zorruno

Home > Home Automation / Weather

Using an SDR and RTL_433 in docker with MQTT

NOVEMBER 3, 2020 AUGUST 12, 2022 HOME AUTOMATION / WEATHER
COMMENTS: 0

Summary

Mainly to grab weather station data, but also useful for other 433MHz data, this is the setup I use in Docker. It uses a cheap Realtek USB SDR (software Defined Radio), and pushes data received straight to an MQTT server.

For the weather station it is proving a lot more reliable than using my USB Fineoffset weather station receiver. I'd like to build it back in to weewx though, as software like weewx does a lot of nice summaries/calcs that are necessary, whereas a setup like this just shows live data such as a rain count that is just a total cumulative number of mm.

You can pass the setup parameters to the container as env variables, but I found it more structured to set up a conf file for rtl_433 and tweak that to suit.

This setup will grab a bunch of known RF data for various remotes, transmitters etc and automatically convert the data to values then push to MQTT.

Hardware

This is the USB SDR I have
https://www.adafruit.com/product/1497
I suspect anything with the Realtek RTL2832 chipset in it would be fine however (or others if supported by RTL_433)

Docker Compose

docker-compose.yaml

```yaml
version: '3.3'

services:
  rtl-433tomqtt:
    image: 'bademux/rtl_433tomqtt:latest'
    hostname: rtl-433tomqtt
    container_name: rtl-433tomqtt
    restart: unless-stopped
    volumes:
      - /dockervolumes/rtl-433tomqtt/rtl_433.conf:/etc/rtl_433/rtl_433
      - /etc/localtime:/etc/localtime:ro
      - /etc/timezone:/etc/timezone:ro
    devices:
      - /dev/bus/usb:/dev/bus/usb
      # - /dev/bus/usb/001/002:/dev/bus/usb/001/002
```

USB device and UDEV

rtl_433 looks for the SDR device in /dev/bus/usb/xxx/xxx . This location can change if you unplug/replug the USB device or move USB ports (or reboot).

I set up a UDEV rule to set the owner of the device (based on it's vendor ID) and let the rtl_433 software just look for it under /dev/bus/usb . It is probably better not to expose all the usb devices this way to the container, but this works.

Note also, the Symlink below is created, and Debian also creates a /dev/swradio0 – neither of which rtl_433 looks for.

This is the UDEV rule and it sits at /etc/udev/rules.d/99-rtl_sdr.rules

99-rtl_sdr.rules

```
# RTL SDR
SUBSYSTEM=="usb", ATTRS{idVendor}=="0bda", ATTRS{idProduct}=="2838", M
```

rtl_433 conf file

the main thing to change in the conf file is the mqtt server address, and change the mqtt topic too if you wish (I'm using rtl_433 as the topic). The data will be pushed to mqtt under this topic, then protocol number then channel number.

```
output mqtt://192.168.1.10:1883,retain=0,devices=rtl_433/P[protocol:255]/C[channel:0]
```

