsch! Wie gut das der Serialmanager das für uns macht.
`0,40,40,F1` War ja unser Kommando. Der Serialmanager fordert nun von Neuen Node einen Parolekey an.
`0,40,40,FF,8`
`FF` ist das Anforderungskommando `8` ist die *Anfrage ID*. Die muss nicht 8 sein. Der Node Antwortet mit
`RX:00,FF,12,FE,08,A7,49,ED,C9,05,C4,04,6E,FF,47,B6,42,23,95,0E,FF`
`RX:00,FF` Empfangene Daten für den Server, `12` Länge, `FE` Kenner für Parolekey `08` die ID der ursprünglichen Anfrage. `A7,49,ED,C9,05,C4,04,6E,FF,47,B6,42,23,95,0E,FF` der angeforderte Key.
`Key,46,62,E4,9A,ED,70,C8,2E,9C,D0,CA,D0,F9,5B,90,46`
Das ist nun der aus dem Passwort 16xFF und dem Parolekey berechnete Autorisierungskey. Den hat der Serialmanager berechnet. Er wird für die Autorisierung von Kommandos Server->Node (S>N) benötigt. Nun muss noch ein Autorisierungskey für die Übertragung Node->Server (N>S) erzeugt werden. AE ist das Kommando dafür.
`0,40,40,AE,FD,AE,94,89,8E,A4,39,74,F3,D7,B0,74,D0,22,EB,ED,9,46,62,E4,9A`
Der Teil `FD,AE,94,89,8E,A4,39,74,F3,D7,B0,74,D0,22,EB,ED` ist der Parolekey dafür

`9` ist die *Anfrage ID*. `46,62,E4,9A` ist ein Teil des S>N Autorisierungskey. Einige Kommandos benutzen nur 4 Bytes des Autorisierungskey so eben auch dieses.

```
Key: 7EEBC4829D30CF0B046AA5D4BE79D03F
```

Das ist nun der aus dem Passwort 16xFF und dem Parolekey berechnete N>S Autorisierungskey. Er dient gleichzeitig als temporäre Nodeidentifikation.

```
T40X:00,40,16,AE,FD,AE,94,89,8E,A4,39,74,F3,D7,B0,74,D0,22,EB,ED,09,46,62,E4,9
A
```

T40X: Zeigt wieder an, dass das Kommando erfolgreich übertragen wurde.

`T40X:00,40,02,FF,0A` fordert nun wieder einen Parolekey an, da ja Teile vom vorigen S>N Autorisierungskey verbraucht wurden.

```
RX:00,FF,12,FE,0A,D7,D5,85,52,2C,FF,B0,8E,34,F5,66,A6,38,02,CA,C8
Key,55,E5,8E,D0,F9,C4,BF,1C,B0,05,2F,08,BE,5E,E8,91
```

Empfang von Parole und Berechnung des neuen S>N Autorisierungskey.

So nun setzen wir für den Node das Systempasswort. Da ja nur die ersten 4xFF zu 4x3 gemacht werden müssen sieht das dann so aus.

```
 >nRF24L01Net: 0,40,40,98,C,3,3,3,3
```

Für die Initialisierten Nodes ist das Kommando zum EEPROM beschreiben nicht F8 sondern `98`, sonst wird einfach ab EEPROM Adresse C 3,3,3,3 geschrieben, Damit ist das gesamte Passwort jetzt 4x3,12xFF.

Die Antwort sieht so aus

```
0,40,40,98,C,3,3,3,3#b,55,E5,8E,D0,F9,C4,BF,1C,B0,05,2F,08,BE,5E,E8,91
T40X:00,40,17,98,0C,03,03,03,03,0B,55,E5,8E,D0,F9,C4,BF,1C,B0,05,2F,08,BE,5E,E
8,91
T40X:00,40,02,FF,0C
RX:00,FF,12,FE,0C,FE,A8,B8,85,90,BB,FA,79,53,01,13,64,ED,E7,80,04
Key,7F,B4,74,94,6A,02,71,DE,90,A4,F2,88,4E,89,32,30
```

Schön zu sehen das ab #b die Autorisierung angehängt wird, b ist wieder die *Anfrage ID*, der Rest der zuvor erzeugte S>N Autorisierungskey. Darauf folgt wieder die Erzeugung eines neuen Key.

