ScannerList

Wolfgang Hallmann – DF7PN – ©2020 ff. – Version 1.2 Nov 2020

(englisch below)

Für was?

Zur Verfolgung von Wettersonden im Frequenzbereich 400-406 MHz nutzen viele das Softwarepaket von OE5DXL – Chris, einem Funkamateur aus Österreich. Mit dem Softwarepaket kann man unter Unix und Windows und einem DVBT-Stick die Sonden empfangen und deren Daten auswerten.

GitHub: https://github.com/oe5hpm/dxlAPRS

ScannerList unterstützt das Programm von Chris, welches eine Liste von Frequenzen gleichzeitig scannt, die man vorher gefüllt haben muss. Dieses Füllen klappt sogar während das Programm läuft. Bedeutet: man kann die Liste jederzeit korrigieren und sie wird sofort verwendet ohne Programmunterbrechung.

Einige haben sich Scripte gebastelt, die mit Hilfe des Programms „*rtl_power*“ einen Bereich scannen und die Empfangswerte umrechnen in Frequenzen, wenn ein brauchbarer Dezibel-Wert ausgegeben wird. Diese Auswertung übernimmt „*ScannerList*“ mit einigen Annehmlichkeiten.

Warum sowas?

Eigentlich könnte man auch ohne *ScannerList* auskommen. Aber einen Sinn muss es ja haben, werdet ihr denken, dass es das Teil gibt. Also hier die Vorteile:

Eigentlich füllt man in die Konfig-Datei sdrcfg der dxl-Software Zeile für Zeile alle Frequenzen die man das Programm überwachen lassen will. Es gibt einige Frequenzen die immer zuverlässig benutzt werden. Was aber mit den vielen anderen? Gerade die DFM-Modelle des Militärs senden alle Nase lang wo anders. He he macht das mal, schaufelt die Liste mal ordentlich voll und dann schaut auf die CPU Auslastung. Alle Angaben müssen ausgewertet werden. Das kostet CPU Zeit und hält das Zimmer warm. Das muss nicht sein.

Mit *ScannerList* werden nur die wirklich gerade aktiven Frequenzen zur Überwachung angewiesen. Die sdrcfg bleibt klein und die CPU kalt. Will man auf Nummer-sicher-gehen, kann man in eine Liste trotzdem feste Frequenzen hinterlegen, also die, die immer täglich aufschlagen. Dann ist der Decoder halt gleich beim ersten Piep am Arbeiten. Läuft *ScannerList* aber alle 1-2 Minuten per Cron-Job ist das zeitlich kein großer Verlust.

Damit *ScannerList* keinen falschen Alarm gibt und evtl. doch mal Störungen nicht erkennt und sinnlos zur Decodierung in Auftrag gibt, kann auch eine Blacklist bestückt werden.

Funktionsweise

Für einen Scanner benötigt man einen dedizierten DVBT-Stick der nur dafür zur Verfügung steht.

Gemessen wird in Intervallen, nicht im Dauerbetrieb.

Wir benötigen ein Script (Beispiel: scanner_prod.sh) welches per CRONJOB alle 2 Minuten (bei mir) aufgerufen wird. Es startet den „rtl_power“-Scan mit einem Block von 402 bis 404 MHz, danach läuft „*ScannerList*“ und werte die csv-Datei von rtl_power aus. Erstellt eine für das DXL-Paket konforme

Frequenzliste als Datei. Diese wird per SCP (Filetransfer) auf den jeweiligen Rechner übertragen. Die gleichen Schritte laufen dann nochmal für 404 bis 406 MHz ab. Die Frequenzliste geht an einen anderen Rechner. Warum nur 2 MHz? Weil die DVBT-Sticks i.d.Regel nur diesen Bereich maximal mit einem Rutsch abbilden können.

Der wirklich interessante Schritt ist die Auswertung der Dezibel-Werte vom rtl_power und daraus Sonden heraus zu lesen, von Störungen zu unterscheiden, die weniger Interessant sind.