rtl_433.conf

```
# config for rtl_433

# A valid config line is a keyword followed by an argument to the end
# Whitespace around the keyword is ignored, whitespace is space and ta
# Comments start with a hash sign, no inline comments, empty lines are
#
# Boolean options can be true/false, yes/no, on/off, enable/disable, o
#
# All options will be applied in the order given, overwritting previou
#
# Config files can be nested/stacked (use multiple -c and config_file
#
# If no -c option is given the first found of this list will be loaded
# - ./rtl_433.conf
# - ~/.config/rtl_433/rtl_433.conf
# - /usr/local/etc/rtl_433.conf
# - /etc/rtl_433.conf

## General options
```

```
# as command line option:
#  [-v] Increase verbosity (can be used multiple times).
#        -v : verbose, -vv : verbose decoders, -vvv : debug decoders,
# 0 = normal, 1 = verbose, 2 = verbose decoders, 3 = debug decoders, 4
verbose 0

# as command line option:
#   [-c <path>] Read config options from a file
#config_file

## Tuner options

# as command line option:
#   [-d <RTL-SDR USB device index>] (default: 0)
#   [-d :<RTL-SDR USB device serial (can be set with rtl_eeprom -s)>]
#   [-d "" Open default SoapySDR device
#   [-d driver=rtlsdr Open e.g. specific SoapySDR device
# default is "0" (RTL-SDR) or "" (SoapySDR)
device        0

# as command line option:
#   [-g <gain>] (default: 0 for auto)
# For RTL-SDR: gain in tenths of dB ("0" is auto).
# For SoapySDR: gain in dB for automatic distribution ("" is auto), or
# E.g. "LNA=20,TIA=8,PGA=2" for LimeSDR.
gain          0
#gain   25

# as command line option:
#   [-t <settings>] apply a list of keyword=value settings for SoapySD
# E.g. "antenna=A,bandwidth=4.5M,rfnotch_ctrl=false"
#settings      antenna=A,bandwidth=4.5M

# as command line option:
#   [-f <frequency>] [-f...] Receive frequency(s) (default: 433920000
# default is "433.92M", other resonable values are 315M, 345M, 915M an
frequency     433.92M

# as command line option:
#   [-H <seconds>] Hop interval for polling of multiple frequencies (d
# default is "600" seconds, only used when multiple frequencies are gi
hop_interval  600

# as command line option:
#   [-p <ppm_error] Correct rtl-sdr tuner frequency offset error (defa
# default is "0"
```

```
ppm_error     0

# as command line option:
#   [-s <sample rate>] Set sample rate (default: 250000 Hz)
# default is "250k", other valid settings are 1024k, 2048k, 3200k
sample_rate   250k

## Demodulator options

# as command line option:
#   [-R <device>] Enable only the specified device decoding protocol (
# see "protocol" section below.

# as command line option:
#   [-G] Enable blacklisted device decoding protocols, for testing onl
#register_all false
#register_all true

# as command line option:
#   [-X <spec> | help] Add a general purpose decoder (prepend -R 0 to
# see "decoder" section below.

# as command line option:
#   [-Y level=<dB level>] Manual detection level used to determine pul
#pulse_detect level=0

# as command line option:
#   [-Y auto | classic | minmax] FSK pulse detector mode.
#pulse_detect auto

# as command line option:
#   [-n <value>] Specify number of samples to take (each sample is 2 b
samples_to_read 0

## Analyze/Debug options

# as command line option:
#   [-a] Analyze mode. Print a textual description of the signal. Disa
#analyze false

# as command line option:
#   [-A] Pulse Analyzer. Enable pulse analysis and decode attempt
analyze_pulses false

# as command line option:
#   [-b] Out block size: 262144 (default)
#out_block_size
```

```
