FOSHKplugin - generic version

Even if the core task of the program is actually the conversion of incoming data from a weather station in WU or Ecowitt format to UDP for any target like Loxone Miniserver, it does offer a few functions that go beyond this.

This plugin connects various weather stations and sensors from the manufacturer Fine Offset Electronics (FOSHK) to a Loxone Miniserver or any other smarthome-system via UDP.
Actually all weather stations are supported where a custom server can be set up as a destination for transmitting the data in WU or Ecowitt format. If you can change or redirect the DNS server for the weather station, it works with any weather station which send it's data to Weather Underground or Ecowitt.
The "custom server" for the GW1000 (or HP2551C or WH2910) is configured remotely by FOSHKplugin itself. For other weather stations you have to configure the custom server via WS View or the manufacturer's configuration program.

Version History...

Version 0.10 - not yet released - public beta test

- changed: maxdailygust is now maxdailygustkmh in metric data (kmh) - the unit of maxdailygust is mph - **ATTENTION! may cause compatibility issues!**
- fixed: typo in generic-FOSHKplugin-install.sh fixed sytemd –> systemd
- fixed: UDP command Plugin.intvlwarning=enable/disable triggered co2warning instead (copy/paste-Error)
- fixed: min/max values for leaf moisture sensors 1-8 were not written to the daily CSV
- fixed: WS90 rain-handling - replacing the new names with the old ones now works correctly (e.g. "rrain_piezo" –> "rainratein")
- fixed: scanWS MAC address output
- fixed: Sending of additional temp/hum sensors now according to WeatherCloud API documentation: temp02/hum02
- fixed: sunhours calculation - formula corrected like WeeWX extension (Jterrettaz)
- fixed: modifications for altered AWEKAS-API (response OK)
- fixed: possible error in the creation of the daily CSV fixed
- improved: improved 10 minutes average calculation "winddir_avg10m" (avgWind)
- improved: ignore "@" in the field identifier of OUT_TEMP and OUT_HUM to standardise the assignments as with FWD_REMAP
- improved: Pushover notification for missing weather station changed to red text color
- improved: existing values can now be assigned via FWD_REMAP to the WeatherCloud keys co, no, no2, so2, o3, tempagro, et, pwrsply, battery and noise
- improved: sending pm25 & aqi of WH41/43 channel #1 to WeatherCloud if no WH45 present (WH45 still preferred)
- improved: conversion to WU: tempNf, humidityN, qcStatus, softwareType
- improved: better error handling for saving as file via ftp(s)
- improved: error handling for Weathercloud forwards
- improved: error handling for Pushover push notifications (automatic retry)
- improved: parameter -scanWS gathers all weather stations now - including GW2000
- improved: WH45 for WU - if no WH41 present, use data from WH45 instead

- improved: internal functions stringToDict & getfromDict
- improved: if the structure of the daily CSV is changed, a new file is created
- new: introducing REBOOT_WARNING - check wether current runtime ⇐ last runtime - if so: warn (log, push, UDP: rebootwarning=1 dailyboot=#count)
  can be deactivated with http: /FOSHKplugin/rebootwarning=disable and UDP: Plugin.rebootwarning=disable
  LoxBerry: commands are included as virtual outputs in the template
- new: "dailyboot" (number of restarts) and "rebootwarning" as keys, reset at midnight and is also part of the daily CSV (if EVAL_VALUES active)
  LoxBerry: Keys "rebootwarning" and "dailyboot" are included as virtual inputs in the template
- new: "sunshine" represents the presence of sunshine; can be artificially prolonged with SUNSHINE_HOLD to prevent constant changes
  LoxBerry: key "sunshine" also included in template as virtual input
- new: introducing SUNSHINE_HOLD = seconds - Hold time in seconds for value sunshine, this time continues to be output sunshine = True, even if there is no sunshine (default: 0)
- new: FWD_TYPE = MIYO; sends temperature, wind and rain state (rain = rainratemm > 0 or hourlyrainmm > 1 or dailyrainmm > 1) to a MIYO cube (irrigation system)
  rain forecast is still missing; could be implemented too via API AerisWeather or AccuWeather (both in test)
- new FWD_TYPE = INFLUX2MET and INFLUX2IMP - support of InfluxDB2 - Python 3.6 or later and Python lib influxdb-client are required!
  bucket = dbname; org = fwd_sid; token = fwd_pwd
  missed forwards are queued and transmitted when the destination is available again
- new: (Loxone) virtual input FOSHK-getMinMax for command Plugin.getminmax integrated in Loxone template
- new: send the fwd_type, prgname & prgver as an argument with postFile
- new: FWD_WARNING, FWD_WARNINT, http://ipaddress:portnumber/FOSHKplugin/fwdstat, enable/disable via http & UDP
- new: BATTERY_WARNING may be enabled/disabled via http & UDP
- new: support for leafwetness sensors for Ambient Weather stations
- new: change the date/time output format via Config\DT_FORMAT (default = %d.%m.%Y %H:%M:%S - dd.mm.yyyy hh:mm:ss) for all (!) date/time outputs: log, csv, push
- new: forward warning via Pushover
  after a configurable number of successive unsuccessful attempts, a push message is sent through Pushover
- new: statistics page for forwards: http://ipaddress:portnumber/FOSHKplugin/fwdstat
- new: scan weather station page: http://ipaddress:portnumber/FOSHKplugin/scanWS
- new: missed forwards for forward types EW and RAWEW may be queued and sent when destination is available again (with FWD-xx\FWD_QUEUE = True)
- new: exclude list for battery warning: BATTERY_WARNEXCLUDE - a comma separated list of keys to exclude from battery warning e.g. wh90batt
- new: Awekas-API: deliver weather report conditions & tendency
  missed forwards are automatically queued as FOSHKplugin-queue/FWD-nr/FOSHKplugin-queued-data-nr.csv in config dir (or dir configured as FWD_QDIR)
- new: Adaptations for installation under (Open)Suse, Synology, Fedora (zypper, ipkg, dnf) in generic version
- new: custom push notifications via Pushover (user-defined push messages when values exceed or fall below a definable value)
- new: Conversion to UTF-8 of all programme parts (may cause problems)
- new: Weather Report for Awekas - automatically report rain, storm and thunderstorm