Signalerkennung

Kurz: wie scannt rtl_power? Schaut man sich die Parameter an, gibt man nicht nur den Frequenzbereich an, sondern auch das Intervall in dem zwischen den Bereichen abgetastet werden soll. Ich empfehle hier 1.000 Herz-Schritte (1 Khz).

Zusammen mit den üblichen Angaben für DEVICE-NR, GAIN und FREQUENZKORREKTUR erhält man eine Ausgabe im CSV-Format. Vorne stehen die Parameter mit denen empfangen wurde und danach folgen für jeden Step (1 KHz) Dezibelwerte. Diese werden in negativen Zahlen angegeben. Nimmt ein Signal zu (wird besser), steigt der Wert bis gegen 0 an.

Kurz: wie sehen Sonden-Signale aus? Nimmt man ein SDR-Programm und lässt es auf ein solches Signal los, sieht man eine belegte Breite von zwischen ungefähr 6 bis 14 Khz – je nach Sondentyp.

Der Scanner tastet im 1 KHz Raster ab und wird daher nebeneinander mehrere gute Signalwerte sehen (also hier 6 bis 12 Stück). Genau das macht sich das Programm zunutze um Störsignale erkennen zu können. Störungen sind in der Regel einzelne recht schmalbandige, meist unmodulierte Träger ohne Informationsinhalt. Kurzum, sie fallen aus dem Raster.

Wie erkennt man nun ob ein einzelner Messwert ein Signal darstellt? Ich bekomme ja nur eine Zahl. Noch dazu bekomme ich immer eine Zahl, auch wenn nur Rauschen – also kein Signal - vorhanden ist.

Gute Frage. Dazu muss man entweder wissen, welche Werte nur Rauschen darstellen und ab wann ein brauchbares Signal aus dem Rauschen hervortritt, oder man ermittelt den Durchschnittswert über 100 Daten und nimmt ihn als Rauschpegel. Hier hilft ein Blick in die CSV Datei die uns rtl_power ausgehändigt hat. Die CSV Datei kann man in EXCEL oder ähnlichen Programmen laden und betrachten. Auf den meisten Frequenzen wird prozentual gesehen nur Rauschen sein. Mit dem Auge findet man das ziemlich schnell. Wann ist ein Signal dekodierbar von der dxl-Software? Dazu kann man sich die Mühe machen und mit einem zusätzlichen Stick an gleicher Antenne parallel ein SDR Programm laden und auf der gleichen Frequenz zuhören. Im Blick hält man die dxl-Software und achtet darauf, ab wann der Pegel zur Dekodierung ausreichend ist (Log von sondeudp im /tmp Pfad). Im SDR-Empfänger liest man jetzt den Wert ab. In der Praxis ist das sehr oft ein Wert der 5 bis 10 Dezibel besser als das Rauschen ist.

ScannerList kann man jetzt über zwei Methoden nach Signalen suchen lassen. Einmal über den Parameter -L (fester Wert über Rauschen z.B: Rauschen = -35 db, wäre dann -25 db ein guter Wert). Alles in Richtung Null wäre ein Signal was sich zu beobachten lohnt.

Die andere Methode ist eine Automatik. Hier wird über Parameter -n ein relativer Dezibel-Wert angegeben, ab dem ein Signal über dem Rauschen näher betrachtet werden soll. Z.B. -n 5 (positive Zahl). Rauschen = -35, Signale die 5 db besser sind wären gut: ab -30 db ist das so.

