Blick unters Federkleid des KiwiSDR

05.11.2017 08:07

Erschienen auf https://www.hamspirit.de/8654/kiwisdr/

2016 haben John Seamons, ZL/KF6VO, Michael Jones und Jonathan Piat auf Kickstarter 70757 $ zusammengetragen, um einen Multi-User Web SDR Empfänger für 0-30 MHz zur Produktionsreife zu bringen. Gleich an dieser Stelle ist zu erwähnen, dass dieser wohl ohne die großartige Arbeit von András Retzlergar nicht möglich gewesen wäre. So lässt die Randbemerkung in einem seiner spannenden Blogeinträge vermuten, dass es offenbar doch die ein oder andere Diskussion gab, inwieweit eine GPL Lizenz Chancen und Risiken birgt.

Namensgeber des Kiwi ist offenbar ein exotischer Vogel, auch bekannt als Schnepfenstrauß. Einen direkten Zusammenhang konnten wir nicht herausfinden, aber

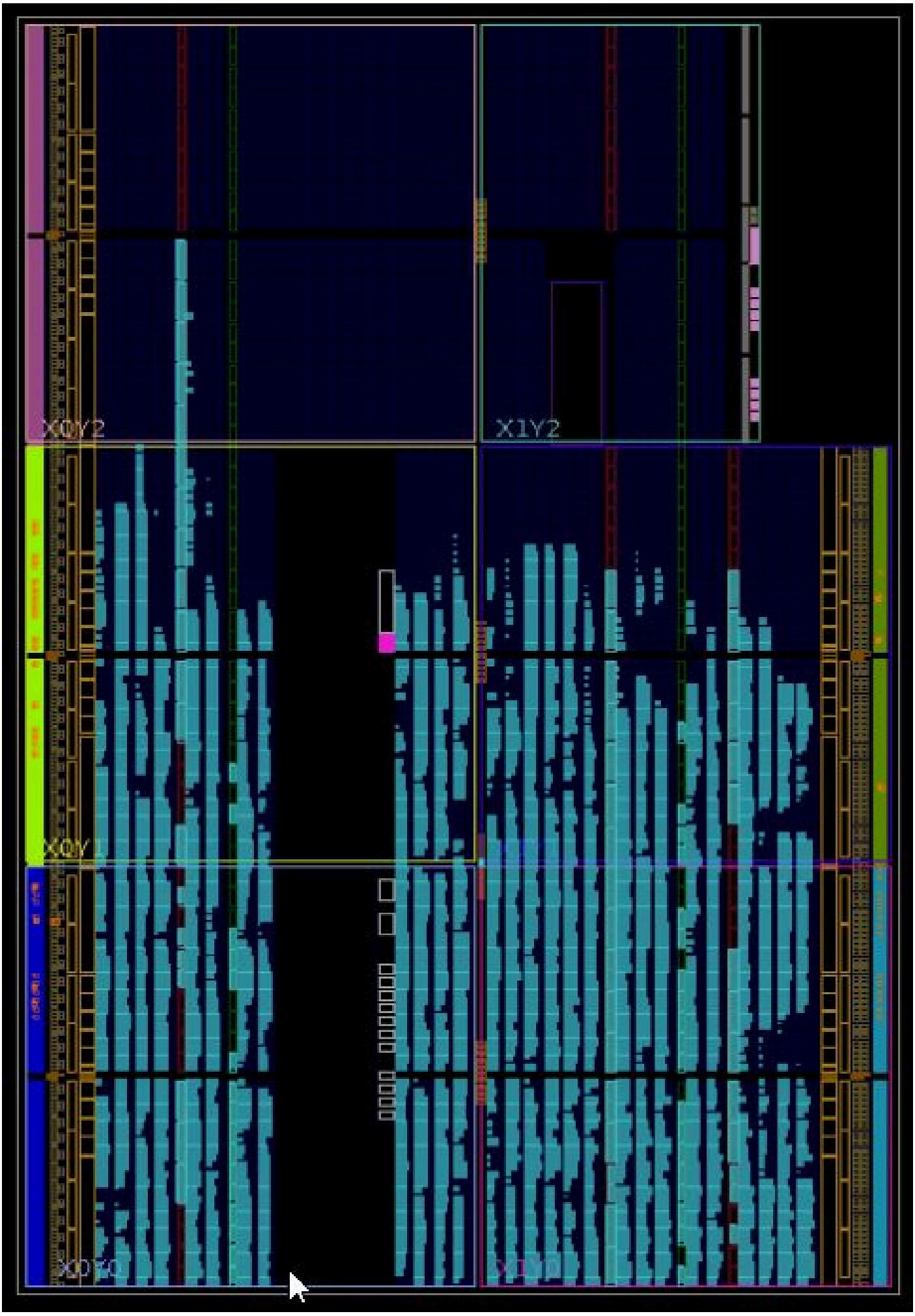
das Präfix ‘ZL/' von John's Call lässt einen Bezug zu seiner Wahlheimat und dessen Nationaltier vermuten.