- new: FWD_TYPE = BANNER - automatic generation of banners and stickers with current weather data
- new: further queryable key names for getvalue, JSON, ... and forward types MQTT, InfluxDB, BANNER and TAGFILE:
  prgname (FOSHKplugin)
  prgver (current version number)
  winddirtext (if available, 10min winddir-mean otherwise winddir as short text (N, NE, NNO, ...)
  aqtime (Time of processing by FOSHKplugin)
  pchange1in (1 hour pressure change in inHg)
  pchange3in (3 hour pressure change in inHg)
  lightningmi (lightning distance in miles)
  starttime (start time of FOSHKplugin)
- new: FWD_TYPE = TAGFILE - user-defined output format based on tags and templates
- new: config option Export\LIMIT_WINDGUST = n to prevent processing of unrealistic values for windgustmph and maxdailygust (e.g. for WS80/WS90).
  if value >= n the windgustmph will be renamed to _windgustmph (thus not processed); last "good" maxdailygust will be used as maxdailygust
- new: calculation of windrun (in miles) and windrunkm (in km) and daily solar radiation sum (srsum)
- new: with getvalue, the additional parameter &comma can be used to force the output of a comma (",") instead of the dot (".") in numeric values
- new: with Config\LINK_ADR = address in Config file you may specify a name or address for all links created by FOSHKplugin (e.g. for use on public web server)
- new: support of Debian Bookworm based distributions by using virtual environment venv (problem with installation of required Python libraries with pip - PEP 668)
- new: with Export\ADD_DEWPT = True you can enable/disable the dew point calculation for indoor sensor, WH31 and WH45 - default: False
  the keys are dewptinf (indoor T/H sensor), dewptNf (WH31; where N=1..8), dewptf_co2 (for WH45) and for metric units: dewptinc, dewptNc, dewptc_co2
  Loxone: new metric keys dewptinc, dewptNc, dewptc_co2 added to the Loxone template
- new: with Logging\COLOR_PRINT (default: True), messages in the console window are highlighted in colour (ERROR = red, WARNING = yellow and after a warning has been cancelled = green; can be deactivated with COLOR_PRINT = False
- new: get complete dictionary with http://ipaddress:portnumber/FOSHKplugin/getFullDict (with options like separator, sorted, json)
- new: supports the Prometheus time series database - see https://foshkplugin.phantasoft.de/generic#prometheus
- new: Support for the old HP1001 console (conversion to WU format)
- new: use "&human" to output the time as readable time (e.g. dd.mm.yyyy hh:mm:ss) for time-specific getvalue queries; output format and locale may be specified (defaults to DT_FORMAT & LANGUAGE)
  Example: http://192.168.15.100:8096/FOSHKplugin/getvalue?key=aqtime&human&format="%A %x %H:%M:%S"&locale=nl_NL.UTF-8 –> zaterdag 03-02-24 10:17:22
- new: with Export\ADD_SPREAD = True (default: False) there will be additionally spread values for indoor, outdoor and WH45 sensor as well as all WH31 calculated
- changed: all incoming get-requests will be URL-decoded now
- new: enable signal quality acquisition on supported consoles with Export\ADD_SIGNAL = True (default: False)

Version 0.09 - 02.04.2022

- Minor bug fixes and optimisations
- intensive code cleaning - renaming and standardisation of the conversion functions
- fixed bug with incoming data in WU protocol and activated EVAL_VALUES
- HTTP requests with &refresh=n refresh the displayed page every n secondsExample: http://ipaddress:portnumber/APRS&refresh=30 updates the displayed page of the APRS output string every 30 seconds or: http://ipaddress:portnumber/status&minmax&refresh=60 shows the current values including status and min/max values and updates the page every 60 seconds
- for incoming data in WU format, equate barominrelin with baromrelhpa, conversion from WU to EW modified for WH6006
- internal WU server: WN34 and WN35 compatibility established
- when converting to Ambient Weather, wh80batt is correctly set to battout
- better logging for FWD_EXEC - FWD number is logged for better allocation; display of a change is only done in case of an actual change
- leaf moisture level for Meteotemplate, WC, Awekas, Weather365 and WSWin - instead of 0..99 now 0..15 (float) is sent as level
- alternative names for RAWEW (EWRAW), RAWUDP (UDPRAW), RAWCSV (CSVRAW), AMBRAW (RAWAMB) and TXTFILE (TEXTFILE) introduced
- in the FWD_URL for output formats REALTIMETXT, CLIENTRAWTXT, CSVFILE, WSWIN, TXTFILE and RAWTEXT a file name can now also be transferred
- when creating the WSWin-CSV, new data can now be appended to the existing file via http(s)/POST and ftp(s).
- new, improved sunhours calculation sunhours (according to https://github.com/Jterrettaz/sunduration) with dynamic, location-dependent threshold (thanks Werner!), requires Coordinates\LAT and Coordinates\LON\\without LAT/LON or with Sunduration\SUN_CALC = False the already known calculation with fixed threshold of 120W/m² is used
  can be used with Sunduration\SUN_MIN (minimum threshold, default=0) and Sunduration\SUN_COEF (default=0.8 - too little sunshine recorded: decrease value; too much: increase) can be modified.
  for compatibility reasons this function must be activated with Sunduration\SUN_CALC = True
- html-query for WSWIN implemented - http://ipadresse:portnummer/WSWIN outputs a WSWin-compatible data line of the last values
- new forward type EWUDP (UDPEW) - converts incoming EW, WU and AMB messages to Ecowitt/UDP (e.g. for Personal Weather Tablet/UDP broadcast listener)
- FWD_IGNORE for filtering all outgoing keys now valid for all forwards - keys in this list are not sent out
- Remap function FWD_REMAP implemented - output keys can now be defined with values of all known internal keys
  Some targets support only a selection of sensors, e.g. Ambient Weather supports only one internal/external PM2.5 sensor or Awekas or WSWin only 4 soil moisture sensors.
  However, FOSHKplugin always transmits logically sequentially - i.e. starts with the first sensor in each case and transmits the valid max. number of channels in each case.
  With FWD_REMAP, a corresponding assignment or selection can be made.
  Example: FWD_REMAP = @tf_ch1c=@tf_ch8c # sets the metric key tf_ch1c (corresponds to metric temperature value 1st channel) to the value of tf_ch8c (8th channel)
  values of other keys (with @) and static values can be assigned (e.g. @tf_ch1c=12.3) and own keys can be defined (soiltemp2=@tf_ch7c);
- new forward type APRS allows data to be sent to CWOP
  call sign is passed as FWD_SID, a possibly necessary password can be passed with FWD_PWD; FWD_URL contains address:port of destination

can also be retrieved via http with http://ipadresse:portnummer/APRS?user=CALLSIGN
- Weather365: Ground temperatures of sensors 2..4 are now also transmitted - note the possibly necessary remapping!
- MeteoTemplate: Support of WN35 (leaf moisture) and WN34 as soil temp/TSn - depth can be added with TS0n=cm as ADD_ITEM or via FWD_REMAP
- MeteoTemplate: battery values of PM2.5 sensors are now output with PPnBAT instead of PMnBAT
- Support of the WS90 sensor (wh90batt)
- If output data is to be processed by script (FWD_EXEC) but not sent, this can be realised with a return message of "EXECONLY" from the script: *echo EXECONLY* as the last output command in the script.