Nun hat dies den Vorteil, dass sich der Rauschpegel im Laufe eines Tages auch mal ändern kann (Störungen etc). Dazu wurde eine Untersuchung des Rauschpegels pro 100 Messpunkte eingeführt. Der Pegel wird für jedes Segment herangezogen. Ist dann wieder von Vorteil, wenn der Rauschpegel z.B. in einem größeren Frequenzbereich höher wäre z.B. -20 db und dadurch gegenüber der Methode

mit -L bei fixer Vorgabe schon in die nähere Betrachtung gezogen würde, weil er die Signalschwelle (aus Beispiel oben von -20) erreicht. Ganz plötzlich laufen dann die Listen voll obwohl dort nichts ist. Gut, hier werden dann keine breiten Signale erkannt und die Muster fallen wieder durchs Raster. Kostet aber unnötige CPU-Zeit.

So wer hier noch folgen kann, wird bestimmt die automatische Rauschpegelerkennung nutzen wollen. Wer hier nichts verstanden hat, dem rate ich das jetzt auch. Nehmt den Parameter „-n 5“, dann müsste das funktionieren.

Weitere Erkennungsattribute

Signale vom Rauschen zu trennen, war der erste Schritt. Wir schauen jetzt, ob mindestens 4 Werte nebeneinander in die Erkennung gekommen sind. Ab jetzt wird es interessant. Es wird die Mittenfrequenz errechnet. Höchste Frequenz minus niedrigste Frequenz = Bandbreite. Niedrigste Frequenz + halbe Bandbreite = Mitte.

Nun haben die Sonden ein Sende-Frequenzraster von minimal 10 KHz. D.h. bei einigen KHz Abweichung kann man trotzdem guten Gewissens auf volle 10 KHz auf- oder abrunden. Das ist jetzt die Frequenz, die wir uns merken – hier ist Betrieb – darauf lassen wir den dxl-Empfänger los.

Signal kurz weg – kein Problem

Durch Funk-Abschattungen kann ein Signal auch mal kurzzeitig verschwinden, kommt aber gleich wieder. Um zu vermeiden, dass der Scanner das mal für einige Minuten aus der Frequenzliste herausnimmt, gibt es in ScannerList noch eine Holdingliste. Sie merkt sich sämtliche Frequenzen für einen kleinen Zeitraum, den Sie per Parameter wählen können. Er steht ohne Angabe intern immer auf 10 Minuten, kann aber durch „-h“ verändert werden. Ich empfehle aus der Praxis Werte zwischen 5-15 Minuten.

Frequenzliste per SCP übertragen

Wer wie ich 3 Rechner betreibt und einer davon den Scanner betreibt, fragt sich bestimmt wie er die sdrcfg-Dateien an die anderen Rechner übertragen kann, ohne das ein manueller Eingriff notwendig wird. Das geht mit dem integrierten Zeilen-Kommando „SCP“.

Wenn ihr das mal manuell probiert, werdet ihr sehen, dass hier Eingaben verlangt werden (Bestätigungen). Das können wir nicht gebrauchen, da alles zeitgesteuert in einem Script ablaufen soll.

Das lässt sich durch die Nutzung eines Zertifikats einstellen.

Einmalig geben wir an der Quelle ein:

- ssh-keygen (kein PW angeben)
- ssh-copy-id user@zielrechner
  user ist der Benutzer z.B: „pi“
  zielrechner könnte eine IP-Adresse sein.
  mit „ssh-copy-id“ wird das in erster Zeile erstelle Zertifikat auf den anderen Rechner kopiert.
- scp kopiert dann künftig ohne Fragen zu stellen, da sich beide Rechner „kennen“:
  (kein SUDO verwenden, weil dann User=root kein Zertifikat hat)
  scp /pfad/dateiname user@rechnerZiel:/pfad/{zieldateiname]}