# as command line option:
#   [-M time[:<options>]|protocol|level|stats|bits|oldmodel] Add vario
# Use "time" to add current date and time meta data (preset for live i
# Use "time:rel" to add sample position meta data (preset for read-fil
# Use "time:unix" to show the seconds since unix epoch as time meta da
# Use "time:iso" to show the time with ISO-8601 format (YYYY-MM-DD"T"h
# Use "time:off" to remove time meta data.
# Use "time:usec" to add microseconds to date time meta data.
# Use "time:utc" to output time in UTC.
#   (this may also be accomplished by invocation with TZ environment v
#   "usec" and "utc" can be combined with other options, eg. "time:uni
# Use "protocol" / "noprotocol" to output the decoder protocol number 
# Use "level" to add Modulation, Frequency, RSSI, SNR, and Noise meta 
# Use "stats[:[<level>][:<interval>]]" to report statistics (default: 
#   level 0: no report, 1: report successful devices, 2: report active
# Use "oldmodel" to use to old model keys. This will be removed shortl
report_meta level
#report_meta stats
report_meta time:iso
report_meta protocol

# as command line option:
#   [-y <code>] Verify decoding of demodulated test data (e.g. "{25}fb
#test_data {25}fb2dd58

## File I/O options

# as command line option:
#   [-S none|all|unknown|known] Signal auto save. Creates one file per
#     Note: Saves raw I/Q samples (uint8 pcm, 2 channel). Preferred mo
signal_grabber none

# as command line option:
#   [-r <filename>] Read data from input file instead of a receiver
#read_file FILENAME.cu8

# as command line option:
#   [-w <filename>] Save data stream to output file (a '-' dumps sampl
#write_file FILENAME.cu8

# as command line option:
#   [-W <filename>] Save data stream to output file, overwrite existin
#overwrite_file FILENAME.cu8

## Data output options
```

```
# as command line option:
#   [-F kv|json|csv|mqtt|syslog|null] Produce decoded output in given
#     Without this option the default is KV output. Use "-F null" to r
#     Append output to file with :<filename> (e.g. -F csv:log.csv), de
#     Specify MQTT server with e.g. -F mqtt://localhost:1883
#     Add MQTT options with e.g. -F "mqtt://host:1883,opt=arg"
#     MQTT options are: user=foo, pass=bar, retain[=0|1], <format>[=to
#     Supported MQTT formats: (default is all)
#       events: posts JSON event data
#       states: posts JSON state data
#       devices: posts device and sensor info in nested topics
#     The topic string will expand keys like [/model]
#     E.g. -F "mqtt://localhost:1883,user=USERNAME,pass=PASSWORD,retai
#     Specify host/port for syslog with e.g. -F syslog:127.0.0.1:1514
# default is "kv", multiple outputs can be used.
#output json
output mqtt://192.168.1.10:1883,retain=0,devices=rtl_433/P[protocol:25

# as command line option:
#   [-C] native|si|customary Convert units in decoded output.
# default is "native"
#convert si
convert native

# as command line option:
#   [-T] specify number of seconds to run
#duration 0

# as command line option:
#   [-E] Stop after outputting successful event(s)
stop_after_successful_events false

## protocols to enable (command line option "-R")

  protocol 1    # Silvercrest Remote Control
  protocol 2    # Rubicson Temperature Sensor
  protocol 3    # Prologue, FreeTec NC-7104, NC-7159-675 temperature se
  protocol 4    # Waveman Switch Transmitter
#   protocol 6   # ELV EM 1000
#   protocol 7   # ELV WS 2000
  protocol 8    # LaCrosse TX Temperature / Humidity Sensor
  protocol 10   # Acurite 896 Rain Gauge
  protocol 11   # Acurite 609TXC Temperature and Humidity Sensor
  protocol 12   # Oregon Scientific Weather Sensor