Version 0.08 - 27.06.2021

- minor bugfixes
- extended debug options in the Loxone version (service.sh support and service.sh supzip - for automatic collection of debug data.)
- Better logging in the send log: the number of the forward is also logged in order to be able to find faulty blocks in the config file more easily
- Adaptable logging level - less logging with a view to the essentials
  Logging can now be fine-tuned via Logging\LOG_LEVEL in the config file - with
  - ALL, as before, all lines are logged
  - INFO - all lines except ERROR, WARNING, INFO and OK are hidden
  - WARNING - all lines except ERROR and WARNING and OK are hidden
  - ERROR - only lines with ERROR and OK are output
  For reasons of compatibility, ALL is preset - however, I recommend LOG_LEVEL INFO - so everything that was not successful is logged
- LOG_LEVEL can also be adjusted during operation via http://ipaddress:portnumber/FOSHKplugin/loglevel=[ALL, INFO, WARNING, ERROR] - however, the value set in the config file applies again after a restart
- with Logging\LOG_ENABLE = False, logging can be switched off globally without making changes to the log file names
- more flexible firmware update check (duplicate options for broken Ecowitt-file)
- Increased transmission security with http(s) and ftp(s) forwards through transmission repetition: 3 attempts are now made to deliver the data; the second attempt takes place after 5 seconds and the third after another 10 seconds
  however, it is not repeated if the return code indicates a local error (400..499)
- Support of forward via MQTT for both metric values (FWD_TYPE = MQTTMET) and imperial values (FWD_TYPE = MQTTIMP)
  requires python3-setuptools and paho-mqtt (will be installed automatically)
  MQTT broker is defined via the FWD_URL: ipaddress:portnumber@hierarchy%prefix - topic name is the name of the key
- Support of the wetter.com API (still under weewx pseudonym)
- Support of the weather365.net API
- Support of the wettersektor.de API
- Possibility to export the data as realtime.txt or clientraw.txt per file, http(s), ftp(s) with FWD_Typ = REALTIMETXT or CLIENTRAWTXT (experimental)
  the FWD_URL specifies whether the file should be posted via http(s), transferred via ftp(s) or stored as a file in the filesystem
  a realtime.txt-compatible string can be queried and saved via http://ipaddress:portnumber/realtime.txt

a clientraw.txt-compatible string can be queried and saved
via http://ipaddress:portnumber/clientraw.txt

- Storage as WSWin-compatible CSV file wswin.csv for automatic import by WSWin via file monitoring
  the storage location must be readable (if necessary, writable) for Windows computers via the Samba share
  the import can be done automatically and irregularly by WSWin - WSWin simply reads in all lines that have not yet been processed
  no X-CSV is necessary
- Export option as text or CSV file with included key names per file, http(s) and ftp(s) with FWD_TYPE = TXTFILE or CSVFILE - existing files are overwritten; these formats can also be requested via http://ipaddress:portnumber/FOSHKplugin.txt or .csv
- export as a text file with raw keys/values with FWD_TYPE = RAWTEXT - existing file will be overwritten on each interval
- separator for output via /STRING and /RAW can now be %20 (as a space) or a whole word in addition to a single character
- Calculation of the cloud height cloudf (in feet) or cloudm (in meters) implemented, Coordinates\ALT = True in the config file with height above sea level in meters required - requires EVAL_VALUES = True
- Sunshine duration sunhours implemented - shows the duration of the daily sunshine duration in hours (solarradiation >= 120W/m²) - requires EVAL_VALUES = True
- sunhours, co2 and leafwetness (for the coming WN34) supported in meteotemplate; sunhours also with Awekas-API
- Coordinates can be configured under Coordinates\ALT, LAT, LON
  ALT is used to determine the cloud height (spread * 122)
  LAT/LON are only used for transferring to the export formats Awekas-API, clientraw.txt and Weather365.net
- Config\UDP_MAXLEN (default = 2000) defines the maximum length of a UDP datagram
  If the length of the packet to be sent is greater than specified, the packet is divided into several datagrams, each of which has an approximate length UDP_MAXLEN and contains the identifier SID=FOSHKweather at the beginning of each datagram.
  The original line is separated in such a way that the key=value assignment is retained - the separation always takes place **BEHIND** UDP_MAXLEN the next time a space is found.
  Values that contain spaces are surrounded by double quotation marks so that there is a possibility of parsing on the UDP server side.
  Example neighborhood="New York City" for a value with spaces - but neighborhood=Berlin (without spaces)
- With Warning\CO2_WARNING = True a warning (UDP, http, log, Pushover) can be activated if the CO2 value is higher than the one configured under Warning\CO2_WARNLEVEL
- With Config\UDP_STATRESEND = n, a cycle time (n seconds) can be defined in which the warning messages are sent regardless of status changes
- Changes in ptrend1 and ptrend3 - if the air pressure rises or falls rapidly (+0.7 / -0.7 or +2/-2), a trend of 2 or -2 is output
- ptrend1, pchange1, ptrend3 and pchange3 are now also forwarded via Ecowitt (Type = EW) if EVAL_VALUES = True
  pchange1 and pchange3 contain the difference between the current value and the value from 1 or 3 hours ago in inHg
- with CSV\CSV_DAYFILE = /path/to/filename.csv the creation of a additional daily CSV /path/to/filename.csv with the min/max values of the day and some day-wide calculations will be activated

- Parameters bool, status, units and separator are now possible for all requests (as far as it makes sense - so not e.g. with RAW)
- Query /JSON extended - generates a JSON with numerical boolean values when called with the parameter bool - can be extended with minmax, status and units
  Example: http://ipaddress:portnumber/JSON?minmax&status&bool outputs the last data record with numerical boolean values 1/0 instead of strings True/False
- MQTTMET/MQTTIMP-Forward now outputs numerical boolean values (0/1 instead of False/True)
- InfluxDB support: both via pull (as JSON via telegraf) and natively (FWD_TYPE = INFLUXMET/INFLUXIMP) integrated
  the database is specified in the FWD_URL line: FWD_URL = http://192.168.15.237:8086@Database creates a database Database on 192.168.15.237:8086 and transmits the data unencrypted at the configured interval; username and password can be transferred via FWD_SID and FWD_PWD
- Config-options Config\REBOOT_ENABLE and Config\RESTART_ENABLE to enable reboot of weather station and restarting the FOSHKplugin service (default: False = forbidden) on request requests are http://ipaddress:port/FOSHKplugin/rebootWS to reboot the configured weather station
  and http://ipaddress:port/FOSHKplugin/restartPlugin to restart the FOSHKplugin service
- New forward type RAWTEXT enables the storage of incoming values from the weather station as a text file locally in the file system and remotely via http(s)/POST and ftp(s)
- Firmware update status (updatewarning) is now updated immediately after receipt of new data from the updated weather station
- slightly improved installation routine generic-FOSHKplugin-install.sh - clarification of UDP server and port, interception of incorrect URL file names during update
- Adjustment for WH6006 (dateutc with "%3A" instead of ":", indoorhumidity)

Version 0.07 - 19.02.2021

- Fixed bug with IGNORE_EMPTY: UDP sending to Loxone did not work if IGNORE_EMPTY was deactivated
- Log output: "custom mode" renamed to "custom server"
- now sends the text OK to the sender in addition to the response code 200 when the data is received via http
- Bugfix: known issue regarding socket problems and Chrome hopefully solved - the socket should be released again after a 5 seconds timeout
- Troubleshooting thunderstorm warning (not every thunderstorm was reported)
- Bugfix: Program error with PM2.5 values above 500 fixed
- Bugfix: Handling of %20 in the dateutc field (e.g. from the WH2600 LAN)
- Config parsing made more robust with regard to Boolean values (mkBoolean)
- Forwards can now be activated/deactivated (FWD_ENABLE = True/False) and commented (FWD_CMT) in the config file
- Multi-instance: several instances of FOSHKplugin can now be operated in parallel - in different directories
- Support of the Ambient Weather format for incoming messages as well as forward (AMB/RAWAMB)
  in the absence of yearlyrainin, totalrainin is used and in the absence of rainratein, hourlyrainin is used
- Forward input data in Weathercloud format via GET as type WC possible
- Forward input data in Meteotemplate format via GET as type MT possible
- Forward input data in Awekas format via GET as type AWEKAS possible
- support of WH45 (PM25, PM10, CO2 sensor) with additional AQI and CO2-level calculation

