Radiosonden PlugIn SDR++ (Bullseye OS für Raspberry PI) Vorraussetztung ist ein bereits installiertes SDR++ (im Home Verzeichnis)

Dann in dem Ordner /home/pi/SDRPlusPlus ist eine CMakeLists.txt
Diese muss Editiert werden bzw ser Befehlssatz eingefügt werden.
Die CMakeLists.txt mit dem Texteditor öffnen.
Unter # Decoders folgende Zeile einfügen:

```
option(OPT_BUILD_RADIOSONDE_DECODER "Build the radiosonde decoder module (no dependencies required)" ON)
```

Dann etwas tiefer nochmals unter dem zweiten eintrag von #Decoders diese Zeile einfügen:

```
if (OPT_BUILD_RADIOSONDE_DECODER)
add_subdirectory("decoder_modules/sdrpp_radiosonde")
endif(OPT_BUILD_RADIOSONDE_DECODER)
```

Und zum schluss abspeichern.
Das CMakeLists.txt sollte dann so aussehen bei euch:

```
cmake_minimum_required(VERSION 3.13)
project(sdrpp)

if (${CMAKE_SYSTEM_NAME} MATCHES "Darwin")
    set(CMAKE_INSTALL_PREFIX "/usr/local")
else()
    set(CMAKE_INSTALL_PREFIX "/usr")
endif()

# Compatibility Options
option(OPT_OVERRIDE_STD_FILESYSTEM "Use a local version of std::filesystem on systems that don't have it yet" OFF)

# Sources
option(OPT_BUILD_AIRSPY_SOURCE "Build Airspy Source Module (Dependencies: libairspy)" ON)
option(OPT_BUILD_AIRSPYHF_SOURCE "Build Airspy HF+ Source Module (Dependencies: libairspyhf)" ON)
option(OPT_BUILD_BLADERF_SOURCE "Build BladeRF Source Module (Dependencies: libbladeRF)" OFF)
option(OPT_BUILD_FILE_SOURCE "Wav file source" ON)
option(OPT_BUILD_HACKRF_SOURCE "Build HackRF Source Module (Dependencies: libhackrf)" ON)
option(OPT_BUILD_LIMESDR_SOURCE "Build LimeSDR Source Module (Dependencies: liblimesuite)" OFF)
option(OPT_BUILD_SDDC_SOURCE "Build SDDC Source Module (Dependencies: libusb-1.0)" OFF)
option(OPT_BUILD_RFSPACE_SOURCE "Build RFspace Source Module no dependencies required)" OFF)
option(OPT_BUILD_RTL_SDR_SOURCE "Build RTL-SDR Source Module (Dependencies: librtlsdr)" ON)
option(OPT_BUILD_RTL_TCP_SOURCE "Build RTL-TCP Source Module (no dependencies required)" ON)
option(OPT_BUILD_SDRPLAY_SOURCE "Build SDRplay Source Module (Dependencies: libsdrplay)" OFF)
option(OPT_BUILD_SOAPY_SOURCE "Build SoapySDR Source Module (Dependencies: soapysdr)" ON)
option(OPT_BUILD_SPYSERVER_SOURCE "Build SpyServer Source Module (no dependencies required)" ON)
```

```
option(OPT_BUILD_PLUTOSDR_SOURCE "Build PlutoSDR Source Module
(Dependencies: libiio, libad9361)" ON)

# Sinks
option(OPT_BUILD_AUDIO_SINK "Build Audio Sink Module (Dependencies:
rtaudio)" ON)
option(OPT_BUILD_PORTAUDIO_SINK "Build PortAudio Sink Module
(Dependencies: portaudio)" OFF)
option(OPT_BUILD_NETWORK_SINK "Build Audio Sink Module (no dependencies
required)" ON)