Dank Seeedstudiodurften wir nun eben diesem exotischen Vogel gründlich unter das Federkleid schauen. Dies soll kein weiteres Review der Empfangsqualität werden, zu empfehlen ist das Review von fenu-radio.ch. Die technische Architektur liegt vollständig offen. Das “Design review document” lässt keine Fragen offen, dass es sich um ein gelungenes aber auch kostenbewusstes FPGA Design handelt, was wir im Verlauf der ersten Tests auch so bestätigen konnten. Betrachten wir also zunächst die wesentlichen Elemente des Systementwurfs:

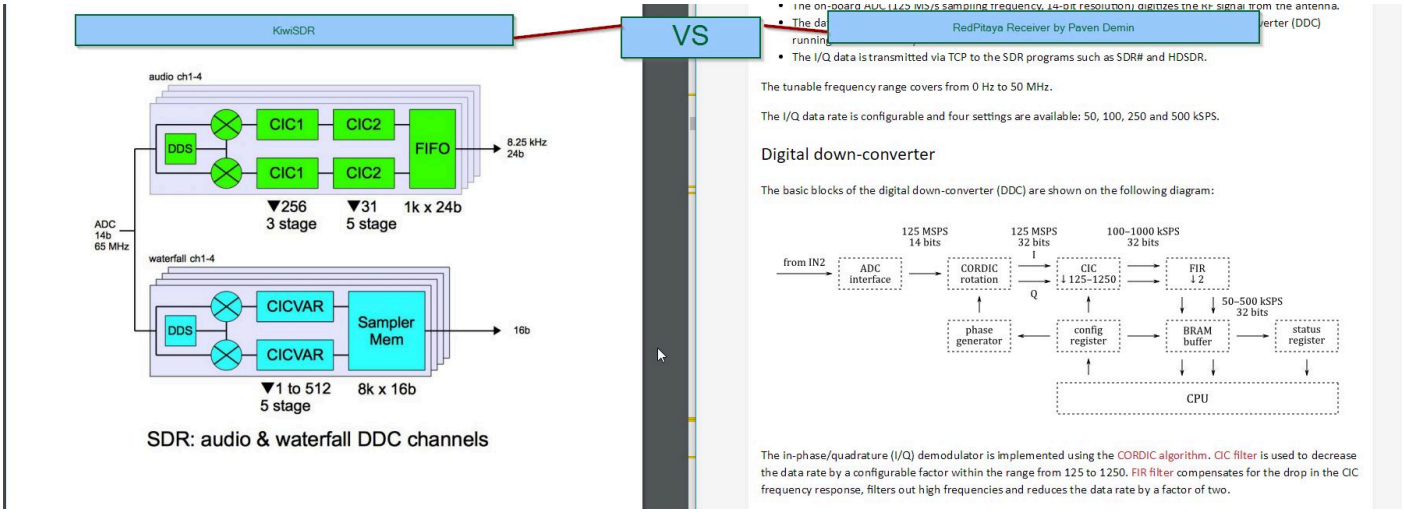
Hinter dem Frontend arbeitet ein LTC2248mit 14-bit Auflösung, welcher die analoge Welt mit 65 MHz abtastet. Hiermit ist nachNyquist-Shannon-Abtasttheorem unser Empfangsbereich dann auch von 0-30 MHz gegeben. Leider hat man im Frontend-Design aus Kostengründen auf ein Software-seitig schaltbares Preamp verzichtet, so ist dem ADC fix ein LTC 6401-20vorgeschalten. Gerade beim Frontend-Design hätte ich ehrlich gesagt ein wenig mehr erwartet und auch mit kostengünstigen Bauteilen hätten sich noch einige Filter/PreAmp Goodies verbauen lassen.

Die Clock wird von einem Conner-Winfield CWX823 series XOauf Takt gehalten. Dieser ADC-Takt wird in der FPGA Software nochmal GPS-gestützt korrigiert, was ein sehr stabiles Design verspricht. Das ist besonders auch dann spannend, wenn man den Kiwi in Umgebungen mit variierenden Temperaturen betreibt. Als GPS Frontend wurde eine – erneut kostengünstige – SingleChip Variante verbaut, SE4150L, welche aber in unserem Test für zuverlässige GPS Signale sorgt.