#   protocol 13  # Mebus 433
#   protocol 14  # Intertechno 433
  protocol 15   # KlikAanKlikUit Wireless Switch
  protocol 16  # AlectoV1 Weather Sensor (Alecto WS3500 WS4500 Ventus
  protocol 17  # Cardin S466-TX2
  protocol 18  # Fine Offset Electronics, WH2, WH5, Telldus Temperatur
  protocol 19  # Nexus, FreeTec NC-7345, NX-3980, Solight TE82S temper
  protocol 20  # Ambient Weather Temperature Sensor
  protocol 21  # Calibeur RF-104 Sensor
#   protocol 22  # X10 RF
  protocol 23  # DSC Security Contact
#   protocol 24  # Brennenstuhl RCS 2044
  protocol 25  # Globaltronics GT-WT-02 Sensor
  protocol 26  # Danfoss CFR Thermostat
  protocol 29  # Chuango Security Technology
  protocol 30  # Generic Remote SC226x EV1527
  protocol 31  # TFA-Twin-Plus-30.3049, Conrad KW9010, Ea2 BL999
  protocol 32  # Fine Offset Electronics WH1080/WH3080 Weather Station
  protocol 33  # WT450, WT260H, WT405H
  protocol 34  # LaCrosse WS-2310 / WS-3600 Weather Station
  protocol 35  # Esperanza EWS
  protocol 36  # Efergy e2 classic
#   protocol 37  # Inovalley kw9015b, TFA Dostmann 30.3161 (Rain and t
  protocol 38  # Generic temperature sensor 1
  protocol 39  # WG-PB12V1 Temperature Sensor
  protocol 40  # Acurite 592TXR Temp/Humidity, 5n1 Weather Station, 60
  protocol 41  # Acurite 986 Refrigerator / Freezer Thermometer
  protocol 42  # HIDEKI TS04 Temperature, Humidity, Wind and Rain Sens
  protocol 43  # Watchman Sonic / Apollo Ultrasonic / Beckett Rocket o
  protocol 44  # CurrentCost Current Sensor
  protocol 45  # emonTx OpenEnergyMonitor
  protocol 46  # HT680 Remote control
  protocol 47  # Conrad S3318P, FreeTec NC-5849-913 temperature humidi
  protocol 48  # Akhan 100F14 remote keyless entry
  protocol 49  # Quhwa
  protocol 50  # OSv1 Temperature Sensor
  protocol 51  # Proove / Nexa / KlikAanKlikUit Wireless Switch
  protocol 52  # Bresser Thermo-/Hygro-Sensor 3CH
  protocol 53  # Springfield Temperature and Soil Moisture
  protocol 54  # Oregon Scientific SL109H Remote Thermal Hygro Sensor
  protocol 55  # Acurite 606TX Temperature Sensor
  protocol 56  # TFA pool temperature sensor
  protocol 57  # Kedsum Temperature & Humidity Sensor, Pearl NC-7415
  protocol 58  # Blyss DC5-UK-WH
  protocol 59  # Steelmate TPMS
  protocol 60  # Schrader TPMS
#   protocol 61  # LightwaveRF
#   protocol 62  # Elro DB286A Doorbell
  protocol 63  # Efergy Optical
#   protocol 64  # Honda Car Key
```

```
####  protocol 67  # Radiohead ASK
  protocol 68  # Kerui PIR / Contact Sensor
  protocol 69  # Fine Offset WH1050 Weather Station
  protocol 70  # Honeywell Door/Window Sensor, 2Gig DW10/DW11, RE208 r
  protocol 71  # Maverick ET-732/733 BBQ Sensor
#   protocol 72  # RF-tech
  protocol 73  # LaCrosse TX141-Bv2, TX141TH-Bv2, TX141-Bv3, TX141W, T
  protocol 74  # Acurite 00275rm,00276rm Temp/Humidity with optional p
  protocol 75  # LaCrosse TX35DTH-IT, TFA Dostmann 30.3155 Temperature
  protocol 76  # LaCrosse TX29IT Temperature sensor
  protocol 77  # Vaillant calorMatic VRT340f Central Heating Control
  protocol 78  # Fine Offset Electronics, WH25, WH32B, WH24, WH65B, HP
  protocol 79  # Fine Offset Electronics, WH0530 Temperature/Rain Sens
  protocol 80  # IBIS beacon
  protocol 81  # Oil Ultrasonic STANDARD FSK
  protocol 82  # Citroen TPMS
  protocol 83  # Oil Ultrasonic STANDARD ASK
  protocol 84  # Thermopro TP11 Thermometer
  protocol 85  # Solight TE44/TE66, EMOS E0107T, NX-6876-917
  protocol 86  # Wireless Smoke and Heat Detector GS 558
  protocol 87  # Generic wireless motion sensor
  protocol 88  # Toyota TPMS
  protocol 89  # Ford TPMS
  protocol 90  # Renault TPMS