option(OPT_BUILD_NEW_PORTAUDIO_SINK "Build the new PortAudio Sink Module
(Dependencies: portaudio)" OFF)

# Decoders
option(OPT_BUILD_FALCON9_DECODER "Build the falcon9 live decoder
(Dependencies: ffplay)" OFF)
option(OPT_BUILD_M17_DECODER "Build the M17 decoder module (no
dependencies required)" OFF)
option(OPT_BUILD_METEOR_DEMODULATOR "Build the meteor demodulator module
(no dependencies required)" ON)
option(OPT_BUILD_RADIO "Main audio modulation decoder (AM, FM, SSB,
etc...)" ON)
option(OPT_BUILD_WEATHER_SAT_DECODER "Build the HRPT decoder module (no
dependencies required)" OFF)
option(OPT_BUILD_RADIOSONDE_DECODER "Build the radiosonde decoder module
(no dependencies required)" ON)

# Misc
option(OPT_BUILD_DISCORD_PRESENCE "Build the Discord Rich Presence
module" ON)
option(OPT_BUILD_FREQUENCY_MANAGER "Build the Frequency Manager module"
ON)
option(OPT_BUILD_RECORDER "Audio and baseband recorder" ON)
option(OPT_BUILD_RIGCTL_SERVER "Rigctl backend for controlling SDR++ with
software like gpredict" ON)
option(OPT_BUILD_SCANNER "Frequency scanner" OFF)
option(OPT_BUILD_SCHEDULER "Build the scheduler" OFF)

# Other options
option(USE_INTERNAL_LIBCORRECT "Use an external version of libcorrect"
ON)
option(USE_BUNDLE_DEFAULTS "Set the default resource and module
directories to the right ones for a MacOS .app" OFF)

# Core of SDR++
add_subdirectory("core")

# Source modules
if (OPT_BUILD_AIRSPY_SOURCE)
add_subdirectory("source_modules/airspy_source")
endif (OPT_BUILD_AIRSPY_SOURCE)

if (OPT_BUILD_AIRSPYHF_SOURCE)
add_subdirectory("source_modules/airspyhf_source")
endif (OPT_BUILD_AIRSPYHF_SOURCE)

if (OPT_BUILD_BLADERF_SOURCE)
add_subdirectory("source_modules/bladerf_source")
endif (OPT_BUILD_BLADERF_SOURCE)
```

```
if (OPT_BUILD_FILE_SOURCE)
add_subdirectory("source_modules/file_source")
endif (OPT_BUILD_FILE_SOURCE)

if (OPT_BUILD_HACKRF_SOURCE)
add_subdirectory("source_modules/hackrf_source")
endif (OPT_BUILD_HACKRF_SOURCE)

if (OPT_BUILD_LIMESDR_SOURCE)
add_subdirectory("source_modules/limesdr_source")
endif (OPT_BUILD_LIMESDR_SOURCE)

if (OPT_BUILD_SDDC_SOURCE)
add_subdirectory("source_modules/sddc_source")
endif (OPT_BUILD_SDDC_SOURCE)

if (OPT_BUILD_RFSPACE_SOURCE)
add_subdirectory("source_modules/rfspace_source")
endif (OPT_BUILD_RFSPACE_SOURCE)

if (OPT_BUILD_RTL_SDR_SOURCE)
add_subdirectory("source_modules/rtl_sdr_source")
endif (OPT_BUILD_RTL_SDR_SOURCE)

if (OPT_BUILD_RTL_TCP_SOURCE)
add_subdirectory("source_modules/rtl_tcp_source")
endif (OPT_BUILD_RTL_TCP_SOURCE)

if (OPT_BUILD_SDRPLAY_SOURCE)
add_subdirectory("source_modules/sdrplay_source")
endif (OPT_BUILD_SDRPLAY_SOURCE)

if (OPT_BUILD_SOAPY_SOURCE)
add_subdirectory("source_modules/soapy_source")