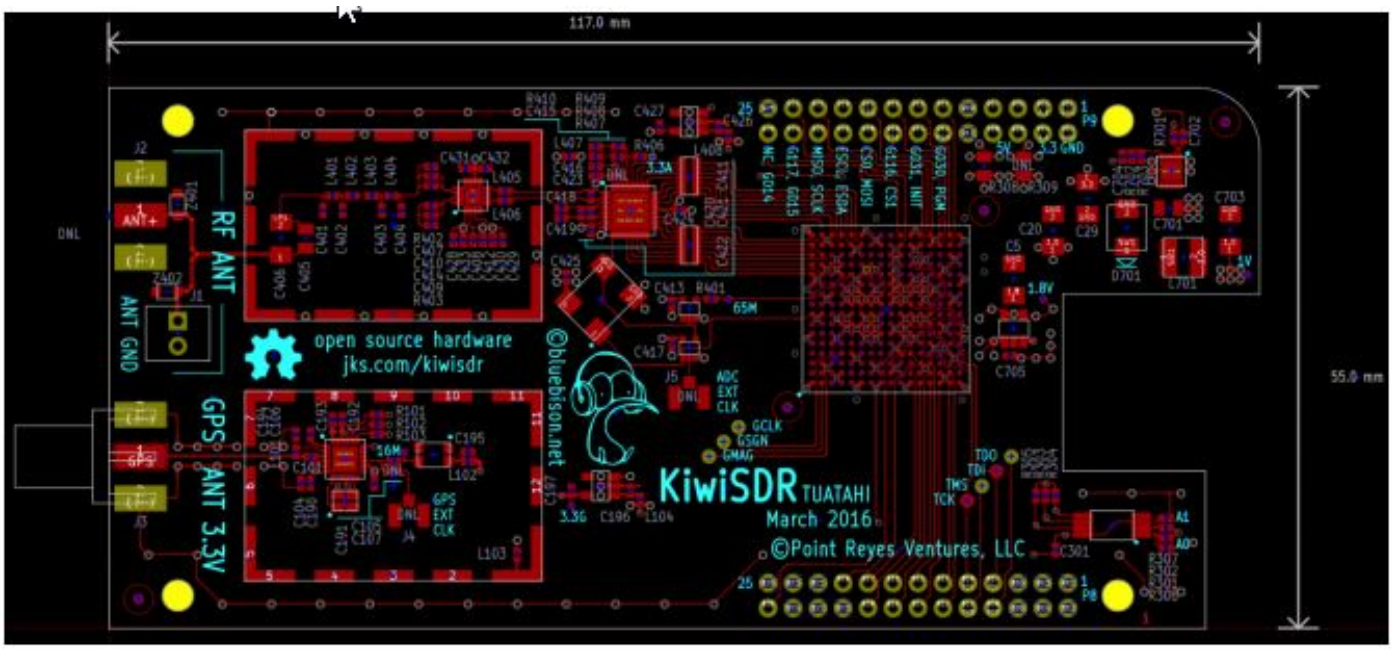
Der DDC Receiver ist vollständig in FPGA Logik implementiert. Laut Design Review Dokument ist zwar nicht mehr sehr viel Platz auf dem FPGA, allerdings noch ein wenig Luft nach oben vorhanden. Dies ist auch insofern relevant, da somit im Falle von Upgrades oder auch Bugfixes noch Kapazität vorhanden bleibt. So sind z.B: nur 40% der verfügbaren DSP Ressourcen erschöpft. (Abbildung Design Review Dokument, Seite 12 – Verbrauch FPGA Ressourcen):


X0Y2
X1Y2
X0Y1
X0Y0
X1Y0

Anders als z.B. der RedPitayasetzt der KiwiSDR nicht auf SOC Hardware, sondern auf einen vergleichsweise günstigeren Xilinx Artix-7 A35 FPGA. Im RedPitaya ist ein Zynq-7000* SoCverbaut, welcher CPU/Memory/Peripherie & FPGA in einem Chip vereint. Pavel Demin hat mit seinen RedPitaya Projekten bereits eindrücklich aufgezeigt, wie effizient sich FPGA-SoC-Designs für Anwendungen im Amateurfunk einsetzen lassen. Vergleichen wir z.B.: Die Implementierung des FPGA basierten Empfängers, so warten keine großen Überraschungen auf. Direkt nach dem ADC erfolgt die IQ Mischung nach CORDIC und durchläuft die CICFilterstufen. Hier punktet das Design des KiwiSDR mit seinem stabilen GPS gestützten 65 MHz Takt.

