

FreeDV Treffen Mitte Juli 2022

1. kurzes Willkommen, Status letzte Funkerfahrten 40m & 80m (10 Minuten)
2. **Fragen** und Antworten zu FreeDV (max. 20 Minuten),
nächster Termin

3. **FreeDV Entwicklung**

SW Struktur: Bauprozedur, github Archive, Änderungen compilieren
wxWidgets, multi-threading, Fenster-Aufbau

3. Terminplanung

- FreeDV Runde **jeden Sonntag**
ab 11:00 Uhr, 40m & 80m
ab 13:30 Uhr, 40m & 80m
- für unsere Gruppentreffen ist der typische Wochentag **DONNERSTAG**

Juli							
kw	Mo	Di	Mi	Do	Fr	Sa	So
26					1	2	3
27	4	5	6	7	8	9	10
28	11	12	13	14	15	16	17
29	18	19	20	21	22	23	24
30	25	26	27	28	29	30	31

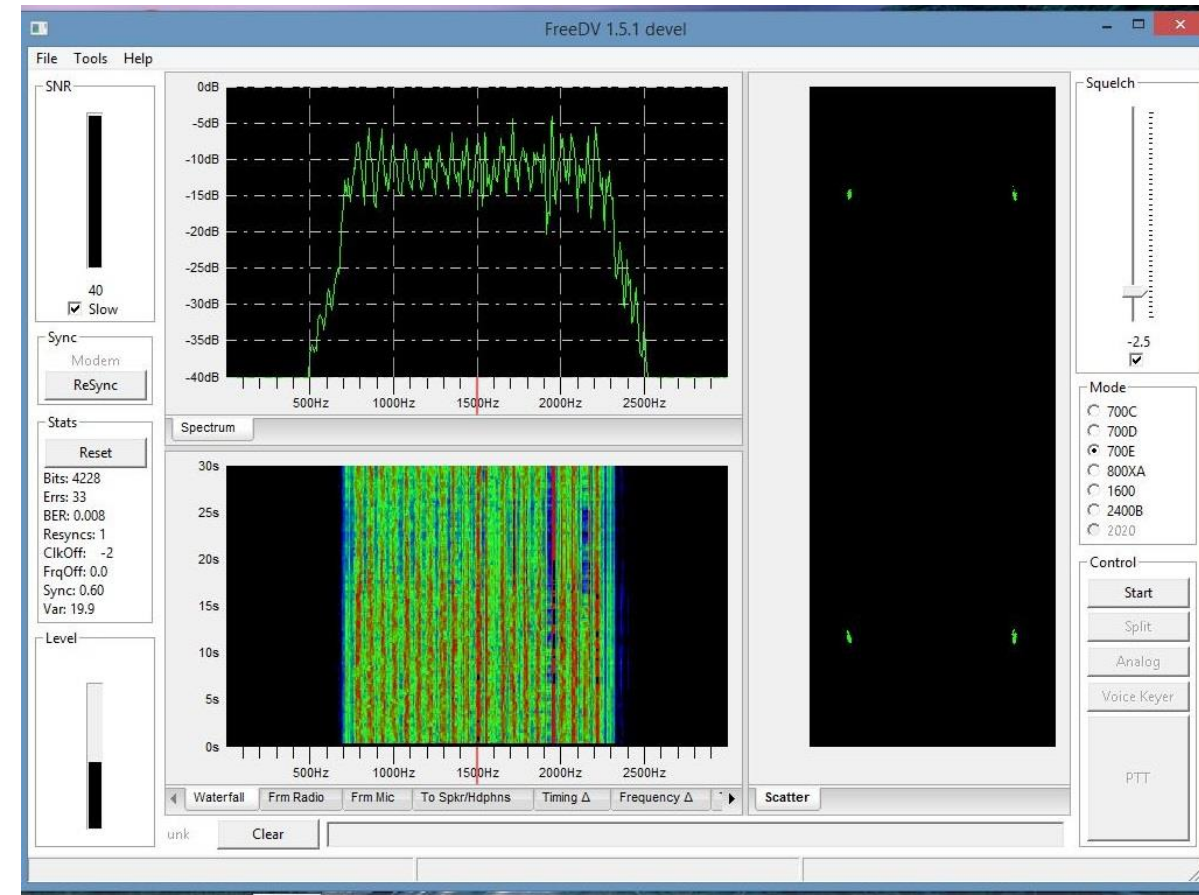
August							
kw	Mo	Di	Mi	Do	Fr	Sa	So
31	1	2	3	4	5	6	7
32	8	9	10	11	12	13	14
33	15	16	17	18	19	20	21
34	22	23	24	25	26	27	28
35	29	30	31				