- Preparation for the new leaf wetness sensor WN35
- Thunderstorm warning: number of flashes (lcount) and min. and max. Distance (ldmin and ldmax) are transmitted
- Improvement with regard to TimeOut behavior; http now has a TimeOut of 8 and UDP of 3 seconds
- Ecowitt-Forward: if totalrain is available - but no yearlyrain, yearlyrain is automatically set with the value of totalrain
- New configuration option Export\OUT_TIME = True sets the time stamp of incoming messages from the weather station to the time of receipt
- fake mode now also activated for incoming messages in WU and Ambient Weather format
- an automatic restart in the event of missing data from the weather station can be configured via Warning\WSDOG_RESTART
- important status messages can now also be sent as push notification via pushover (update, sensor, watchdog, battery, storm and thunderstorm warnings)
- generic: display of all detected weather stations during installation via generic-FOSHKplugin-install.sh (-scanWS)
- Average value calculation (with EVAL_DATA = True) implemented for wind and wind direction, outputs windspdmph_avg10m, winddir_avg10m
- max gust calculation (with EVAL_DATA = True) implemented: windgustkmh_max10m
- Output of the lux value (solarradiation * 126.7) as field brightness (with EVAL_DATA = True)
- with Logging\IGNORE_LOG you can exclude lines from the logging in the standard log (comma-separated list of search words) - for example *crondaemon*
- With FWD_EXEC, a script can be specified for each forward that starts with the output string as a parameter and the last output line is used as the new output string for sending
- Value request: an http request http://ipaddress:portnumber/getvalue?key=[key] outputs the value for the key [key], whereby the key may be the RAW key as well as the converted key name
  Example: curl http://192.168.15.236:8080/getvalue?key=windspeedmph outputs the value "1.34" of the wind speed mph key
  in connection with FWD_EXEC, data from other instances can be queried and integrated
- In addition to the http-GET query /CSV and /CSVHDR with dynamic fields, /SSV and /SSVHDR with static fields are now also supported, based on the field description in CSV\CSV_FIELDS thus the order of the fields is fixed; empty fields (with no content e.g. in the event of a sensor failure) are not skipped but are output as ""
- If Export \FIX_LIGHTNING is active (default), the last known values for lightning and lightning_time will be used as input data if the incoming values are missing
  since the lightning data in the GW1000/DP1500 are not stored in the NVRAM, these values are lost when the device is restarted
- If Warning\LEAKAGE_WARNING is activated, warnings are issued via log, UDP, http or pushover (leakwarning) when a leak is detected
- a backup of foshkplugin.conf is created at every successful start (foshkplugin.conf.foshkbackup)
- Overview page http://ipaddress:portnumber outputs with "?units=e" values in American system of units (inch, mph, ...) ; can be combined with "/status"

Version 0.06 - 02.08.2020

- Revision of the storm warning function, output of air pressure trend 1h / 3h and value of change of air pressure 1h / 3h
- WU-Forward from AqPM2.5 if PM-sensor is present (only pm25_ch1 is forwarded!)
- AQI calculation activated with EVAL_DATA = True and existing DP200/WH41/43
- Experimental: Forward the PM2.5 value to luftdaten.info as type LD, the ID must be entered

under FWD_SID in the config file
- Battery warning via log and UDP implemented; if the supplied batt value falls below an internally defined threshold, a warning is issued
- Fixed issue with output of the name of the sensor again supplying data (SENSOR_MANDATORY)
- The status of warnings for storms, thunderstorms, sensors and batteries are stored temporarily and are therefore retentive
- WU-Forward / JSON renaming from solar radiation to solar radiation
- WU-Forward / JSON support for soil moisture sensors
- WU-Forward: keys with empty value are not transmitted
- WU-Forward: Upload of dewptf (was dewpt) and rainin (was rainratein) fixed
- new formula for dew point calculation (dewpoint) active (requires math)
- now sends a response code 200 to the sender when http data is received
- UDP message for time at wswarning changed from "time:" to "time="
- for all get / post actions: check the return value 200..202 → ok (was 200 only)
- Text errors in help fixed
- Update of generic-FOSHKplugin-install.sh; Fixed a bug when creating the conf file
- Forward of the input data possible without conversion via UDP as type RAWUDP
- Forward of the input data possible without conversion via EW / POST as type RAWEW
- Forward of output data in CSV-format via http/POST as type CSV
- Forward of input data without conversion via http/POST as type RAWCSV
- Forward of output data as for Loxone to another target/network via UDP - with additional status
- Forward timeout handling adjusted (now 3 seconds)
- Output language can be set via LANGUAGE = DE / EN etc. in the config file
- Plugin can be terminated with a UDP command and requested to send the current status (System.shutdown, Plugin.getStatus)
- debug mode enable/disable via UDP-command Plugin.debug=enable/disable
- Separator selectable on http-GET with /RAW
- new http-GET command /STRING to get the input-line via http with selectable separator
- status messages can now be queried in /JSON and when outputting via /STRING or /UDP
- simple authentication via AUTH_PWD implemented; data and requests are only accepted via http if the specified password is contained in the URL (hint: use the PASSKEY-string in Ecowitt-mode)
- hide PASSKEY-value in Log-Files if AUTH_PWD activated
- Handling of unnecessary quotation marks in the config file adjusted
- Preparation for upcoming soil/water temperature sensor WH34 (tf_chNc, tf_battN - where N = 1..8)
- Status also available via http/GET: http://ipaddress:portnumber/FOSHKplugin/status outputs status wswarning, sensorwarning, batterywarning, ...
- Fake mode implemented: Values of an indoor sensor (WH31 / DP50) can be output as values of an outdoor sensor WH32 (temperature, air humidity)
- Updatewarning implemented, reports an available update for the weather station via log/UDP and possibly via http
- For thunderstorm warnings, tswarning is now output as status instead of tstormwarning - **Attention! This affects all outputs both via UDP and via http!**

Version 0.05 - 26.04.2020

- Sturmwarnung bleibt für 60 Minuten nach letzter Grenzwertunter-/überschreitung aktiv; Zeitraum kann via STORM_EXPIRE im Config-File angepasst werden
- Übermittlung des UV-Wertes im WU-Format angepasst, nun in Großbuchstaben UV= statt uv=
- Patch-Funktion für Weather4Loxone ist nun unabhängig von der genutzten Weather4Loxone-

Version (vorhandene fetch.pl wird nicht überschrieben sondern angepasst)