Kommandozeilen Parameter für ScannerList

Parameter	Angabe	Standard	Einheit	Erklärung
-a	1…10	5	KHz	Automatische Frequenz Kontrolle. Wenn Sender nicht genau auf der Sollangabe sendet, wird er innerhalb von 5 KHz tiefer oder höher verfolgt. Es ist der zweite Wert im Aufbau der sdrcfg.txt
-b	5000…15000	Autom.	Herz	Wird kein -b gefunden, erfolgt die Berechnung der Bandbreite automatisch. Gibt man -b an, werden alle Sonden mit dieser Bandbreite empfangen. Empfehlung: Automatik oder 8000. Bei M10 Sonden wird empfohlen diese in die whitelist mit höherer Bandbreite (17) aufzunehmen. Sie kommen selten vor.
-d	Dateiname	Kann Angabe	Zeichen	BLACKLIST // Dateiname mit Pfad kann angegeben werden. Datei enthält pro Zeile eine Frequenz, die nie in die Empfangsliste kommt. Angabe in KHz z.B. 400020. Auf volle 10 KHz auf/abrunden
-f	Dateiname	MUSS	Zeichen	Dateiname mit Pfad zur CSV Datei, die rtl_power erstellt hat.
-h	5…15	10	Minuten	Solange werden Frequenzen weiter in die sdrcfg Datei abgegeben, auch wenn sie gerade nicht empfangen werden. Halteliste-Timer
-H	Dateiname	MUSS	Zeichen	Dateiname mit Pfad zu einer Datei, in der gehörte Frequenzen mit zuletzt gehörtem Zeitstempel verwaltet werden. Wird vom Programm gefüllt – nicht ändern
-L	-50…-15	-20	Dezibel db	Manuelle Signalerkennung. Empfangswerte, die über dem angegebenen Pegel liegen, werden als Signal erkannt. Mit „Über“ ist in Richtung „0“ gemeint. Rauschen liegt ca. bei minus 40 bis minus 30 db. Somit liegt -20 db höher. Zu beachten ist, dass hier nicht die Höhe des Rauschpegels beachtet wird.
-n	3…20 empfohlen: 5	Kann Angabe	Dezibel db	Automatische Signalerkennung unter Beachtung des aktuellen Rauschpegels. Angabe ist relativ zum Rauschpegel zu sehen. Wird -n angegeben, so wird -L ignoriert Pro 100 KHz wird der durchschnittliche Rauschpegel gesichert. Ein Wert von „5“ bedeutet, dass Signale dann erkennt werden, wenn der für diesen Block gemerkte Rauschpegel plus „5“ db höher liegt. Z.B. Rauschen = -35 db, Signale sind gut ab -30 db.
-o	Dateiname	MUSS	Zeichen	Frequenzliste AUSGABE für dxlAPRS-Chain Format entspricht den Anforderungen z.B. “F 405.700 5 0 0 8000” 5 = Wert aus „-a” / 8000 = Bandbreite aus Automatik oder manueller Angabe „-b“

-q	0..99 Empfohlen 80 +/-10	0	Prozent	Squelch-Level, wird in die SDRCFG.TXT mit übernommen und wirkt sich auf die CPU Last von sdrtst aus. Squelch offen = 0, maximale Last. Squelch leicht geschlossen = 90 …10= stark geschlossen. Muss man ausprobieren und die CPULast beobachten. Immer wenn man die SDRCFG mit einem Testwert speichert wirkt sich das in wenigen Sekunden auf die CPU Last aus. Optimaler Wert gefunden? Dann hier einsetzen.
-v		Kann Angabe		Wenn angegeben, werden umfangreiche Verarbeitungsmeldungen ausgegeben.
-w	Dateiname	Kann Angabe	Zeichen	WHITELIST // Angabe einer Datei wie „-d“. Datei enthält pro Zeile eine Frequenz, die immer in die Empfangsliste kommt, wenn sie nicht schon enthalten ist. Angabe in KHz z.B. 402700. Auf volle 10 KHz auf/abrunden

Beispiele:

./**scannerlist** -L -30 -H /tmp/holding.txt -o ~/dxlAPRS/sdrcfg.txt -f /tmp/scan.csv -v >> /tmp/scannerlist.log

./**scannerlist** -q 80 -n 5 -H /tmp/holding.txt -o ~/dxlAPRS/sdrcfg.txt -f /tmp/scan.csv -v >> /tmp/scannerlist.log

rtl_power -f 402M:404M:1000 -d0 -g 38 -p 56 /tmp/scan.csv -1 2>&1 > /dev/null

Empfohlen wird immer eine Schrittweite von 1000 Herz zu nehmen.

