Software

Die Software für das Projekt wird bei github entwickelt.

Signalverarbeitung

Da die Erzeugung des IQ-Basisbandsignals direkt im Raspberry Pi passieren sollte, mussten einige Module zur Signalverarbeitung geschrieben werden. Diese umfassen die APRS- und SSTV-Erzeugung, eine Sprachansage und die Frequenzmodulation.

Bestandteile

Die Software besteht aus kleinen Modulen, die zentral von einer Ablaufsteuerung koordiniert werden.

- Ablaufsteuerung (Aufnehmen und Abspeichern von Bildern, Erzeugung SSTV und APRS, Aussendung) - Python
- LO-Steuerung (Trägerfrequenzerzeugung durch Einstellung des SI570) - C
- APRS-Erzeugung (Mono-NF) - Python
- Robot36-Erzeugung (Mono-NF) - C
- FM-IQ-Modulation (Mono-NF zu Stereo-IQ) - C
- Sprachsynthese (Aneinanderreihen von Zahlen-WAVs mit Pause) - Python / C
- Audioplayer für Stereo-IQ-Daten (aplay) - builtin
- Resampling von Audiodateien (resample) - builtin
- Konvertieren der Webcambilder zu 320×240 für Robot36 (convert) - builtin
- gpsd für Aufnahme von GPS-Daten

IQ-Modulation

- mono-Audiodatei frequenzmodulieren (Frequenzhub und Mittenfrequenz einstellbar)
- stereo-Audiodatei ausgeben

LOCTL / LOCTL570

- GPCLK0 bzw Si570 auf frei wählbare Trägerfrequenz einstellen

APRS

- übergebene Position (Lat, Lon, Höhe, Temperatur) in WAV schreiben
- sollte aus Performancegründen in C reimplementiert werden, stellte sich aber als nicht nötig heraus.

SSTV

- übergebene Bilddatei in Robot-36 kodierte WAV wandeln

Ablaufsteuerung

Parameter T bestimmt die Missions-Steigzeit. Er muss global veränderbar sein.

Im Vorbereitungsbetrieb:

- Start der Software
- Status-LED an
- Warten auf GPS-Fix
- Status-LED blinken
- 1 Minute warten
- Missionsstart (SSTV-Aussendung beginnt)

Im Missionsbetrieb:

- PA an
- Beginn SSTV-Aussendung auf 145.200 (36 Sek), währenddessen:
  - neues Webcambild speichern
  - SSTV, APRS und Ansage für nächste Aussendungen erzeugen
- Aussendung APRS auf 144.800 (3 Sek)
- Aussendung Ansage auf 145.200 MHz (5 Sek)
- PA aus
- Pause, bis 1min vorbei ist
- nach T wird die Nutzlast abgesprengt (siehe Absprengung)
- nach 2*T wird nur noch aller 5 Minuten APRS gesendet, kein SSTV weiter
- nach 3*T wird der Raspberry Pi heruntergefahren

Absprengung:

- Aussendung Signalton
- Mikrotaster abfragen
  - wenn geschlossen: Heizung an bis er offen ist
  - wenn offen: Heizung für 5 Sekunden an

Stromverbrauch nach Herunterfahren

- Muss man den Raspberry Pi von Spannung trennen, oder braucht er nach Herunterfahren nur noch wenig Strom?
- Raspberry + IQ-Mixer + Soundkarte
  - Pon = 10V*0,23A = 2,3W
  - Poff = 10V*0,09A = 0,9W
  - Verringerung der aufgenommenen Leistung auf unter die Hälfte

Das Problem wurde schlussendlich nicht in Software, sondern durch die Kopfschusselektronik gelöst. Damit wird der Raspberry Pi hart vom Akku getrennt, was die Lebensdauer des Peilsenders enorm vergrößert.

Rescue-Modus

Der Raspberry Pi muss bei einem unerwarteten Reboot nach erfolgtem Missionsstart sofort ein Ausklinken auslösen!

Vorbereitung Temperatursensor

- apt-get install i2c-tools lmsensors
- modprobe i2c-dev echo i2c-dev » /etc/modules
- i2cdetect -y 1 –> Anzeige des LM75
- echo lm75 0x48 > /sys/class/i2c-adapter/i2c-1/new_device

From:
https://loetlabor-jena.de/ - **Lötlabor Jena**

Permanent link:
https://loetlabor-jena.de/doku.php?id=projekte:xplorer:software&rev=1399810661

Last update: **2014/05/11 12:17**