Mit dem Kommando 9D wird nun das neue Passwort aktiviert.

```
>nRF24L01Net: 0,40,40,9D
```

Die Antwort lautet.

```
0,40,40,9D#d,7F,B4,74,94,6A,02,71,DE,90,A4,F2,88,4E,89,32,30
T40X:00,40,12,9D,0D,7F,B4,74,94,6A,02,71,DE,90,A4,F2,88,4E,89,32,30
T40X:00,40,02,FF,0E
RX:00,FF,12,FE,0E,E3,4C,01,8C,43,1F,F8,AA,77,8C,FC,5B,5E,33,57,3F
Key,C0,5B,4D,8A,CC,8C,84,E9,EC,4B,1B,54,16,CB,70,1F
```

wer die Antwort nicht versteht bitte nochmal oben lesen.

Mit dem Kommando F0 wird nun dem Serialmanager mitgeteilt, dass er nun für den Node 0,40,40 das Systempasswort verwenden soll. Das Kommando kann auch für die Resynchronisation zwischen Server und Node verwendet der den. F0 ist auch ein Pseudokommando mit dem der Node nichts macht.

```
>nRF24L01Net: 0,40,40,F0
```

Die Antwort lautet.

```
0,40,40,F0
0,40,40,FF,f
T40X:00,40,01,F0
T40X:00,40,02,FF,0F
RX:00,FF,12,FE,0F,4C,87,5C,8C,F8,B1,F4,F4,CF,28,9D,6B,BA,FC,22,E0
Key,C0,68,F8,68,9D,8D,98,C0,E4,EE,C6,01,94,62,EB,46
```

```
0,40,40,AE,84,98,68,24,5E,39,BC,22,11,25,25,80,8C,D4,72,35,10,C0,68,F8,68
Key: E35461346FF083B3CB9941C814AD0E54
T40X:00,40,16,AE,84,98,68,24,5E,39,BC,22,11,25,25,80,8C,D4,72,35,10,C0,68,F8,6
8
T40X:00,40,02,FF,11
RX:00,FF,12,FE,11,68,8B,19,09,5E,77,CA,A8,FD,01,49,BB,04,50,96,A9
Key,39,B2,C2,55,DC,20,D5,06,02,E3,78,16,81,A8,4E,EB
```
Die Antwort solltet ihr euch nun selbst auf drösel können.

Nun fix eine LED an PB1 Arduino D9 mit Vorwiderstand an GND.
```
>nRF24L01Net: 0,40,40,24,7D,3D,10,7C,3C
```
Die Antwort lautet.
```
0,40,40,24,7D,3D,10,7C,3C#12,39,B2,C2,55,DC,20,D5,06,02,E3,78,16,81,A8,4E,EB
T40X:00,40,17,24,7D,3D,10,7C,3C,12,39,B2,C2,55,DC,20,D5,06,02,E3,78,16,81,A8,4
E,EB
T40X:00,40,02,FF,13
RX:00,FF,12,FE,13,A2,10,44,F7,F3,EF,37,17,3D,E8,14,63,CF,C9,14,EF
Key,AB,CA,A7,BB,34,FF,24,4F,A1,13,AB,22,95,82,F6,D7
```

Wenn alles passt leuchtet die LED für 2 Sekunden auf.
Hier endet nun der Quickstart.

Ich hoffe, ich komm zum weiteren dokumentieren bis die Ersten den Nachbau ferig haben.
Es folgt dann die Beschreibung der I/O Funktionen und der möglichen Presets im EEPROM. Sowie der Aufbau der Nodeadressierung und dem Repeating und Routing. Vermaschung und Nutzung von Regel und Schutzwegen.