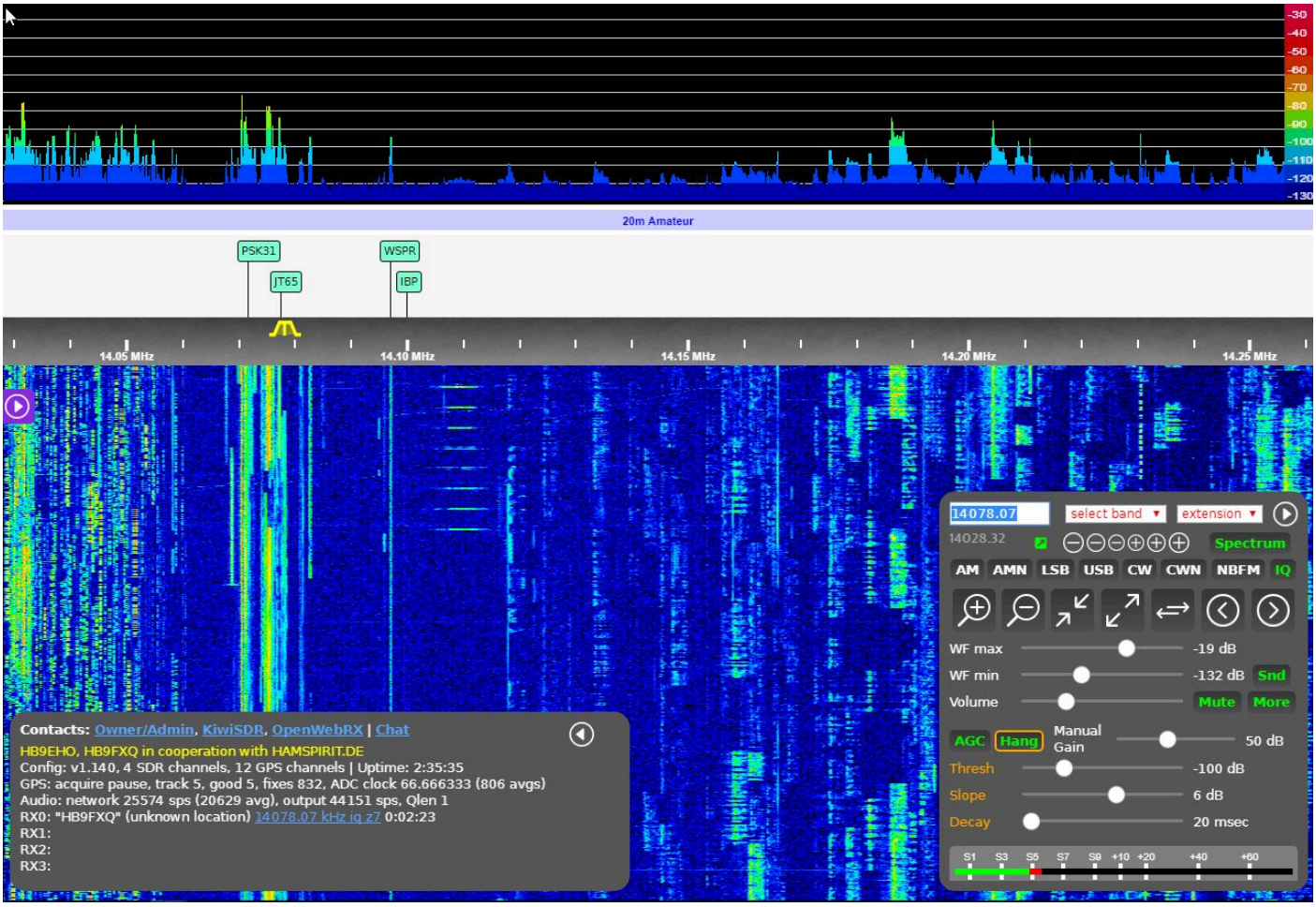


Beim Kiwi fällt auf, dass Wasserfall und Audio hier direkt getrennt implementiert wurden. So ergeben sich pro Client dann auch zwei DDCDatenstreams vom FPGA, welche zum CPU transportiert werden müssen. Da es sich wie gesagt um keinen SoC Chip handelt, musste ein CPU her – naja, und eben alles weitere, was zu einem Ethernet angebundenen PC gehört. Hier hat man sich für den Einsatz des BeagleBone entschieden. Ein kompakter Einplatinencomputer im Stil des RaspberryPi. Aktuell wird der “BeagleBone Green” verwendet, welcher im Vergleich zum etwas teureren “Beaglebone Black” über keinen HDMI Ausgang verfügt, der ohnehin keine Anwendung findet. Der KiwiSDR wird direkt auf den BeagleBone aufgesteckt. Hierzu das Schema aus dem Design Review Dokument:



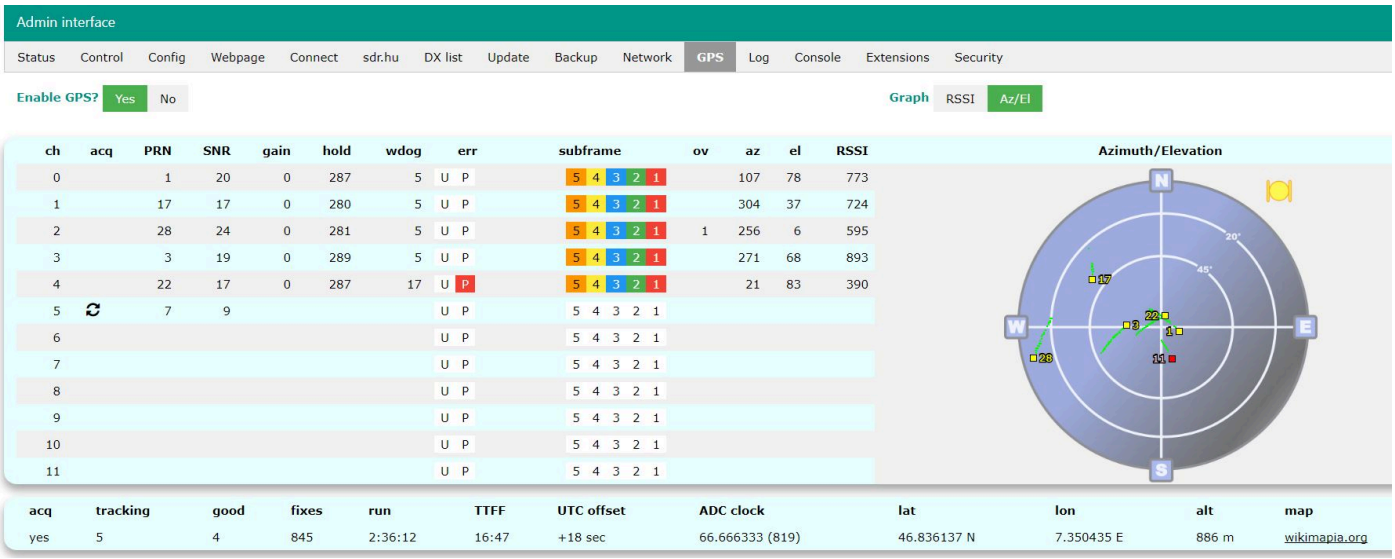
Beim Datentransport vom FPGA zur CPU profitiert man in SoC Architekturen vom kurzen Weg des DMA/Speicherdirektzugriff – dies ist im Kiwi Design nicht möglich. Die Daten werden via 32-bit/48 MHz SPIin den Beagle gefüttert. Hier entsteht ein Flaschenhals, der die Limitation der 4 unabhängig arbeitenden Clients (8xDDC Datenstrom) begründet. Auf dem Beagle selbst läuft ein herkömmlicher Linux Kernel, über dessen /dev/spidev die Daten dann in der Software ankommen. Das Linux Image für den Beagle wird pfannenfertig zur Verfügung gestellt.