Version 0.04 - 20.02.2020

- default-config angepasst - Kommentare hinter Block nicht zulässig!
- verbesserte Buttons (CSS) - Schiebeschalter nun grau bei "off" und grün bei "on"
- erweiterte CGI-Debug-Funktion; default: off; enable mit $myDebug = 1 in der index.cgi
- myDebug für zusätzliche Debug-Informationen auch im Python-Programm implementiert (default: False)
- Beschreiben der Wetterstation via WS-Set sollte nun (endlich) vollumfänglich funktionieren
- Id & Key in den Einstellungen der Wetterstation werden ignoriert und nicht vom Plugin überschrieben

Version 0.03 - 18.01.2020

- USE_METRIC wieder funktional (jetzt also auch imperiale Werte per UDP und CSV möglich)
- weitere mögliche Probleme beim Setzen der Wetterstationsparameter via WS-Set behoben (Path wird nun immer auf defaults gesetzt)
- Prüfung der nutzbaren LoxBerry-Ports (http/udp) optimiert
- Kommunikation mit der Wetterstation überarbeitet - nun jeweils 5 Versuche bei Lesen und Schreiben
- besseres Logging/Debugging bei Fehlern bei Set-WS; "buntere" und besser parse-bare Log-Files; ### entfernt
- generic: conf-File - Vorlage und Hilfstexte überarbeitet
- Ignorierliste Forward\FWD_IGNORE für Forwards eingebaut: definiert - kommasepariert - Felder, die NICHT verschickt werden sollen
- Forward\FWD_TYPE=UDP/EW/RAW für http-Forward der Werte (UDP-Ausgabezeile) an andere Ziele als WU eingeführt
- nun bis zu 10 Forwards mit unterschiedlichen Einstellungen möglich (aktuell nur im Config-File zu pflegen: Forward-1..9 analog zu Forward)
- Watchdog: kommen seit 3*eingestelltem Intervall keine Werte von der Wetterstation, Fehler melden!
  es erfolgt EINE Warnung und bei erneuter Übermittlung der Wetterstation eine Entwarnung im Log sowie per UDP:
  SID=FOSHKweather wswarning=1 last=346611722
  SID=FOSHKweather wswarning=0 last=346616459
  standardmäßig aktiv; kann im Config-File deaktiviert werden: Warning\WSDOG_WARNING=False
  Intervall kann im Config-File eingestellt werden: Warning\WSDOG_INTERVAL=3
  Warnung auch in Loxone-Vorlage enthalten
- Alarm senden (Log, UDP) wenn Sensor (auch mehrere) keine Daten liefert (etwa weil Akku/Batterie leer)
  SID=FOSHKweather sensorwarning=1 missed=wh65batt time=347196201
  SID=FOSHKweather sensorwarning=1 back=wh65batt time=347196201
  aktuell nur im Config-File zu pflegen:
  Warning\SENSOR_WARNING=True sowie Warning\SENSOR_MANDATORY="wh65batt"
- Sturmwarnung: fällt oder steigt der Luftdruck um mehr als 1.75 Hektopascal in einer Stunde, erfolgt eine Warnung vor Starkwind/Sturm
  vgl. http://www.bohlken.net/luftdruck2.htm
  es erfolgt EINE Warnung und bei Entspannung des Luftdrucks eine Entwarnung im Log und per UDP:
  SID=FOSHKweather stormwarning=1 time=346611722

SID=FOSHKweather stormwarning=0 time=346616459
standardmäßig aktiv; kann im Config-File deaktiviert werden: Warning\STORM_WARNING=False
WarnDiff kann im Config-File eingestellt werden: Warning\STORM_WARNDIFF=1.75
Warnung auch in Loxone-Vorlage enthalten
- Vorbereitung Wassersensor WH55 und Blitzsensor WH57 (noch unklar ob lightning_time = timestring oder unixtime!)
- preupgrade-Script: Upgrade-Verzeichnisse werden nun auch ohne Elternverzeichnis angelegt (mkdir -p)
- preuninstall-script entfernt; Deinstallation erfolgt bei LoxBerry ab v2.0.1.1 im uninstall-Script
- Web-Oberfläche: Anzeige der Versionsnummer eingebaut (um Nachfragen zur verwendeten Version im Fehlerfall zu minimieren)
- UDP-Versand an das Zielsystem lässt sich mit UDP_ENABLE=False abschalten
- Ignorierliste für den UDP-Versand eingeführt: Config\UDP_IGNORE (nur im Config-File zu pflegen)

Version 0.02 - 28.12.2019

- ### aus FWD-Log-Nachricht entfernt
- Umrechnung temp1f in temp1c für Innensensor auf Kanal 1 implementiert
- Timeout bei sendReboot, setWSconfig und getWSINTERVAL von 1 auf 2 Sekunden erhöht (somit sollte WS-Set sicherer funktionieren)
- Probleme beim Setzen der Wetterstationsparameter via WS-Set behoben (Id & Key werden - wenn nicht schon vorhanden - gesetzt)
- Umstellung der LoxBerry-Versionsnummerierung damit zukünftig die Auto-Update-Funktion greifen kann

Version 0.01 - 15.12.2019

- erste öffentliche Version

Features:

- accepts http messages from a weather station (GW1000, GW1100, GW2000, HP2551, HP2560, WN19x0, DP1500, HP1000SE Pro, Sainlogic 7 in 1, ELV WS980WiFi, Eurochron EFWS 2900, Ambient Weather WS-2902, WS-2000, WS-5000, ???) locally in WU or Ecowitt protocol via network
- supports custom server on stations from Ambient Weather
- does not require cloud services or internet connection
- sends the converted metric or imperial values via UDP to any host or via broadcast in the network
- is able to feed a MQTT broker
- connection to any database system possible via telegraf
- direct support of InfluxDB
- saves the converted or imperial data sorted and / or extracted as CSV
- enables forwarding to up to 100 servers that are not supported by the weather station itself (e.g. Awekas, PWSWeather, Windy or Luftdaten.info, but you could also use WU in a different interval)
- can feed Meteotemplate, Weathercloud, wetter.com and weather365.net
- may export incoming data to realtime.txt and clientraw.txt
- can serve as an Ecowitt relay (forward in Ecowitt protocol) for Personal Weather Tablet, weewx,

PWS Dashboard and any other program expecting Ecowitt-data
- can forward incoming WU and Ecowitt messages via UDP - also as a broadcast - as they come in
- is able to convert between WU, Ecowitt and Ambient Weather format (within limits)
- can answer queries in WU protocol
- Integrated web server provides the last data record in http, UDP, CSV, RAW and JSON format as well as a simple website
- various watchdogs and warnings can be configured (battery, connection weather station and sensors, storm, thunderstorm, CO2-alert, leakage-alert ...)
- calculates some extra data (dew point, feelslike, AQI, ...)
- allows to gather specific values via http (getvalue)
- it creates an import file for the automatic import of the data into WSWin
- provides the Weather4Loxone plugin with the measured values from local weather station
- No additional software is required (WS View only for teaching new sensors or for configuring the standard forwarding services)
- also works without Loxone / LoxBerry as a systemd service on Linux-systems (a Raspi should be powerful enough) for connecting other systems (generic-FOSHKplugin.zip)
- is free of charge

The target system (e.g. Loxone Miniserver) hardly needs resources with this solution; it does not have to fetch any data or convert values - the plugin automatically sends the already converted data to the Miniserver whenever new measured values arrive from the weather station.
In addition, the measured and partly calculated values are also available to any other services via various interfaces and forwards.