Aufbau der Frames

Die ersten 3 Bytes sind der Framehaeder.
Im 4. Byte steht immer das Kommando. Es maßgeblich vorgibt welche Länge die Autorisierung hat.
Kommandos von 00 bis 9F benötigen eine 16 Byte Autorisierung.
Kommandos von A0 bis DF benötigen eine 4 Byte Autoriseirung.
Kommandos von E0 bis FF benötigen keine Autorisierung.
Jeder Frame vom Server zum Node enthält direkt nach der Kommandosequenz noch zusätzlich eine, ein Byte lange, Anfrage ID. Frames vom Node zum Server besitzen diese nur bei besonderen Kommandos.
Darauf folgt die Autorisierung.
Frames vom Node zum Server haben in der Regel eine 16 Byte Autorisierung die zugleich die temporär ID des Nodes ist.

Ein Frame darf insgesamt niemals länger als 32 Byte werden.

Frame Server zu Serialadapter

1.Byte	2.Byte	3.Byte	4.Byte	(5.Byte)	…
Routingbyte	LSBAdr	Sendover	Kommando	(Daten...)	ID (& Aut)
0	40	41	xx	(xx)	…

Wenn im Sendover nicht die *MyAdr* (LSBAdresse) des Serialadapters steht, sendet der Serialadater den gesamten Frame an den Node mit der LSBAdresse in *Sendover*, hier an den Node mit der 41. Zuvor wird aber *Sendover* durch *Länge Daten* ersetzt, die der Serialadapter selbst berechnet.

Alle anderen Frames

1.Byte	2.Byte	3.Byte	4.Byte	(5.Byte)	…
Routingbyte	LSBAdr	Länge Daten	Kommando	(Daten...)	((ID) & (Aut))
0	40	1A	xx	(xx)	…

Länge Daten ist die Anzahl der nach *Länge Daten* folgenden Bytes, wenn vorhanden, inklusive der *Anfrage ID* und der Autorisierung.
Nur wenn das *Routingbyte* 0 ist und *LSBAdr* = *MyAdr*, dann wird der Frame als eigener erkannt. In jedem anderen Fall wird der Frame repeatet oder geroutet.

Nodeadresse

Bei Frames vom Server zum Serialadapter enthält der Framehaeder quasi nur die Adresse des Zielnodes. Der erste Ordner der Adresse ist logisch betrachtet *Sendover,* da *Sendover* den ersten Knote nach dem Serialadapter bestimmt. Danach folgt das *Routingbyte*, es bestimmt wie das Paket durch das Netzwerk läuft. Erst wenn das *Routingbyte* durch die Verarbeitung der routenden Nodes zu 0 wurde, wird die *LSBAdr*

berücksichtigt. Beginnt also ein solcher Frame mit 0,40,40 so wird der Node mit der logischen Adresse 400040 adressiert. Alle Nodes die Adressenmässig direkt nach dem Serialadapter kommen haben als ihre Adresse *MyAdr:0:MyAdr bzw. im Frame 0,MyAdr,MyAdr.*

Zuordnung der EEPROM Speicherzellen

Byte 0 bis B

Dies Speicherzellen werden sowohl vom Serialadapter als auch von den Nods verarbeitet.

0	1	2	3	4	5	6	7	8	9	A	B
SetId	MyAdr	Main	Spare	Systemadresse				Routingtabelle			
DF	40	FF	FF	B0	0B	1E	50	41	46	48	40

SetId (Byte 0, Wert DF)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Trägerfrequenzkanalindex					Router	Sendepegel	
Dezimal 27					1	3	
1	1	0	1	1	1	1	1

Trägerfrequenzkanalindex:
Der Kanal berechnet sich aus:
Kanal = *Trägerfrequenzkanalindex* x 4 + 3 (27x4+3=111)

Router:
Ist der *Router* = 1 dann wird das *Routingbyte* (Erstes Byte in jedem Frame) verarbeitet.
Routingbyte:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Index 4. Router		Index 3. Router		Index 2. Router		Index 1. Router	
1		0		3		2	
0	1	0	0	1	1	1	0

Wenn das *Routingbyte* nicht 0 ist, dann nimmt er den Wert in *Index 1. Router* vom *Routingbyte*. Index 2. wird dann zu Index 1., 3 zu 2, 4 zu 3.
Den so veränderten Frame sendet er dann an den entsprechenden Node aus der Routingtabelle. In unserem Fall ist der Index 2, das wäre dann an den Node 48.
Wird in der Routigtabelle auf einen Wert gezeigt, der gleich *MyAdr* ist wird der Frame an den Node in *LSBAdr* aus dem Frame gesendet. (EndPointSpeider)
Ist das *Routingbyte* = 0 und *MyAdr* nicht gleich *LSBAdr* wird der Frame an den Node im Index 0 gesendet.
Ist der *Router* = 0 (Repeater) wird der Frame nicht verändert und einfach an den Node im Index 0 gesendet. Hier dann Node 41.