-d = Device-Nummer (Default 0), -g = Lautstärke / Gain. Empfohlen nicht mehr als 38 zu benutzen, da bei weiterer Verstärkung kein besserer Signal/Störabstand erreicht wird. Es steigt dann nur der Rauschpegel mit an. -p = der für jeden DVBT-Stick zu ermittelnde Frequenzkorrekturfaktor. Der ist nötig, damit eine eingestellte Frequenz wirklich auch empfangen wird. Der Wert ist Herstellungsbedingt und muss mit einem SDR-Empfänger und bekannten Signalen oder anderen Methoden ermittelt werden.

English – read here

The advantage?

For the tracking of weather probes in the frequency range 400-406 MHz, many people uses the software package from OE5DXL - Chris, a radio amateur from Austria. With the software package, you can receive the probes and analyze their data under Unix and Windows using a simple DVBT stick.

His GitHub: https://github.com/oe5hpm/dxlAPRS

ScannerList supports the program dxl-package. It scans a list of frequencies at the same time. This filling works even while the program is running. Means: you can correct the list at any time and it is used immediately without any program interruption.

Some have created scripts that use the "rtl_power" program to scan a range and convert the received values to frequencies when a usable decibel value is found. This evaluation assumes "ScannerList" with some amenities.

Why this?

Actually, you could do this without *ScannerList*. But it must have a meaning, that this program exists. So here are the advantages:

In fact, you fill in the config file “sdrcfg” inside the dxl software line by line all the frequencies you want the program to monitor. There are some frequencies which are always used reliably. But what about the many others? Especially the DFM models of the military transmits on different frequencies all the time. If you like to decode them all you have to shovels the list properly full. After this have a look at the CPU utilization. All data must be evaluated. This costs CPU time and keeps the room warm. It does not have to be.

With ScannerList only the really active frequencies are instructed for monitoring. The “sdrcfg” remains small and the CPU cold. If you want to be on the safe side, you can still put fixed frequencies into a list, they come over daily. The daily changing frequencies will be detected by ScannerList every 1 or 2 minutes per cron job. You don’t lose very much packets in a rising signal within this interverall.

To prevent *ScannerList* from a false alarm you can equip some really bad noise in a Blacklist.

How it works

For a scanner you need a dedicated DVBT stick which is exclusively provided.

Measured at intervals, not in continuous operation.

We need a script (example: scanner_prod.sh) which is called by CRONJOB every 2 minutes (I use here). It starts the "rtl_power" scan with a block from 402 to 404 MHz, then runs "ScannerList" and evaluates the csv file from rtl_power. Creates a frequency list conforming to the DXL package as a file. This is transferred by SCP (file transfer) to the respective computer. The same steps then run again for 404 to 406 MHz. The frequency list goes to another computer. Why only 2 MHz? Because the DVBT Sticks can only receive this range with a single slide.

The really interesting step is the evaluation of the decibel values of the rtl_power and to read from them probes, to distinguish from disturbances, which are less interesting.

About that signal recognition

short: how rtl_power scans? Have a look at the parameters. They not only specifies the frequency range, but also the interval between the ranges. I recommend 1,000 cycles (Herz) (known as 1 Khz).

Together with the usual data for DEVICE-NR, GAIN and FREQUENCY CORRECTION, you will get a file output in CSV-format (Excel). Each line starts with some parameter followed by decibel values for each step (1 KHz). These are given in negative numbers. When a signal increases, the minus-value increases up to 0.

short: how a probe signal looks like? If you use an SDR program and focus on a signal, you can see a bar with a range between 6 up to 12 KHz - depending on the probe type.