nächstes Treffen in **3 Wochen**

→ **Donnerstag, 4. August 2022, 20:00 Uhr**

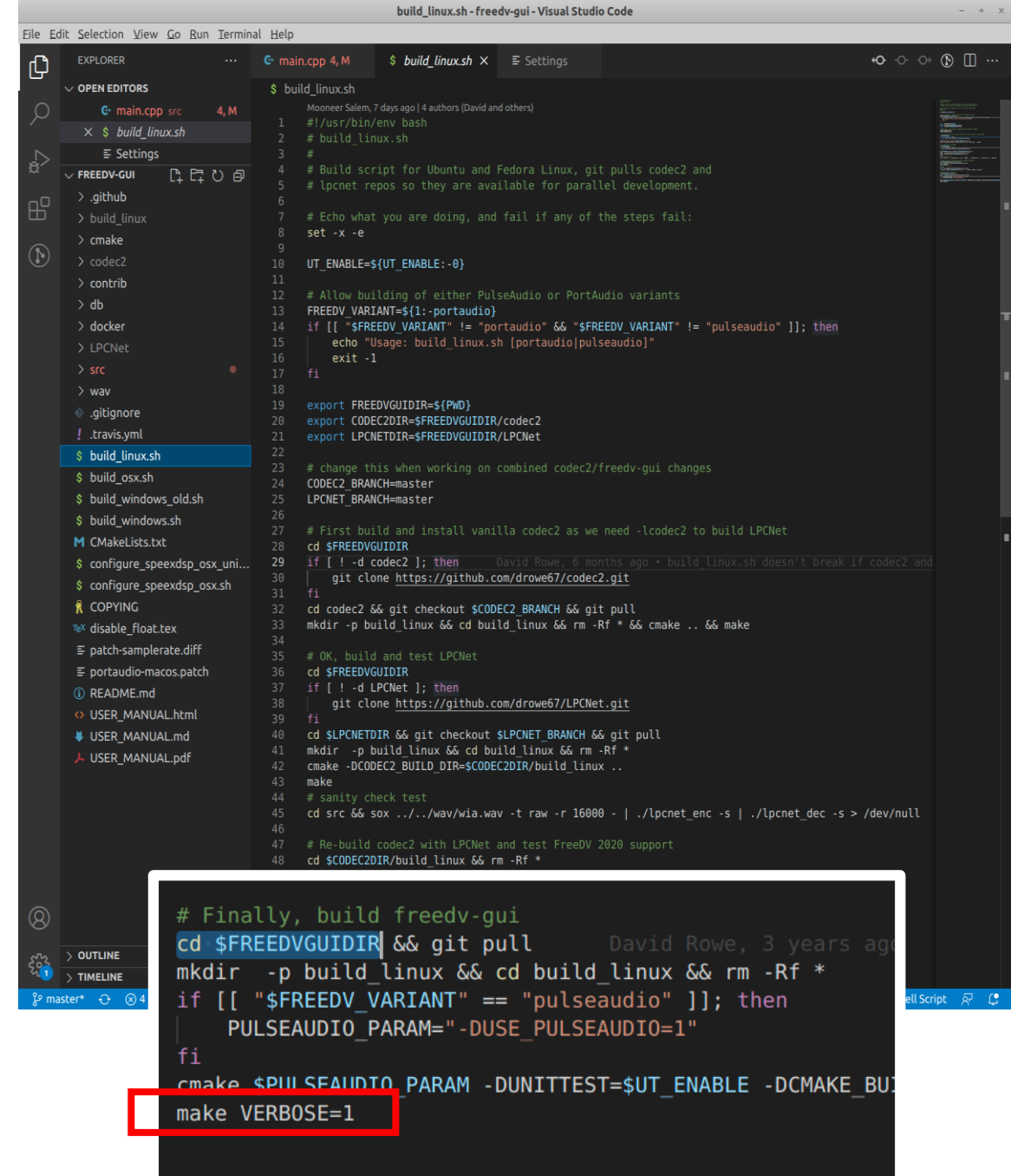
4. SW Struktur

- **Start** mit dem „git clone“ der freedv-gui Quellen
- weiter mit einem sympatischen Editor („nano“ im Terminal, sublime Text, Visual Studio Code)
- werft mal einen Blick in „*Readme.md*“
- das ist nur Text mit wenig Formatierungs-Hinweisen und sieht im Ergebnis erstaunlich gut aus!
- und lest man „*USER_MANUAL.md*“ mit einer sehr guten Zusammenfassung zu FreeDV



4. SW Struktur

- check von *build_linux.sh*
- zusätzlich „git clone“ von *codec2* und *LPCNet*
- falls das Verzeichnis nicht existiert:
→ git clone
wenn das Verzeichnis schon existiert:
→ *git checkout && git pull*
(aktualisiert ohne lokale Änderungen zu überschreiben)
- „./*build_linux.sh*“ führt man typischerweise nur einmal aus
- einen erneuten Compiler-Vorgang startet am besten im richtigen Unterverzeichnis mit „*make*“