Sendepegel:
Beim nRF24L01+ kann man 4 Sendepegel einstellen 3 ist maximal, 0 ist minimal

MyAdr:
MyAdr ist die LSBAdresse des Nodes, ist das *Routingbyte* gleich 0 und *MyAdr* = *LSBAdr* aus dem Frame, dann nimmt der Node die Daten für sich.

Main:
Alle Frames die mir 00 FF beginnen werden an den in *Main* eingetragenen Node gesendet. Diese Frams müssen alle den Weg zum Server finden.

Spare:
Scheitert das senden eines 00 FF Frames an *Main* so wird dieser an den unter *Spare* eingetragenen Node gesendet.

Systemadresse:
Diese 4. Bytes müssen auf allen Nods gleich sein. Die Kodierung 484D4854 darf nicht verwendet werden. Sie ist die Grundeinstellung wenn ein Knoten nicht Initialisiert ist.

Routingtabelle:
Siehe Oben.

Byte C bis 1B (Systempasswort)

Der Serialadapter nutzt diese und alle nachfolgenden Bytes nicht.
In diesen 16 Bytes Steht das *Systempasswort*.

Byte 1C (TasterBlock)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
(A7)	(A6)	A5	A4	A3	A2	A1	A0
1	1	1	1	1	1	1	1

Unkofiguriert steht auf allen 1 und sind damit gesperrt. Wird ein Bit auf 0 gesetzt so kann der Node ein Event an den Server senden, wenn eine angeschlossene Taste gedrückt wird.

Byte 1D (PrePORTC)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
(A7)	(A6)	A5	A4	A3	A2	A1	A0
1	1	1	1	1	1	1	1

Unkofiguriert sind alle PullUps eingeschaltet.

Byte 1E (PreDDRD)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PD7	PD6	PD5	na	na	PD2	PD1	PD0
1	1	1	X	X	1	1	1

Unkofiguriert sind alle Pins als Eingang geschaltet. Sonst ist dort wo eine 1 ist der Port ein Ausgang. FF macht das Gleiche wie 00.

Byte 1F (PreACSR)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ACD	ACBG	na	ACI	!Event	ACIC	ACIS1	ACIS0
1	1	X	1	0	1	1	1

Bitte die Beschreibung der Dokumentation zum ATMega8 entnehmen.
!Event:
!Event lässt sich nicht aktivieren. Kann aber vom Server aus gesetzt werden. Ist der Wert 1 so wird kein Event an den Server gesendet. ACD ist auch erst mal 1, damit wird auch kein Event ausgelöst.

Byte 20 (PreADCSRA)

Ist der Wert gleich FF dann ist der AD-Wandler abgeschaltet. Sonst entspricht es der ADCSRA Dokumentation vom ATMega8,
ATmega8 Datenblatt
nur Bit 3 kann nicht gesetzt werden.

Byte 21 (PreADMUX)

Entspricht der ADMUX Dokumentation vom ATMega8.

Byte 22 bis 25

Byte 22	Byte 23	Byte 24	Byte 25
PreADLowLimitH	PreADLowLimitL	PreADHighLimitH	PreADHighLimitL

PreADLowLimit:
Wird der Wert unterschritten so wird ein Event gesendet.
PreADHighLimit:
Wird der Wert überschritten so wird ein Event gesendet.

Bytepaare ab 26 (direkt Aktion)

Es gehören immer 2 Bytes zusammen. Gelesen wird, bis das erste Byte den Wert FF hat.
Das Erst Byte enthält den *direkt Aktion* Code das Zweite den *PortSetCode*.
Damit kann ein Event, ohne Zutun des Servers, den Status eines PortPin verändern.