endif (OPT_BUILD_SOAPY_SOURCE)

if (OPT_BUILD_SPYSERVER_SOURCE)
add_subdirectory("source_modules/spyserver_source")
endif (OPT_BUILD_SPYSERVER_SOURCE)

if (OPT_BUILD_PLUTOSDR_SOURCE)
add_subdirectory("source_modules/plutosdr_source")
endif (OPT_BUILD_PLUTOSDR_SOURCE)


# Sink modules
if (OPT_BUILD_AUDIO_SINK)
add_subdirectory("sink_modules/audio_sink")
endif (OPT_BUILD_AUDIO_SINK)

if (OPT_BUILD_PORTAUDIO_SINK)
add_subdirectory("sink_modules/portaudio_sink")
endif (OPT_BUILD_PORTAUDIO_SINK)

if (OPT_BUILD_NETWORK_SINK)
add_subdirectory("sink_modules/network_sink")
endif (OPT_BUILD_NETWORK_SINK)

if (OPT_BUILD_NEW_PORTAUDIO_SINK)
add_subdirectory("sink_modules/new_portaudio_sink")
```

```
endif (OPT_BUILD_NEW_PORTAUDIO_SINK)


# Decoders
if (OPT_BUILD_FALCON9_DECODER)
add_subdirectory("decoder_modules/falcon9_decoder")
endif (OPT_BUILD_FALCON9_DECODER)

if (OPT_BUILD_M17_DECODER)
add_subdirectory("decoder_modules/m17_decoder")
endif (OPT_BUILD_M17_DECODER)

if (OPT_BUILD_METEOR_DEMODULATOR)
add_subdirectory("decoder_modules/meteor_demodulator")
endif (OPT_BUILD_METEOR_DEMODULATOR)

if (OPT_BUILD_RADIO)
add_subdirectory("decoder_modules/radio")
endif (OPT_BUILD_RADIO)

if (OPT_BUILD_WEATHER_SAT_DECODER)
add_subdirectory("decoder_modules/weather_sat_decoder")
endif (OPT_BUILD_WEATHER_SAT_DECODER)

if (OPT_BUILD_RADIOSONDE_DECODER)
add_subdirectory("decoder_modules/sdrpp_radiosonde")
endif(OPT_BUILD_RADIOSONDE_DECODER)


# Misc
if (OPT_BUILD_DISCORD_PRESENCE)
add_subdirectory("misc_modules/discord_integration")
endif (OPT_BUILD_DISCORD_PRESENCE)

if (OPT_BUILD_FREQUENCY_MANAGER)
add_subdirectory("misc_modules/frequency_manager")
endif (OPT_BUILD_FREQUENCY_MANAGER)

if (OPT_BUILD_RECORDER)
add_subdirectory("misc_modules/recorder")
endif (OPT_BUILD_RECORDER)

if (OPT_BUILD_RIGCTL_SERVER)
add_subdirectory("misc_modules/rigctl_server")
endif (OPT_BUILD_RIGCTL_SERVER)

if (OPT_BUILD_SCANNER)
add_subdirectory("misc_modules/scanner")
endif (OPT_BUILD_SCANNER)

if (OPT_BUILD_SCHEDULER)
add_subdirectory("misc_modules/scheduler")
endif (OPT_BUILD_SCHEDULER)

add_executable(sdrpp "src/main.cpp" "win32/resources.rc")
target_link_libraries(sdrpp PRIVATE sdrpp_core)

# Compiler arguments for each platform
if (MSVC)
    target_compile_options(sdrpp PRIVATE /O2 /Ob2 /std:c++17 /EHsc)
```

```
elseif (CMAKE_CXX_COMPILER_ID MATCHES "Clang")
    target_compile_options(sdrpp PRIVATE -O3 -std=c++17 -Wno-unused-
command-line-argument -undefined dynamic_lookup)
else ()
    target_compile_options(sdrpp PRIVATE -O3 -std=c++17)
endif ()


# Copy dynamic libs over
if (MSVC)
    add_custom_target(do_always ALL xcopy /s
\"$<TARGET_FILE_DIR:sdrpp_core>\\*.dll\" \"$<TARGET_FILE_DIR:sdrpp>\" /Y)