The screenshot shows the Visual Studio Code interface with the *build_linux.sh* script open in the editor. The script is a shell script that sets up the build environment for the *freedv-gui* project. It includes comments in German and English, and contains several *git clone* and *git pull* commands to fetch the *codec2* and *LPCNet* repositories. The script also sets up the build environment with *cmake* and *make*. A terminal window is open at the bottom, showing the execution of the script. The terminal output shows the script's progress, including the cloning of the repositories and the building of the project. A red box highlights the *make VERBOSE=1* command in the terminal.

```
build_linux.sh
1 #!/usr/bin/env bash
2 # build_linux.sh
3 #
4 # Build script for Ubuntu and Fedora Linux, git pulls codec2 and
5 # lpcnet repos so they are available for parallel development.
6
7 # Echo what you are doing, and fail if any of the steps fail:
8 set -x -e
9
10 UT_ENABLE=${UT_ENABLE:-0}
11
12 # Allow building of either PulseAudio or PortAudio variants
13 FREEDV_VARIANT=${1:-portaudio}
14 if [[ "$FREEDV_VARIANT" != "portaudio" && "$FREEDV_VARIANT" != "pulseaudio" ]]; then
15     echo "Usage: build_linux.sh [portaudio|pulseaudio]"
16     exit -1
17 fi
18
19 export FREEDVGUIDIR=${PWD}
20 export CODEC2DIR=${FREEDVGUIDIR}/codec2
21 export LPCNETDIR=${FREEDVGUIDIR}/LPCNet
22
23 # change this when working on combined codec2/freedv-gui changes
24 CODEC2_BRANCH=master
25 LPCNET_BRANCH=master
26
27 # First build and install vanilla codec2 as we need -lcodec2 to build LPCNet
28 cd $FREEDVGUIDIR
29 if [ ! -d codec2 ]; then
30     git clone https://github.com/drowe67/codec2.git
31 fi
32 cd codec2 && git checkout $CODEC2_BRANCH && git pull
33 mkdir -p build_linux && cd build_linux && rm -Rf * && cmake .. && make
34
35 # OK, build and test LPCNet
36 cd $FREEDVGUIDIR
37 if [ ! -d LPCNet ]; then
38     git clone https://github.com/drowe67/LPCNet.git
39 fi
40 cd $LPCNETDIR && git checkout $LPCNET_BRANCH && git pull
41 mkdir -p build_linux && cd build_linux && rm -Rf *
42 cmake -DCODEC2_BUILD_DIR=${CODEC2DIR}/build_linux ..
43 make
44 # sanity check test
45 cd src && sox ../../wav/wia.wav -t raw -r 16000 - | ./lpcnet_enc -s | ./lpcnet_dec -s > /dev/null
46
47 # Re-build codec2 with LPCNet and test FreedV 2020 support
48 cd $CODEC2DIR/build_linux && rm -Rf *
```

```
# Finally, build freedv-gui
cd $FREEDVGUIDIR && git pull
mkdir -p build_linux && cd build_linux && rm -Rf *
if [[ "$FREEDV_VARIANT" == "pulseaudio" ]]; then
    PULSEAUDIO_PARAM="-DUSE_PULSEAUDIO=1"
fi
cmake $PULSEAUDIO_PARAM -DUNITTEST=${UT_ENABLE} -DCMAKE_BUILD_TYPE=Debug ..
make VERBOSE=1
```

4. SW Struktur

wie funktioniert eine GUI basierte Software? (1/2)

- Event-basiert
- definierte Fenster und Dialoge mit einzelnen Schaltflächen und Anzeige-Elementen (widgets)
- mit den Elementen sind Funktionen verbunden (wird ein Element aktiviert, wird die Funktion ausgelöst)
- Dialog-Fenster (Pop-Ups) sind fertig definiert und werden nur angezeigt oder wieder „versteckt“
- das Hauptprogramm ist im Prinzip eine leere Schleife,
alles passiert nach einem **Maus-Klick** oder nach Ablauf eines **Timers**
- wichtig: mehrere Prozesse innerhalb eines Programms (oder auch Threads)
Prozess: „parallele Ausführung“, gleiches Programm aber **isolierter** Speicherbereich
Thread: „parallele Ausführung“, gleiches Programm im **gleichen** Speicherbereich

4. SW Struktur

wie funktioniert eine GUI basierte Software? (2/2)

- Fenster-Inhalte sind aus einzelnen kleinen Inhalten platziert und werden miteinander angeordnet; verschiedene Toolkits haben hier jeweils eigene Methoden
- FreeDV verwendet wxWidgets; hier werden verschiedene „sizer“ verwendet
- Beispiel: dlp_ptt.cpp
- Code-Modifikation für Modus 2020
- gdb zum Debuggen
- ...