  protocol 91  # inFactory, FreeTec NC-3982-913 temperature humidity s
  protocol 92  # FT-004-B Temperature Sensor
  protocol 93  # Ford Car Key
####  protocol 94  # Philips outdoor temperature sensor (type AJ3650)
  protocol 95  # Schrader TPMS EG53MA4, PA66GF35
  protocol 96  # Nexa
  protocol 97  # Thermopro TP08/TP12/TP20 thermometer
  protocol 98  # GE Color Effects
  protocol 99  # X10 Security
  protocol 100 # Interlogix GE UTC Security Devices
#   protocol 101 # Dish remote 6.3
  protocol 102 # SimpliSafe Home Security System (May require disablin
  protocol 103 # Sensible Living Mini-Plant Moisture Sensor
  protocol 104 # Wireless M-Bus, Mode C&T, 100kbps (-f 868950000 -s 12
  protocol 105 # Wireless M-Bus, Mode S, 32.768kbps (-f 868300000 -s 1
#   protocol 106 # Wireless M-Bus, Mode R, 4.8kbps (-f 868330000)
#   protocol 107 # Wireless M-Bus, Mode F, 2.4kbps
  protocol 108 # Hyundai WS SENZOR Remote Temperature Sensor
  protocol 109 # WT0124 Pool Thermometer
  protocol 110 # PMV-107J (Toyota) TPMS
  protocol 111 # Emos TTX201 Temperature Sensor
  protocol 112 # Ambient Weather TX-8300 Temperature/Humidity Sensor
  protocol 113 # Ambient Weather WH31E Thermo-Hygrometer Sensor, EcoWi
  protocol 114 # Maverick et73
  protocol 115 # Honeywell ActivLink, Wireless Doorbell
  protocol 116 # Honeywell ActivLink, Wireless Doorbell (FSK)
#   protocol 117 # ESA1000 / ESA2000 Energy Monitor
#   protocol 118 # Biltema rain gauge
  protocol 119 # Bresser Weather Center 5-in-1
#   protocol 120 # Digitech XC-0324 temperature sensor
  protocol 121 # Opus/Imagintronix XT300 Soil Moisture
#   protocol 122 # FS20
#   protocol 123 # Jansite TPMS Model TY02S
###  protocol 124 # LaCrosse/ELV/Conrad WS7000/WS2500 weather sensors
  protocol 125 # TS-FT002 Wireless Ultrasonic Tank Liquid Level Meter
  protocol 126 # Companion WTR001 Temperature Sensor
  protocol 127 # Ecowitt Wireless Outdoor Thermometer WH53/WH0280/WH02
  protocol 128 # DirecTV RC66RX Remote Control
#   protocol 129 # Eurochron temperature and humidity sensor
  protocol 130 # IKEA Sparsnas Energy Meter Monitor
  protocol 131 # Microchip HCS200 KeeLoq Hopping Encoder based remotes
  protocol 132 # TFA Dostmann 30.3196 T/H outdoor sensor
  protocol 133 # Rubicson 48659 Thermometer
###  protocol 134 # Holman Industries iWeather WS5029 weather station
####  protocol 135 # Philips outdoor temperature sensor (type AJ7010)
  protocol 136 # ESIC EMT7110 power meter
  protocol 137 # Globaltronics QUIGG GT-TMBBQ-05
  protocol 138 # Globaltronics GT-WT-03 Sensor
  protocol 139 # Norgo NGE101
  protocol 140 # Elantra2012 TPMS
  protocol 141 # Auriol HG02832, HG05124A-DCF, Rubicson 48957 temperat
  protocol 142 # Fine Offset Electronics/ECOWITT WH51 Soil Moisture Se
  protocol 143 # Holman Industries iWeather WS5029 weather station (ol
  protocol 144 # TBH weather sensor
  protocol 145 # WS2032 weather station
  protocol 146 # Auriol AFW2A1 temperature/humidity sensor
###  protocol 147 # TFA Drop Rain Gauge 30.3233.01
###  protocol 148 # DSC Security Contact (WS4945)
  protocol 149 # ERT