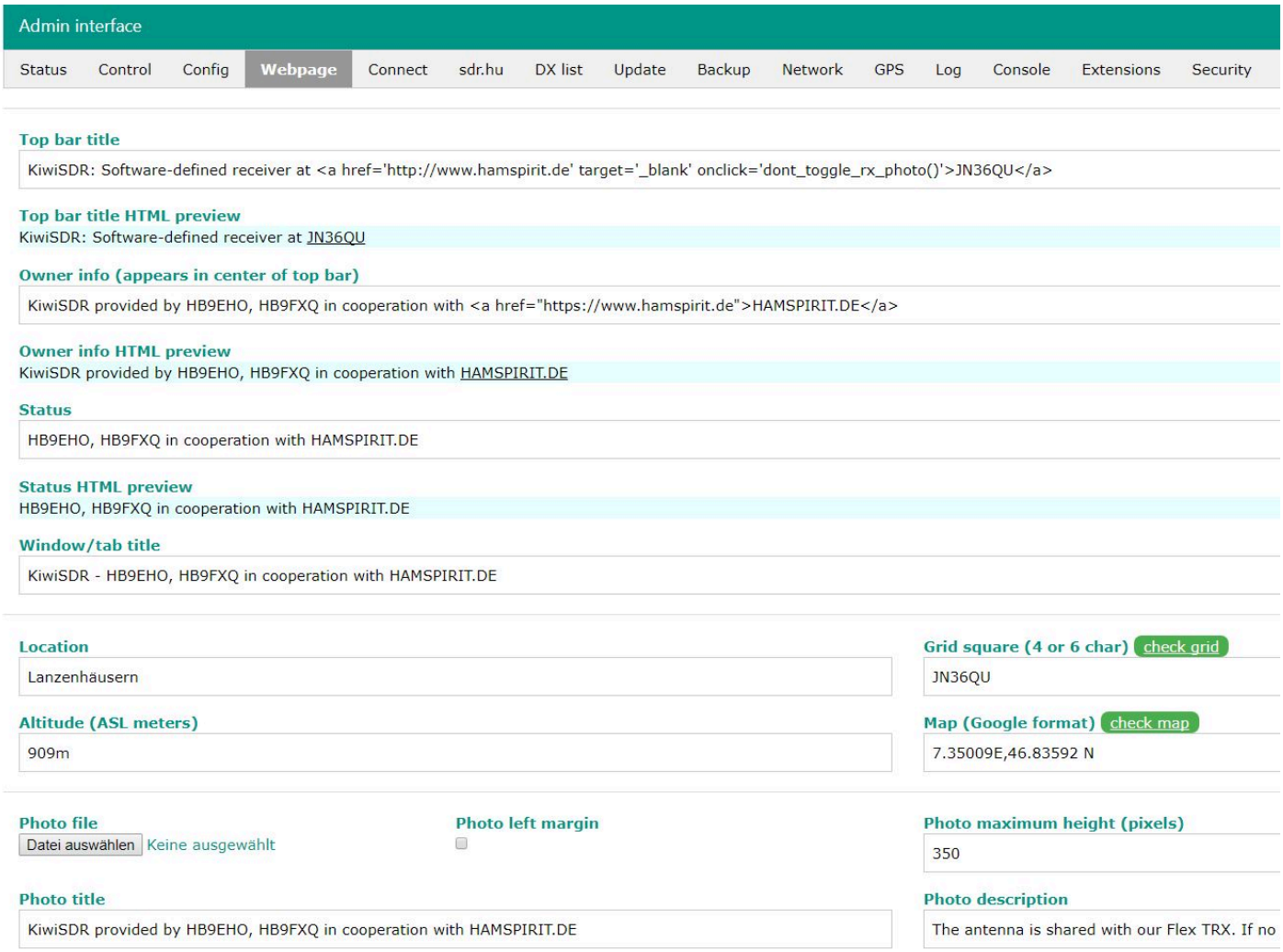
Betrachten wir also die Software. Diese kann aus Anwendersicht als durchaus sehr gelungen beschrieben werden, da hier die Stärke von OpenWebRX voll zum tragen kommt.



Die Inbetriebnahme stellt keine grosse Herausforderung dar. Ein einfaches PortForward am Router zum WebFrontend genügen, um den Kiwi ans Internet zu bringen. Für Netzwerke, in denen z.B. eine strikte Firewall kein Port Forward erlaubt oder man an Carrier-grade NAT durch seinen Provider leidet, steht ein ProxyService zur Verfügung. Die Admins werden es einem nicht danken :-).

Für die Administration selbst werden dem Kiwi Betreiber keine Linuxkenntnisse abverlangt. Eine einfache Weboberfläche ermöglicht die Konfiguration und Überwachung des Kiwi's im Browser.





Der Go-Live unseres KiwiSDR hat sich dann allerdings um einige Zeit verzögert, da man bei der Software offenbar mehr auf schnelle Time-To-Market als auf Sicherheit bedacht war. Genau diese Haltung der Hersteller verschafft solchen kleinen internetfähigen Geräten auch den Beinamen #InternetOfTarget Device.