The scanner scans in 1 KHz steps. A good probe signal will cover several good signal values next to each other (i.e. here 6 to 12 pieces). This is exactly how the program tries to detect signals. Disturbances are usually individual rather narrow-band, mostly unmodulated carriers without information content. So while to small they fall out of the grid.

How do you recognize if a single measured value level is a signal and not only noise? I only got a negative value.

Good question. For this purpose, you must either know which values represent only noise otherwise when a useful signal above the noise is present. To automatically detect the ground noise level we calculate the average value over 100 steps and safe it as a noise level. Here is a look at the CSV file which has given us rtl_power. The CSV file can be loaded and viewed in EXCEL or similar programs. On most frequencies, only noise will be seen. The human eye will find this value very fast.

Next Question: which level is good for decoding by dxl-receiver?

To evaluate this, you need two receivers. One uses by the dxl program and one for an live sdr-radio to measure the signal noise difference. If you compare an upcoming signal on both machines you will see that a level up to 5 between 10 db over noise gets decoded by the dxl-Software.

ScannerList has to options to detect a good probe signal. First it will use fixed noise levels given by user with parameter –L (if you know that your noise level is about -35, use –L -30 to take all values between -30 up to 0 as possible probe signals. What about changing noise levels over a day period?

Therefore I invent the automatic mode you can activate by “-n “. The following positive value will be used as a relative value above noise. As example: noise level assumed as -35, you set –n 5, good signals get watched from -30 dB up 0. The noise level gets automatically detected every 100 measure points.

This is an advantage you will never miss. On a day period the noise level may change a bit. The automatic gets the average value of 100 values every time. So the relative value “-n” always flows based on it.

You could follow all this way down here? He he, congrats. All other readers: use the recommendation “-n 5” – it’s the easy way.

More attributes

To seperate all that values from noise it is only the first step in our work. While probe signals are wider then 4 KHz, we let the programm look for about minimum 4 good values following on each other. The next one is to evaluate the middle of the frequency. We need the width divide it by 2 and add this to the beginning of the first signal. Voila: a probe frequency is locked.

A good help in getting better frequencies is the knowledge of then the following: The international plan of useable frequencies is in a raster of 10 KHz. So a middle frequency of e.g. 402.698 should count up to the next full 10 KHz – this is why the program save it as 402.700. The dxl-receiver has an inplemented automatic frequency controller we set to +/- 5 KHz. So this probe on 402.698 will be catched without problems.

Losing a signal for a while – no problem

Sometimes a signal may be week for some minutes, but come back after the drop out. To compensate this the program uses a hold list for each signal with a last-head-timestamp. After a declared timeout value „-h“ the frequency will be dropped out the list. Up to this time it will be hold always it is not be decodable. This value is set to 10 minutes. Good values are between 5 to 15 minutes.

Copy the frequency list to other computers using SCP

Every one of you uses a scanner on a separate computer, wants to know how the *dxlsdr.cfg* file gets on this other machine. Use „SCP“. This copy tool is integrated on almost every Unix computer system. It helps you copying files across the network. There is a small wall to climb over. If you try this tool manually you will find out that some questions have to get answered until the file is over there. Not good for an automatic transfer job, isn´t it? To solve this, we will use a certificate.

We going to make this one-time setup on the source machine (scanner):

- ssh-keygen (no password please)
- ssh-copy-id user@destinationcomputer
  user is something like „pi“
  as destinationcomputer: use the ip-adress
  „ssh-copy-id“ drops a copy of the fresh certificate on the destination machine .
- scp never ask again for access privilages – until now the computers trust each other (oh, do not use SUDO, if so, SUDO assumes user is root, so the certificate will be not found, we created this for user „pi“)
  scp /path/filename user@destcomputer:/path/{destfilename]}