Operation:
FOSHKplugin acts as a web server and returns different values depending on the requested URL.
In addition to "updateweatherstation" to accept a incoming data record in WU format (Weather Underground protocol) the integrated web server processes other http call parameters in
GET: http://serverip:port/[URLpath]

URLpath	description
/CSVHDR	the field names (the header) of the last data record are output as CSV semicolon separated. If units=e is also specified, the fields for the imperial values are output.
/CSV	all reported metric values of the last data record are output as CSV semicolon separated (units=e supplies the imperial values)
/SSVHDR	the field names (the header) configured in CSV\CSV_FIELDS are output as CSV semicolon separated.
/SSV	all reported metric values of the last data record are output as CSV semicolon separated with fixed asignment based on CSV\CSV_FIELDS (units=e supplies the imperial values)
/UDP	the last UDP string is output via http; with additional ?status in URL the output will also include all status.
/RAW	the data set supplied by the weather station is output unchanged via http; separator can be changed with separator=Z, where Z is a single character
/STRING	output the converted data record and the current status separated with ";" via http; by adding units=e in the URL will output with the imp. values; separator can be changed with separator=Z, where Z is a single character. With additional ?status in URL the output will also include all status. example: http://ipadresse:port/STRING?units=e?separator=, will output imp. values with comma as separator
/JSON	output via http as JSON (metric by default; by adding units=e in the URL, the output is made with the imp. values). With additional ?status in URL the output will also include all status (wswarning, sensorwarning, stormwarning, ...).
/realtime.txt	output a realtime.txt (Cumulus) file
/clientraw.txt	output a clientraw.txt (Weather Display) file
/getvalue?key=[keyname]	output the value only for given keyname; any keyname is allowed (RAW, converted); if keyname not found "" wil be output - usefull for processing values via curl or wget - if you need bool values like True/False instead of numerical values 1/0 add a *&bool* to the URL
/	simple website with the current metric data in tabular form - status includes all status-messages; additional units=e shows all values in imperial
/FOSHKplugin/state	status of the service; if active: "running"
/FOSHKplugin/status	status of the service, watchdog, missing sensor, battery, for storm, thunderstorm, ... as a simple webpage
/FOSHKplugin/help	help screen with all possibilities to access
/FOSHKplugin/keyhelp	shows all available keys known to FOSHKplugin
/FOSHKplugin/debug=enable	enable debug mode for extended messages in the log file
/FOSHKplugin/debug=disable	disable debug mode for extended messages in the log file

/FOSHKplugin/pushover=enable	temporarily enable push notification via Pushover (configuration must exist!)
/FOSHKplugin/pushover=disable	temporarily disable push notification via Pushover
/FOSHKplugin/patchW4L	"Patch" a Weather4Loxone installation (copy local grabber scripts and activate local retrieval by W4L)
/FOSHKplugin/recoverW4L	Restore the original Weather4Loxone configuration before "patching"
/FOSHKplugin/rebootWS	reboot the connected weather station (Config\REBOOT_ENABLE has to be True)
/FOSHKplugin/restartPlugin	restart the running FOSHKplugin service (Config\RESTART_ENABLE has to be True)
/observations/current/json/units=m	Feedback of a WU-compatible data record with metric values (°C, kmh, mm, hPa)
/observations/current/json/units=e	Feedback of the WU-compatible data record with imperial values (°F, mph, in, inHg)
/w4l/current.dat	Feedback of a W4L-compatible current.dat:\\ 1617139459|Tue, 30 Mar 2021 23:24:19 +0200|CET|Europe/Berlin|+0200|Hohen Neuendorf||DE|52.669481|13.266531|53|8.2|8.2|92|Westsüdwest|256|0.0|0.0|8.2|1027.09|7.0||0.00|7.5|0|0.0|0.0||||||||||||||

If you have defined a PASSKEY in config-file to only accept incoming http-request containing this PASSKEY you always have to add something like **?auth=[PASSKEY]** to the URL. The only exception is the call of FOSHKplugin/state - this works even without authentication. This is useful if FOSHKplugin should not work in a secure local network but directly on the Internet - for example on a root server. Without this security mechanism, anyone could otherwise submit data or query or change states. Basically, however, I advise against operating on "unsafe" hosts that are freely available on the Internet.

In POST mode, weather station data is accepted in Ecowitt format if the keyword "report" is contained in the URL. Since in Ecowitt format significantly more values can be transmitted from the weather station (such as the battery values of the sensors), I recommend this operating mode (which is also set by the plugin for WS-Set).

Download:
generic-FOSHKplugin.zip (current stable version v0.09)
public beta test: FOSHKplugin v0.10

Installation:
create a local directory
sudo mkdir /opt/FOSHKplugin
#
change into the created directory
cd /opt/FOSHKplugin
#
get the current version of the plugin via wget
wget -N http://foshkplugin.phantasoft.de/files/generic-FOSHKplugin.zip
or use a local zip-file
#
unpack ZIP-File
unzip generic-FOSHKplugin.zip
#
Allow execute right for generic-FOSHKplugin-install.sh (this script)
chmod u+x generic-FOSHKplugin-install.sh
#
Run generic-FOSHKplugin-install.sh (this script)
sudo ./generic-FOSHKplugin-install.sh –install

Configuration options

metric units - USE_METRIC:
If activated, the values for UDP dispatch and CSV export delivered by the weather station in US units

are converted directly by the plugin

skip empty values - IGNORE_EMPTY:
When activated, values -9999 coming from the weather station may not be sent to the target machine via UDP

use Loxone time - LOX_TIME:
This switch determines whether the UTC time should be converted to the Loxone time. When activated, an additional field loxtime is added in Loxone-compatible time format (seconds since 01/01/2009).

optional calculations - EVAL_VALUES:
When activated, the values for dew point, wind chill temperature, heat index and perceived temperature and - if a particulate matter sensor DP200 / WH41 / WH43 is available - the current AQI value and its 24-hour average are calculated from the available measured values and those coming from the weather station Data for export processing (UDP, WU, CSV, W4L, ...) added. Values already coming from the weather station may NOT be overwritten. If the storm warning is activated, the air pressure trend and the change in air pressure are also calculated (for the last hour and for the last 3 hours).

optional elements - ADD_ITEMS:
appends a string with static values to the raw data line coming from the weather station; any existing variable names with the same name are overwritten. Useful to pass on a few fields (such as geolocation: lat / lon / elev or location: neighborhood) via UDP / WU / CSV / W4L etc. These fields go through the entire export processing and therefore appear in all output formats - except RAW-exports of course. This function can also be used to exclude values from the weather station from further processing. To do this, an empty value must be assigned to a variable (ie: &variable3=&variable4=value4) and "skip empty values" must be activated.
Format: &variable1=value1&variable2=value2

Log files:
Are very useful for commissioning and in the event of problems. However, you should consider whether it is really sensible to keep a permanent log. If an SD card is used as the storage medium, the SD card will eventually be written down.
Especially the export log - if a very short interval is set - can quickly become very large, because for every message coming from the weather station - depending on the configuration - an entry for UDP, forwarding (FWD) and CSV is also generated.