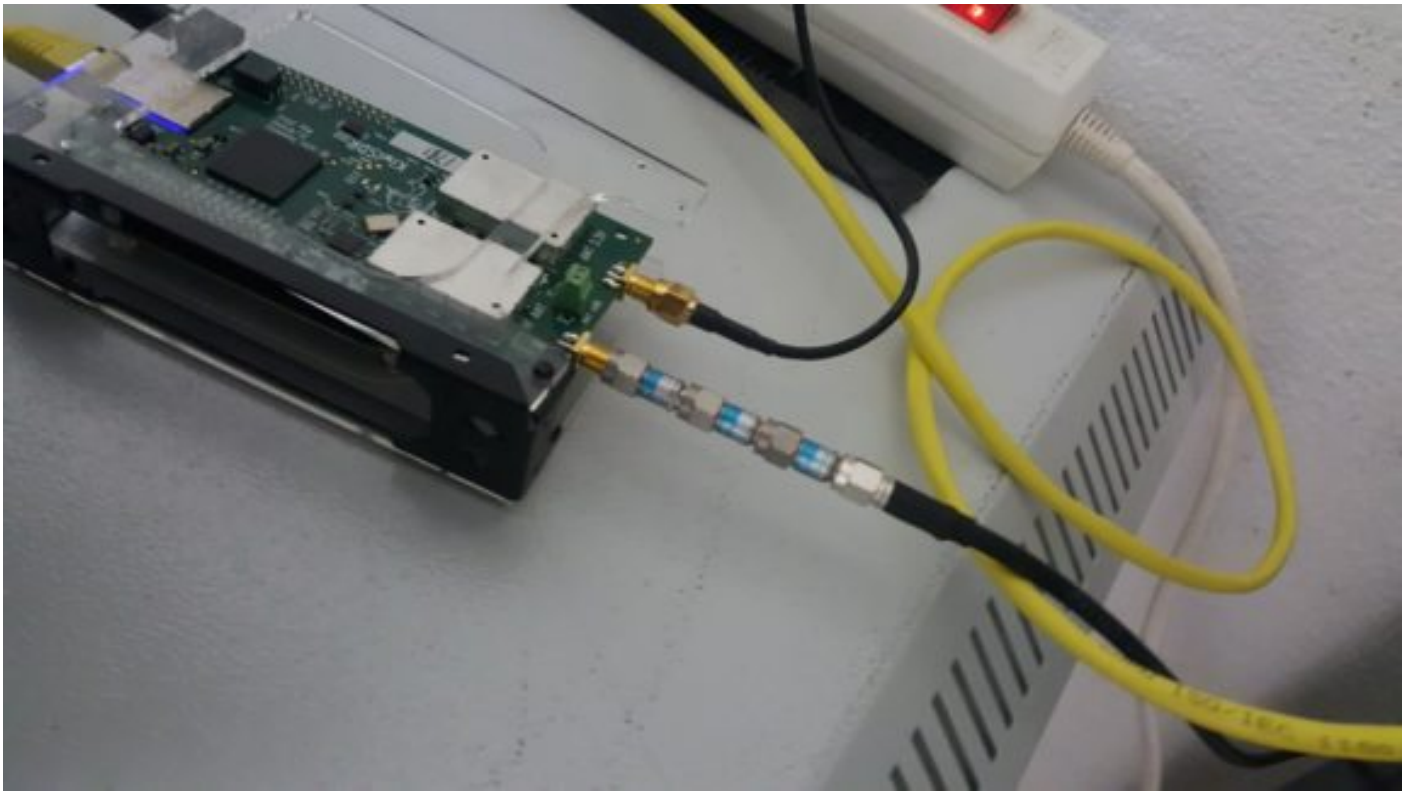
So habe ich zunächst auf einige Missstände im Bereich der Sicherheit der Software hingewiesen, bevor der Empfänger dann ans Netz durfte. Diese wurden von John auch sehr zeitnah berücksichtigt und bereinigt. Allerdings nur die notwendigsten

Punkte. Es verbleiben aktuell u.A. noch Fragen z.B. zu einer Art Backdoor im SSH und den Klartext Eingabefeldern etc….

SSL/TLS suchten wir vergeblich, auch der einfache Betrieb hinter einem HTTPS-Forward Proxy ist vorerst nicht möglich, wofür es dann doch große Abzüge in der A-Note gibt. Betreiber eines KiwiSDR sind aus meiner Sicht aktuell gut beraten, ihre Firewall gut zu konfigurieren und den Kiwi auch im internen Netz stark in seinen Möglichkeiten einschränken und zu isolieren. Solche Sicherheitslücken bieten ja nicht nur Angriffsfläche auf den KiwiSDR selbst – was zu verkraften wäre – sondern stellen auch ein Einfallstor ins eigene Netz dar. Wir werden auch weiterhin versuchen die Entwickler auf Schwachstellen hinzuweisen und die Situation ein wenig kritisch zu beobachten.

Nun durfte der Kiwi ja letztlich dann doch online gehen, und zwar an einem ganz besonderen Standort. HB9EHO/ HB9FXQbetreiben gemeinsam eine Remotestation in JN36QU. In der Zeit, in der sie ihren Flex-6500 TRX nicht verwenden, klemmt der KiwiSDR von HAMSPIRIT.de nun an einer OptiBeam OB15-7. Ihr findet den Receiver auf SDR.HU – wenn auch euch der Empfang überzeugt, so gebt ihm doch dort bitte gerne auch euer Vote. Wenn kein Signal anliegt nicht wundern, es handelt sich meist nicht um lange Zeit.

