Software

Bestandteile

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

- Ablaufsteuerung
- PiFM (Python-Wrapper)
- APRS (Python-Wrapper)
- Robot36 (Python-Wrapper)

IQ-Modulation

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

PIQ

- GP0CLK auf Trägerfrequenz einstellen
- Stereo-WAV (beinhaltet IQ-Daten) über Soundkarte abspielen
- GP0CLK abschalten

APRS

- übergebene Position (Lat, Lon, Höhe, Temperatur) in WAV schreiben

SSTV

- übergebene Bilddatei in Robot-36 kodierte WAV wandeln

Ablaufsteuerung

Im Vorbereitungsbetrieb:

- Start der Software
- Warten auf GPS Fix
- LED an
- Testaussendungen (APRS, Pause, SSTV, Pause, APRS)
- LED blinken
- 1 Minute warten
- Missionsstart (LED aus)

Im Missionsbetrieb:

- Speichern aktuelle Positionsinformation
- Aussendung APRS auf 144.800 MHz (2 Sek)
- 1 Bild speichern
- Ansage 3xLAT, 3xLON auf 145.200 MHz (10 Sekunden)
- Aussendung SSTV auf 145.200 MHz (36 Sekunden)
- Pause (10 Sekunden)
- Aussendung APRS (3 Sek)
- Pause 20 Sekunden
- nach X Minuten (je nach Version) wird die Nutzlast abgesprengt
- nach X + TBD Minuten wird der Raspberry Pi heruntergefahren und/oder abgeschaltet

Absprengung:

- Mikrotaster abfragen
- wenn geschlossen: Heizung an bis er offen ist
- wenn offen: Heizung für ?? Sekunden an
- Aussenden Signalton!

offene Fragen

- Muss man den Raspberry Pi von Spannung trennen, oder braucht er nach Herunterfahren nur noch wenig Strom? –> scheinbar ist es so, dass er nach dem runterfahren noch mehr strom braucht - also besser anlassen.

Aufsetzen des Produktivsystems

Installation des Grundsystems

- http://distribution.hexxeh.net/raspbian/raspbian-r3.zip herunterladen und per dd auf SD-Karte schreiben
- Raspberry Pi starten lassen, IP per DHCP zuweisen → pi:raspberry
- root-Passwort ändern
- rm /etc/ssh/ssh_host_* && dpkg-reconfigure openssh-server
- apt-get update
- apt-get install ntp fake-hwclock
- apt-get install tasksel
- tasksel install standard
- apt-get update && apt-get dist-upgrade
- rpi-update

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

> Was spricht gegen ein eigenes Kernelimage/Linuximage(runterladen, dd, fertig)? Dann hat man immer alles zusammen und muss nicht immer von vorn alles selber bauen. Ich werde wohl eh ein Github Repo aufmachen für mein StagePi, da könnte man auch XplorerPi als Branch vom offiziellen Zweig pflegen? — *Sebastian Weiß 2014/02/02 18:42*
>
> > erstmal muss es reproduzierbar erzeugt sein, dann wird eh ein image draus gemacht. am kernel gibts keine veränderungen, daher wird hier der ganze „userspacekram“ ausgehend von einer vanilla installation dokumentiert

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

Permanent link:
https://www.loetlabor-jena.de/doku.php?id=projekte:xplorer:software&rev=1391544412

Last update: **2014/02/04 20:06**