To deactivate a specific log file, simply remove the name of the respective file.
It is possible to switch logging on and off globally via the switch in the config file Logging\LOG_ENABLE = True/False (default: True).

Forwards - Forward-1..99:
There is only one external destination for sending via "Customized Upload" in the configuration of a weather station. Since we are already using this for FOSHKplugin, you can set a forwarding to an additional service (such as Awekas) here. The plugin currently supports 100 forwarding destinations via the config file.

From v0.07 you can enable/disable a Forward through FWD_ENABLE = True/False. In this way, all settings are retained, even if the forward is not currently required. FWD_ENABLE = False completely deactivates this forward until it is activated again with FWD_ENABLE = True.
For your own notes (for example what this forward is actually intended for) there is FWD_CMT, in

which any string can be stored permanently after the equal sign.

When specifying the URL, please note that only the measured values are added by the plugin. Any authentications or update commands must therefore already be entered here.
For an upload to Weather Underground (which is of course also possible directly via the weather station), such a line would look like this:

https://rtupdate.wunderground.com/weatherstation/updateweatherstation.php?ID=[meine ID]&PASSWORD=[my password]&action=updateraw&

I successfully tested the delivery to the services Awekas, Windy and PWSWeather:

URL for Awekas:
http://ws.awekas.at/weatherstation/updateweatherstation.php?ID=[awekasid]&PASSWORD=[awekaspassword]&
(https should also work)

URL for Windy:
https://stations.windy.com/pws/update/[windyAPIkey]?

URL for PWSWeather:
http://www.pwsweather.com/pwsupdate/pwsupdate.php?ID=[PWS-ID]&PASSWORD=[PWS-Password]&
(shortly:
https://pwsupdate.pwsweather.com/api/v1/submitwx?ID=[PWS-ID]&PASSWORD=[PWS-Password]&)

Other WU-compatible services should also work.
If the field remains free, no forwarding takes place.

"FWD_TYPE" defines the format in which the forwarded messages are to be sent to the weather station.
The WU format should be selected for WU-compatible servers. For other scenarios there is also the UDPGET format, in which the possibly converted metric values are sent as with UDP (but not separated by spaces but by html-conforming "&"). Virtual http inputs should be possible with this.
The EW format is experimental. Incoming messages from the weather station are converted into the Ecowitt format and forwarded in the Ecowitt protocol via HTTP mail. This means that other hosts can also be operated using the Ecowitt protocol (relay).
With type RAW, the incoming data is forwarded via http-get without conversion. In order to send the original RAW string via POST without any extension in the EW format, the RAWEW type is recommended. The RAW data can also be sent via UDP via RAWUDP. Destination-ip: destination-port must then be specified as FWD_URL. If you need to send the output data to another destination via UDP you may use the FWD_TYPE UDP. The destination address and port are defined via destination-ip: port as FWD_URL.
The values of an existing particulate matter sensor DP200 / WH41 required for the luftdaten.info service can be sent via type LD:

URL for Luftdaten:
https://api.sensor.community/v1/push-sensor-data/

The sensor ID required for registration must be entered in the config file under FWD_SID. The interval for sending the particulate matter sensor values should be configured to 150 seconds (FWD_INTERVAL = 150 in the config file). In addition to the PM2.5 value, the service also expects the PM10 value (which the DP200 / WH41 particulate matter sensor cannot deliver). Therefore, the plugin sends a dummy value of 1 for PM10.

Overview of the different forward options:

FWD_TYPE	input-Format	out-Transport	out-Format
WU	WU, EW, AMB	GET	Weather Underground (WU–>WU or EW–>WU)
RAW	WU, EW, AMB	GET	like input (WU–>WU or EW–>EW)
UDPGET	WU, EW, AMB	GET	like output to Loxone with header and possibly conversion, however, URL-compatible with "&" instead of spaces
WC	WU, EW, AMB	GET	Weathercloud
MT	WU, EW, AMB	GET	Meteotemplate (API)
AMB	WU, EW, AMB	GET	Ambient Weather
AWEKAS	WU, EW, AMB	GET	Awekas (API)
WETTERCOM	WU, EW, AMB	GET	Wetterarchiv/wetter.com (API)
EW	WU, EW, AMB	POST	enhanced Ecowitt (WU–>EW or EW–>EW)
RAWEW	WU, EW, AMB	POST	untouched Ecowitt (EW–>EW or WU–>EW)
LD	WU, EW, AMB	POST	Luftdaten.info-Format (only PM2.5, PM10, Temp, Humidity, rel. Pressure, abs. Presssure)
CSV	WU, EW, AMB	POST	like output to Loxone with possibly conversion, with semicolon as separator instead of spaces but without header
RAWCSV	WU, EW, AMB	POST	like input (WU–>WU or EW–>EW), with semicolon as separator instead of spaces but without header
WEATHER365	WU, EW, AMB	POST	weather365.net API
UDP	WU, EW, AMB	UDP	like output to Loxone with header and possibly conversion via UDP (FWD_URL = destination:port)
RAWUDP	WU, EW, AMB	UDP	like input-Format but transmission via UDP (EW→EW via UDP or WU→WU via UDP)
REALTIMETXT	WU, EW, AMB	various	sends a realtime.txt (Cumulus export file) via http(s)/POST or ftp(s) to remote destinations or save the file locally in the filesystem
CLIENTRAWTXT	WU, EW, AMB	various	sends a clientraw.txt (Weather Display export file) via http(s)/POST or ftp(s) to remote destinations or save the file locally in the filesystem
CSVFILE	WU, EW, AMB	various	sends a CSV-file with current data only via http(s)/POST or ftp(s) to remote destinations or save the file locally in the filesystem
TXTFILE	WU, EW, AMB	various	sends a TXT-file with current data only via http(s)/POST or ftp(s) to remote destinations or save the file locally in the filesystem
RAWTEXT	WU, EW, AMB	various	sends a TXT-file with current raw data only via http(s)/POST or ftp(s) to remote destinations or save the file locally in the filesystem
MQTTMET	WU, EW, AMB	MQTT	sends incoming data in metric units to a MQTT-broker
MQTTIMP	WU, EW, AMB	MQTT	sends incoming data in imperial units to a MQTT-broker
WSWIN	WU, EW, AMB	File	saves a WSWin-compatible wswin.csv in the file system, which can be read in automatically by WSWin via file monitoring
INFLUXMET	WU, EW, AMB	InfluxDB	sends metric values to a InfluxDB database
INFLUXIMP	WU, EW, AMB	InfluxDB	sends imperial values to a InfluxDB database

Data fields from the ignore list maintained under "FWD_IGNORE" are not sent for the relevant forward.

With "FWD_INTERVAL" an interval independent of the weather station can be configured (in seconds). If this field is left blank, it will be sent at the weather station's send interval.

Save as CSV:
The measurement results can also be saved as a comma-separated file (CSV). The storage location and the file name are specified under CSV_NAME. The problem with writing to SD cards already mentioned for log files also applies here. If necessary, a more suitable medium (such as NFS) should be selected here.