Da der ADC mit der starken Yagi-Antenne doch sehr übersteuert, haben wir aktuell 9 dB Dämpfung zwischen Kiwi und Antenne verbaut:



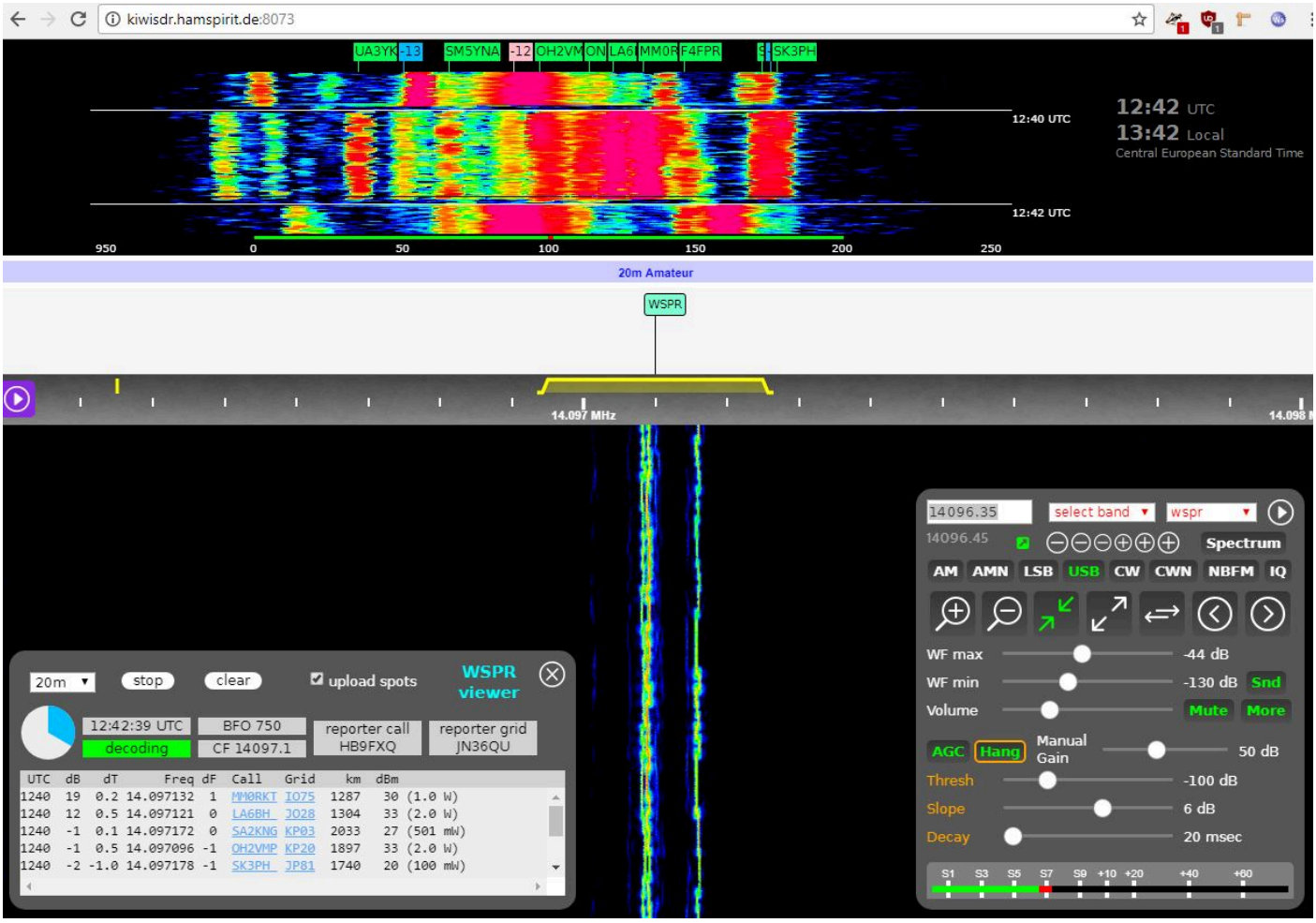
Zum Abschluss noch einige Impressionen:

OptiBeam in Richtung Norden, der Heimposition des Prosistel PST-61D Rotors. Dank ländlicher Gegend sehr QRM arm, allerdings suchen wir aktuell nach einer periodisch breitbandig auftretenden Störung im Bereich um 20m:

Teile der Infrastruktur der Remotestation HB9EHO/HB9FXQ:

Außenmontage der GPS-Antenne:

WSPR direkt im Browser mit der KiwiSDR Extension:



In dieser rauschfreien Region steht der KiwiSDR.