Commandline parameters for *ScannerList*

Key name	Key value	default	unit	remarks
-a	1…10	5	KHz	Automatic frequency control. Catching probe signals below or upper 5 KHz.
-b	5000…15000	automatic	Herz	Bandwith. If -b is omitted, the bandwith will be automatically calculated. If -b is given, all probes are received with this value. Recommended: omit for use automatic or

				use „-b 8000“. M10 probes are better placed in whiteliste file with fixed width 17
-d	file name	No must	chars	BLACKLIST // Some unwanted frequencies may be listed within this file name. They will be dropped out in the destination list. One frequency per line. Unit is KHz e.g. 400020. Round up/down to next full 10 KHz
-f	file name	Must have	chars	This is the full qualified file name of the output file of „rtp_power“. We need the CSV (Excel) format as input
-h	5…15	10	minutes	This is the timer in minutes to hold active frequencies even if they are out for a while
-H	file name	Must have	chars	This is the full qualified work file name to handle the hold timers (see above). Do not touch this file.
-L	-50…-15	-20	Decibel db	For **manual signal detection**. Received values with a better signal value will be used regardless of how big the noise is. Remember that a noise value is around -30 to -40 db. So -20 db is a better value.
-n	3…20 recommended: 5	No must	Decibel db	For **automatic signal detection** considering chancing noise levels. If -n is given, -L will be ignored. For automatic noise detection, we need an extra checkup. Every segment on 100 KHz the average noise level will be calculated and recorded. So if -n is set to 5 this means, that every value with 5 db above the noise level, will be marked. e.g. noise = -35 db, signals are good above a value of -30 db.
-o	file name	Must have	chars	This is a full qualified file name for the final frequency list output needed by dxlAPRS-Chain as input where to listen. The file format is exactly the one this program expects. e.g. “F 405.700 5 0 0 8000” 5 = value from key „-a” / 8000 = Bandwidth from automatic or key „-b“
-q	0...99 recomm: 80 +/- 10	0	percent	Squelch-level, uses as param in sdrcfg.txt in each frequency line. If default 0, you see high cpu usage on sdrtst. Squelch slightly closed = 90 downto 10 = completly closed. You have to try the best value. Save some values in sdrcfg.txt and save while sdrtst is running. Watch cpu usage. If not going down check next reduced by 10 until usage gets lower. Optimal value depends on hardware. If found a good value, add it here.
-v		No must		Verbose (anything you need to follow the process can be redirected to a log file)

-w	file name	No must	chars	WHITELIST // Like key „-d“. You can give a file name here which contains frequencies on each line. They will be always added to the output list for scanning, even if there is no signal yet. Give values in KHz e.g. 402700. Round up/down to next full 10 KHz

Some samples:

./**scannerlist** -L -30 -H /tmp/holding.txt -o ~/dxlAPRS/sdrcfg.txt -f /tmp/scan.csv -v >> /tmp/scannerlist.log

./**scannerlist** -n 5 -H /tmp/holding.txt -o ~/dxlAPRS/sdrcfg.txt -f /tmp/scan.csv -v >> /tmp/scannerlist.log

rtl_power -f 402M:404M:1000 -d0 -g 38 -p 56 /tmp/scan.csv -1 2>&1 > /dev/null

Always a step value of 1000 Herz is recommended.

-d = Device number (Default 0), -g = Gain. It is not recommended to exceed the value above 38/40 units or to run it on automatic. Otherwise the noise level increases without any more advantage.
-p = Please check the correction factor of each individual stick first before use it. It depends on the production process and is a value in KHz. There are some recommended procedures to check this published in the internet. I recommend to use a SDR receiver program like sdr# for a better precision.