#  protocol 150 # Klimalogg
##  protocol 151 # Visonic powercode
##  protocol 152 # Eurochron EFTH-800 temperature and humidity sensor
##  protocol 153 # Cotech 36-7959 wireless weather station with USB
##  protocol 154 # Standard Consumption Message Plus (SCMplus)
##  protocol 155 # Fine Offset Electronics WH1080/WH3080 Weather Stati
##  protocol 156 # Abarth 124 Spider TPMS
##  protocol 157 # Missil ML0757 weather station
##  protocol 158 # Sharp SPC775 weather sensor
##  protocol 159 # Insteon
##  protocol 160 # Interval Data Message (IDM)
##  protocol 161 # Interval Data Message (IDM) for Net Meters
# protocol 162 # ThermoPro-TX2 temperature sensor
##  protocol 163 # Acurite 590TX Temperature with optional Humidity
##  protocol 164 # Security+ 2.0 (Keyfob)
##  protocol 165 # TFA Dostmann 30.3221.02 T/H Outdoor Sensor
##  protocol 166 # LaCrosse Technology View LTV-WSDTH01 Breeze Pro Win

## Flex devices (command line option "-X")

# Some general decoder definitions for various devices, enable as need
#
# For details about decoder definition run "rtl_433 -X help"
#

# If you enable these decoders you'll likely want to add ",match=<YOUR

# Elro DB270 - wireless doorbell
#
# Device information and test files:
# https://github.com/merbanan/rtl_433_tests/tree/master/tests/elro/db2
#
# Output sample:
# {"time" : "2018-02-14 19:11:16", "model" : "Elro_DB270", "count" : 4
#  "rows" : [{"len" : 25, "data" : "ebeaaa8"}, {"len" : 25, "data" : "
#            {"len" : 25, "data" : "ebeaaa8"}, {"len" : 25, "data" : "
#
#decoder n=Elro_DB270,m=OOK_PWM,s=300,l=930,r=11000,g=1500,repeats>=4,

# Euroster 3000TX - programmable room thermostat
#
# Device information and test files:
# https://github.com/merbanan/rtl_433_tests/tree/master/tests/euroster
#
# Output sample:
# {"time" : "2018-02-14 19:20:20", "model" : "Euroster_3000TX", "count
#  "rows" : [{"len" : 32, "data" : "41150515"}]}
#
#decoder n=Euroster_3000TX,m=OOK_MC_ZEROBIT,s=1000,r=4800,bits=32

# Byron BY series door bell
#
# Device information and test files:
# https://github.com/merbanan/rtl_433_tests/tree/master/tests/Byron-BY
#
# Output sample:
# {"time" : "@1.572864s", "model" : "doorbell#1", "count" : 25, "num_r
#decoder n=Byron_BY_Doorbell,m=OOK_PWM,s=500,l=1000,r=3300,g=1200,repe
```

```
# Kerui alarm system (PIR and door sensors)
#  short is 333 us
#  long is 972 us
#  packet gap 11000 us
#decoder n=Kerui,m=OOK_PWM,s=333,l=972,r=11000,g=1100,bits=25,invert,g

# Golden Security GS-WDS07 door and window sensor
#  short is 476 us + 1344 us
#  long is 1364 us + 448 us
#  packet gap 13972 us
#decoder n=gswds07,m=OOK_PWM,s=476,l=1364,r=15000,g=1600,bits>=24,bits

# Generic SCV2260 4-button remote (see rtl_433_tests/tests/generic_rem
#  short is 472 us + 1412 us
#  long is 1428 us + 472 us
#decoder n=generic_remote_01,m=OOK_PWM,s=472,l=1428,r=1800,g=1600,bits

# Generic PT2260 PIR (see rtl_433_tests/tests/PT2262/01)
#  short is 440 us + 1536 us
#  long is 1428 us + 548 us
#  packet gap 15348 us
#decoder n=pt2260_pir,m=OOK_PWM,s=440,l=1428,r=16000,g=1700,bits=25,in
```

RTL_433 Github https://github.com/merbanan/rtl_433
Plugin for Weewx to use an SDR to gather data https://github.com/matthewwall/weewx-sdr
Github for this docker build https://github.com/bademux/rtl_433toMQTT
Dockerfile for this build https://github.com/bademux/rtl_433toMQTT/blob/master/Dockerfile

Notes

I was getting errors after enabling more than about 148 protocols in the rtl_433 conf file. Not sure why and haven't investigated further.

This is how to determine which /dev/usb/bus ID the device is on. It might be better to pass this as an env variable and expose this device only.

```
vidPid="0bda:2838"
devPath="/dev/bus/usb/$(lsusb -d $vidPid | sed 's/^.*Bus\s\([0-9]\+\)\
chown $USER $devPath
```

Categories: Home Automation Weather

Tags: Debian Docker Home Automation Linux MQTT RTL_433 SDR Weather Weatherstation WeeWX

Leave a Reply

Login with Facebook

Your email address will not be published. Required fields are marked *

Comment *

Name *

Email *

Website

☐ **Save my name, email, and website in this browser for the next time I comment.**

Post Comment