Die Kommandocods

Ein Kommadocode steht immer an auf den 4.Byte des Frames

00 (IO_Status) Server > Node

Der Adressierte Node sendet folgendes zurück:
0,FF,1D,00,PINB, PORTB, DDRB, PINC, PORTC, DDRC, PIND, PORTD, DDRD, ACValue, ADValueH, ADValueL

01 (IO_Tasten) Node > Server (Event)

Nach einem Event sendet der Node folgendes zurück:
0,FF,15,01,*EventCode*, *Counter*, *ADValueH*, *ADValueL*

Events:

EventCode	Counter	direkt Aktion	
02	0	10	Unterschreiten von ADLowLimit
03	0	18	Überschreiten vom ADHighLimit
04	0	20	AC wurde negativ
05	0	28	AC wurde positiv
Taster	Klicks	Taster x 8 + Kicks	

Taster an	EventCode
A0	1E
A1	1D
A2	1B
A3	17
A4	0F

02 (IO_PortSet) Server > Node

Keine Antwort vom adressierten Node
Nach dem Kommando können bis zu 11 *PortSetCods* angegeben werden.
PortSetCods:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Toggel	DDR	Port (1=B,2=C,3=D)		Pin des Ports 0-7			Status

Wenn *Toggel* gestzt bitte *Status* immer 0 setzen. *DDR* = 0 schreibt auf PORT

0E (O_Tstatus) Server > Node

Der adressierte Node sendet folgendes zurück:
0,FF,13,0E,ICR1L,ICR1H (siehe ATmega8 Datenblatt)

0F (IO_ReadSet) Server > Node

Nach dem Kommando gibt man die EEPROM Adresse an, ab der gelesen werden soll.

EEPROM Adresse	00 - 1B	1C - FF	100 - 11B
Wert nach Kommando		1C - FF	00 - 1B

Der adressierte Node sendet folgendes zurück:
0,FF,1D,0F,[12 Bytes startend ab der angegebenen Adresse]

10 (IO_ADCSetup) Server > Node

Keine Antwort vom adressierten Node
Nach dem Kommando müssen 3 Bytes angegeben werden.

1.Byte	2.Byte	3.Byte
ADCSRA bevor ADMUX eingestellt wird	ADMUX	ADCSRA nachdem ADMUX eingestellt wurde

ADCSRA und ADMUX entsprechen der Beschreibung in der Dokumentation zum ATmega8. Bei ADCSRA kann das Bit 3 nicht gesetzt werden.

11 (IO_ADCLimits) Server > Node

Keine Antwort vom adressierten Node
Nach dem Kommando müssen 4 Bytes angegeben werden.

1.Byte	2.Byte	3.Byte	4.Byte
ADLowLimitH	ADLowLimitL	ADHighLimitH	ADHighLimitL

Wird vom AD-Wandler der Wert *ADLowLimitH*:*ADLowLimitL* unterschritten so tritt das Event 02:00 ein. Der *EventCode* dazu ist 10.
Wird vom AD-Wandler der Wert *ADHighLimitH*:*ADHighLimitL* überschritten so tritt das Event 03:00 ein. Der *EventCode* dazu ist 18.

14 (IO_ACSetup) Server > Node

Keine Antwort vom adressierten Node
Nach dem Kommando muss ein Byte angegeben werden.
Das Byte entspricht der Dokumentation zum ATmega8 für das Register ACSR.
In ASCR kann das Bit 5 und das Bit 3 nicht gesetzt werden.
Das Bit 3 hat aber die Funktion von *!Event*.
Setzt man das Bit *!Event* auf 1. So werden keine Events vom Analog Komparator erzeugt. Ist das Bit 0, so werden folgende Events erzeugt.
Der Komperatorvergleich wird negativ so tritt das Event 04:00 ein. Der *EventCode* dazu ist 20.
Der Komperatorvergleich wird positiv so tritt das Event 05:00 ein. Der *EventCode* dazu ist 28.

20 (IO_PWMSetup) Server > Node

Keine Antwort vom adressierten Node
Nach dem Kommando müssen 4 Bytes angegeben werden.
TCCR1A, TCCR1B, ICR1H, ICR1L (ICR wird vor TCCR geschrieben)
Die Bytes entsprechen der Dokumentation zum ATmega8 für die genannten Register.