So go on and have fun

If Questions, ask me on df7pn@darc.de

History

---------- Historie Deutsch ----------

Kopieren nach dxlAPRS/src
übersetzen mit:
gcc -o scannerlist scannerlist.c -lm

Parameter: Keine Änderung

version 1.01 / 04.01.2018 / folgende internen Änderungen:
- Abstände beim Scann zwischen Signalen von 3 auf 4 KHz erhöht bevor angenommen wird das sei ein neues Signal (M10)
- Bandbreite bei automatischer Erkennung wird nicht mehr um 3 KHz reduziert wenn > 9 KHz.
- NEU: Kürzung der Bandbreite auf 30 KHz nur dann, wenn breiter als 30 KHz
- Whiteliste: Bandbreite wird in die Ausgabe mit übernommen (überschreibt auch automatisch erkannte Breite)
- Signalbreite > 12 KHz, dann wird AFC auf 0 gesetzt.

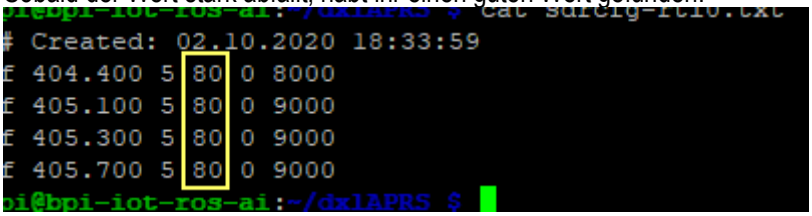
23.11.2018

Version 1.2

version 1.02 / 02.10.2020 /

Neuer Parameter:
„-q nnn“ Squelchwert für die sdrcfg.txt (Frequenzliste)

Standard: 0 (Parameter nicht gesetzt)
Squelch leicht geschlossen: 90
In Stufen immer stärker geschlossen, wenn Wert reduziert wird
Empfohlen: 90/80 durch probieren ist das Optimum zu finden.
Test: sdrtst sollte laufen und eine sdrcfg.txt mit einigen Einträgen 3-4 enthalten.
CPU Nutzung von sdrtst (Command: top) liegt im weit zweistelligen Bereich
Im Editor die sdrcfg.txt mit verschiedenen Squelchwerten speichern und immer wieder die CPU Nutzung prüfen.
Sobald der Wert stark abfällt, habt ihr einen guten Wert gefunden.



---------- history english ----------

copy to dxlAPRS/src
compile with:
gcc -o scannerlist scannerlist.c -lm

Params: no change

version 1.01 / 04.01.2018 / the following internal changes:

- Increased scanning distance between signals from 3 to 4 KHz before assuming this is a new signal (M10)

- Bandwidth at automatic detection is no longer reduced by 3 KHz if > 9 KHz.

- NEW: Reduction of bandwidth to 30 KHz only if wider than 30 KHz.

- Whitelist: Bandwidth is included in the output (also overwrites automatically detected width)

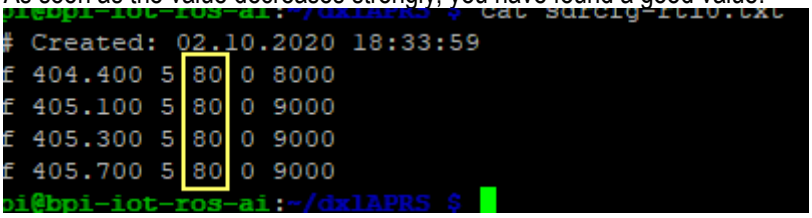
- Signal width > 12 KHz, then AFC is set to 0.

Version 1.2

version 1.2 / 02.10.2020 /

New parameter:
"-q nnn" squelch value for the sdrcfg.txt (frequency list)

Default: 0 (parameter not set)
Squelch slightly closed: 90
Closed more and more in steps of 10 counters
Recommended: 90/80 by trying to find the optimum.
Test: sdrtst should run and a sdrcfg.txt contains some entries 3-4.
CPU usage of sdrtst (Command: top) is in the two digit range
In the editor save the sdrcfg.txt with different squelch values and check the CPU usage again and again.
As soon as the value decreases strongly, you have found a good value.



---------- end of file ----------