R-Tron nRF24L01+ Repeating and Routing Arduino based Nods

Überblick:

Adressierbarkeit von ca. 16 Millionen Nods
Nods 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 eingebucht werden
Neue Nods müssen nicht vorkonfiguriert werden

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

- 5x Taster direkt anschaltbar
- 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 AnalogComperator
- Event-Massages für 31 Tasten, ADC und AnalogComperator
- 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 Adaper->Server
- Format HEX und ASCII Server->Adapter, nur HEX Adaper->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 Firware)
- nRF24L01_Seriel.ino.ATMega8.hex (Serialadapter Firmware)

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

- nRF24L01RootingNode.ino (Node Firware)
- 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 Linuxmaschiene 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 Serialadaper 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 Serialadaper 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 ! (Qick 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 K2 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 Serialadpter 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 Serialadapters.
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 Serialadapers 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 Serialadapters 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 unkofiguriert.
`B0,0B,1E,50` ist die Adresse des Netzweks, die darf frei gewählt werden. Nur in ASCII gesehen darf sie nicht „INIT“ also 48,4D,48,54 sein. Alle zum Netz gehörenden Nods müssen hier die gleiche Adresse haben.
`42,41,02,03` ist die Routingtabelle des Nods. Dazu später mehr.

Die Antwort im K1 ist:
```
UD
```
Geben wir nun in K2 wierder
```
>nRF24L01Net: 0,ff,ff,f4
```
ein, so erhaten 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 Initalisierungsmodus 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 (unkofigurierter 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 Nods wurde beschrieben. Damit ist der Knoten ins Netz mit der LSB Adresse 40 eingebucht.

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 Systempaswort 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,E4,9A
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 Kommabdos Server->Node (S>N) benötigt. Nun muss noch ein Authorisierungskey 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 Authorisierungskeys. Einige Kommandos benutzen nur 4 Byts des Authorisierungskeys 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,E4,9A`
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>NAuthorisierungskey 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>NAuthorisierungskeys.

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 Nods 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,E8,
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>NAuthorisierungskey. Darauf folgt wieder die Erzeugug eines neuen Keys.
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 9D wird nun das neue Passwort aktiviert.
Mit dem Kommando F0 wird nun dem Serialmanger mitgeteilt, dass er nun für den Node 0,40,40 das Systempasswort verwenden soll. Das Kommando kann auch für die Resyncronisation zwischen Server und Node verwendet derden. 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,F8,68
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 aufdrösel 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,A8,4E,
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.