    add_custom_target(do_always_volk ALL xcopy /s \"C:/Program
Files/PothosSDR/bin\\volk.dll\" \"$<TARGET_FILE_DIR:sdrpp>\" /Y)
endif ()


if (${CMAKE_SYSTEM_NAME} MATCHES "OpenBSD")
    add_custom_target(do_always ALL cp
\"$<TARGET_FILE_DIR:sdrpp_core>/libsdrpp_core.so\"
\"$<TARGET_FILE_DIR:sdrpp>\")
endif ()

if (${CMAKE_SYSTEM_NAME} MATCHES "FreeBSD")
    target_link_libraries(sdrpp PUBLIC pthread)
    add_custom_target(do_always ALL cp
\"$<TARGET_FILE_DIR:sdrpp_core>/libsdrpp_core.so\"
\"$<TARGET_FILE_DIR:sdrpp>\")
endif ()

if (${CMAKE_SYSTEM_NAME} MATCHES "Linux")
    add_custom_target(do_always ALL cp
\"$<TARGET_FILE_DIR:sdrpp_core>/libsdrpp_core.so\"
\"$<TARGET_FILE_DIR:sdrpp>\")
endif ()

if (${CMAKE_SYSTEM_NAME} MATCHES "Darwin")
    add_custom_target(do_always ALL cp
\"$<TARGET_FILE_DIR:sdrpp_core>/libsdrpp_core.dylib\"
\"$<TARGET_FILE_DIR:sdrpp>\")
endif ()

# cmake .. "-
DCMAKE_TOOLCHAIN_FILE=C:/dev/vcpkg/scripts/buildsystems/vcpkg.cmake" -
DOPT_BUILD_BLADERF_SOURCE=ON -DOPT_BUILD_LIMESDR_SOURCE=ON -
DOPT_BUILD_SDRPLAY_SOURCE=ON -DOPT_BUILD_M17_DECODER=ON -
DOPT_BUILD_SCANNER=ON -DOPT_BUILD_SCHEDULER=ON

# Install directives
install(TARGETS sdrpp DESTINATION bin)
install(DIRECTORY ${CMAKE_SOURCE_DIR}/root/res/bandplans DESTINATION
share/sdrpp)
install(DIRECTORY ${CMAKE_SOURCE_DIR}/root/res/colormaps DESTINATION
share/sdrpp)
install(DIRECTORY ${CMAKE_SOURCE_DIR}/root/res/fonts DESTINATION
share/sdrpp)
install(DIRECTORY ${CMAKE_SOURCE_DIR}/root/res/icons DESTINATION
share/sdrpp)
install(DIRECTORY ${CMAKE_SOURCE_DIR}/root/res/themes DESTINATION
share/sdrpp)
```

```
configure_file(${CMAKE_SOURCE_DIR}/sdrpp.desktop
${CMAKE_CURRENT_BINARY_DIR}/sdrpp.desktop @ONLY)

if (${CMAKE_SYSTEM_NAME} MATCHES "Linux")
    install(FILES ${CMAKE_CURRENT_BINARY_DIR}/sdrpp.desktop DESTINATION
share/applications)
endif ()

# Create uninstall target
configure_file(${CMAKE_SOURCE_DIR}/cmake_uninstall.cmake
${CMAKE_CURRENT_BINARY_DIR}/cmake_uninstall.cmake @ONLY)
add_custom_target(uninstall ${CMAKE_COMMAND} -P
${CMAKE_CURRENT_BINARY_DIR}/cmake_uninstall.cmake)
```

dann im Terminal folgende befehlzeile eingeben:

```
cd home
cd SDRPlusPlus
cd decoder_modules
git clone https://github.com/dbdexter-dev/sdrpp_radiosonde
```

Es hat sich nun der Ordner sdrpp_radio-sonde erstellt.

```
cd
cd SDRPlusPlus
cd build
cmake
make
sudo make install
```

Nun Compiliert sich SDRPlusPlus neu mit dem neuen Plugin.
Dieses dauert nun wieder einige Zeit.
Danach muss das Plugin im Module Manager aktiviert werden damit es funktioniert und verfügbar ist und arbeitet.

Das wars
Viel Spass dabei
Jochen Köster
DC9DD
und
Manuel Lausmann
DO3MLA