21 (IO_PWMOC1A) Server > Node

Keine Antwort vom adressierten Node
Nach dem Kommando müssen 2 Bytes angegeben werden.
OCR1AH, OCR1AL
Die Bytes entsprechen der Dokumentation zum ATmega8 für die genannten Register.

22 (IO_PWMOC1B) Server > Node

Keine Antwort vom adressierten Node
Nach dem Kommando müssen 2 Bytes angegeben werden.
OCR1BH, OCR1BL
Die Bytes entsprechen der Dokumentation zum ATmega8 für die genannten Register.

24 (IO_PortPulse) Server > Node

Keine Antwort vom adressierten Node
Nach dem Kommando müssen 5 Bytes angegeben werden.
PortSetCode, *PortSetCode*, *Wait*, *PortSetCode*, *PortSetCode*

Das Kommando führt die beiden ersten *PortSetCode* aus und führt dann *nach Wait* x 0,128 Sekunden die letzten beiden *PortSetCode* aus. Will man einen *PortSetCode* nicht nutzen, so trägt man einen Code ohne Wirkung ein, z.B. 0.

94 (CG_ReadSet) Server > Node

Der adressierte Node sendet folgendes zurück:
0,FF,1D,94,[Die 12 Bytes der Nodekonfiguration]

98 (CG_SetSet) Server > Node

Keine Antwort vom adressierten Node
Nach dem Kommando muss die Adresse der ersten zu beschreibenden EEPROM Zelle folgen danach folgt mindestens ein Datebyte höchstens 10. Damit kann im EEPROM Zelle 0 bis 109 beschrieben werden.

9C (NN_DefaultPass) Server > Node

Keine Antwort vom adressierten Node
Veranlasst den adressierten Node das Defaultpasswort
(FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF) zu verwenden.

9D (NN_UserPass) Server > Node

Keine Antwort vom adressierten Node
Veranlasst den adressierten Node das Passwort welches im EEPROM 0C – 1B steht zu verwenden.

9E (NN_Addnode) Server > Node

Keine Antwort vom adressierten Node
Nach dem Kommando muss die zukünftige *MyAdr* des zu initialisierenden Nodes angegeben werden. Der adressierte Node sendet auf der Initialeinstelungseinstellung einen NN_SetInit Frame, der den zu initialisierenden Knoten zu seinem Nachbarn macht und vergibt ihm die angegebene *MyAdr.*

AE (AU_RootKey) Server > Node (ab jetzt wird 4Byte Security verwendet)

Keine Antwort vom adressierten Node
Nach dem Kommando müssen die 16 Bytes der Server ParoleKeys folgen.
Es wird der N>SAutorisierungskey syncronisiert.

F4 (NN_ReadInit) Server > Node (ab jetzt wird keine Security verwendet)

Ausschließlich nur zum lesen der Konfigurationsbytes des Serialadapters verwendbar.
Der Serialadapter antwortet mit
CF:00,FF,0D,F4,[Die 12 Bytes der Konfiguration des Serialadapters]

F8 (NN_SetInit) Server > Node

Wie Kommando 98, gibt es nur für den Serialadapter auf der Serielenschnittstelle, oder nur netzintern zur Initialisierung eines Nodes.

Der Serialadapter antwortet mit UD.

FC (AU_Reject) Node > Server

Ein Node meldet das ein Kommando wg. fehlerhaften Autorisierung abgewiesen wurde.
Der Node sendet 0,FF,2,FC,Anfrage ID an den Server.

FE (AU_SendBackKey) Node > Server

Ein Node Antwortet damit auf das Kommando FF und sendet den NodeParoleKey
0,FF,12,FE,Anfrage ID,16 Bytes des NodeParoleKeys
Der Server synchronisieren den S>N Autorisierungskey

FF (AU_RequestKey) Server > Node

Der Server fordet vom Node ein NodeParoleKey an.
Der adressierte Node synchronisieren den S>N Autorisierungskey und antwortet mit Kommando FE.