Field names for CSV - CSV_FIELDS:
All fields desired in the CSV are listed under CSV_FIELDS - separated by a separator (semicolon, comma or space).
Not all fields of a data record are worth saving in the CSV. The contents of the fields SID, PASSKEY, freq or model change very rarely.
By omitting these field names, these fields are excluded from storage. The order of the columns in the CSV file results from the order of the fields specified here.

CSV interval- CSV_INTERVAL:
Here you can define your own time interval for storing a data record in the CSV. If the field remains free, the transmission interval of the weather station is used.

The interval for the CSV and forwarding function cannot be smaller than the set transmission interval of the weather station, since data is only available for further processing when a data record is received from the weather station.

watchdog & warn-functions:

type	config	description
report watchdog	WSDOG_WARNING	will warn if weather station did not report within 3 send-intervals (configurable)
sensor warning	SENSOR_WARNING	will warn if data for mandatory sensor (configurable list of fields e.g. wh65batt) is missed
battery warning	BATTERY_WARNING	will warn if battery level of all known sensors is critical (pre-defined)
storm warning	STORM_WARNING	will warn if air pressure rises/drops more than 1.75 hPa/hour or 3.75hPa/3hr with expiry time of 60 minutes (all values configurable)
thunderstorm warning	TSTORM_WARNING	will warn if lightning sensor WH57/DP60 present, count of lightnings is more than TSTORM_WARNCOUNT and distance is less or equal TSTORM_WARNDIST with expiry time of TSTORM_EXPIRE minutes (all values configurable)
firmware-update warning	UPD_CHECK	will warn if there's a new firmware for the weather station available
leakage warning	LEAK_WARNING	will warn if any WH55 reports a leak
CO2 warning	CO2_WARNING	reports an alarm when a WH45 reports a value above CO2_WARNLEVEL

These warnings are issued - depending on the configuration - in the standard log file, via UDP and via pushover and can be queried via http (?status).

The rules for BATTERY_WARNING are predefined and not adjustable. A warning will be triggered if:
key = [wh65batt, lowbatt, wh26batt, wh25batt] and value = 1
key = batt and length(key) = 5 and value = 1
key = [wh57batt, pm25batt, leakbatt, co2_batt] and value < 2
key = [soilbatt, wh40batt * , wh68batt, tf_batt] and value ⇐ 1.2 * with current hardware revision of WH40 there's no wh40batt at all
key = wh80batt and value < 2.3

and - if recognized as a Ambient Weather station:
isAmbientWeather and (batt in key or batleak in key) and value = 0

Fake-Mode
The values for outside temperature and humidity normally come from either a combination sensor (WH65, WS80) or the dedicated outside sensor WH32.
However, if neither a combination sensor nor a WH32 is available, values of any internal sensor DP50/WH31 (which should of course then be installed outside with appropriate weather protection) can be output as values of the external sensor.
In the config file you have to specify which key should be used for the respective value:

[Export] OUT_TEMP=temp1f OUT_HUM=humidity1	# exchange the keyname temp1f with tempf (or use temp2f, temp3f, ...) # exchange the keyname humidity1 with humidity (or use humidity2, humidity3, ...)

Within FOSHKplugin, the substring "&temp1f =" is simply replaced by "&tempf =" and "&humidity1 =" by "&humidity =" when the Ecowitt line arrives from the weather station. The values themselves remain.
This setting is global and therefore affects all FOSHKplugin exports/forwards/outputs (except for RAW and RAWEW).
The weather station itself, of course, knows nothing of this - so the services configured there (Ecowitt, WU, WOW, etc.) still have no external values.
If you also want to output the values of the indoor sensor as outdoor sensor values for these services, the services within the weather station must be deactivated and carried out by FOSHKplugin instead.
Corresponding forwards must then be defined in the config file (the square brackets must **not** be included):

```
#WU:
[Forward-11]
FWD_INTERVAL = 300
FWD_URL =
https://rtupdate.wunderground.com/weatherstation/updateweatherstation.php?ID=[WU-ID]&PASSWORD=[WU-Password]&action=updateraw&
FWD_TYPE = WU
```

```
#EW:
[Forward-12]
FWD_INTERVAL = 300
FWD_URL = http://cdnrtpdate.ecowitt.net/data/report/
FWD_TYPE = EW
```

```
#Ambient Weather:
[Forward-13]
FWD_TYPE = RAWAMB
FWD_CMT = Forward for Ambient Weather
FWD_URL = https://api.ambientweather.net/endpoint?
FWD_ENABLE = True
FWD_STATUS = False
FWD_INTERVAL = 180
```

```
#WOW:
[Forward-14]
FWD_INTERVAL = 300
FWD_URL = http://wow.metoffice.gov.uk/automaticreading?siteid=[siteid]&siteAuthenticationKey=[siteAuthenticationKey]&
FWD_TYPE = WU
```

```
#Weathercloud:
[Forward-15]
FWD_INTERVAL = 300
FWD_URL = http://api.weathercloud.net/v01/set?wid=[weathercloudid]&key=[key]&
FWD_TYPE = WC
```

```
#Meteotemplate:
[Forward-16]
FWD_INTERVAL = 60        # should be shorter than 5 minutes
FWD_URL = http://192.168.15.100/template/api.php?PASS=[meteotemplatepwd]&
FWD_TYPE = MT
```

```
#Awekas-API:
[Forward-17]
FWD_CMT = forward im Awekas-API-Format
FWD_TYPE = AWEKAS
FWD_URL = http://data.awekas.at/eingabe_pruefung.php?
FWD_SID = Awekas-ID
FWD_PWD = Awekas-password
FWD_INTERVAL = 60
FWD_ENABLE = True
```

```
#wetter.com-API:
[Forward-18]
FWD_CMT = wetter.com via API
FWD_TYPE = WETTERCOM
FWD_URL = http://interface.wetterarchiv.de/weather
FWD_SID = station-ID
FWD_PWD = station-password
FWD_INTERVAL = 300
FWD_ENABLE = True
```

```
#weather365-API:
[Forward-19]
FWD_CMT = weather365 via API
FWD_TYPE = WEATHER365
FWD_URL = https://channel1.weather365.net/stations/index.php
FWD_SID = station-ID
FWD_INTERVAL = 300
FWD_ENABLE = True
```

```
#realtime.txt:
[Forward-20]
FWD_CMT = export realtime.txt via ftp
FWD_TYPE = REALTIMETXT
FWD_URL = ftp://anydomain.de/httpdocs/mydomain.phantasoft.de/upload # use a path name for saving the file in the file system
FWD_SID = ftp-username
FWD_PWD = ftp-password
FWD_INTERVAL = 30
FWD_ENABLE = True
```

```
#clientraw.txt:
[Forward-21]
FWD_CMT = export clientraw.txt to filesystem
FWD_TYPE = CLIENTRAWTXT
FWD_URL = /some/path/anyhwere/in/filesystem/
FWD_INTERVAL = 30
FWD_ENABLE = True
```

```
#Wettersektor:
[Forward-22]
FWD_CMT = send data to wettersektor.de via POST
FWD_TYPE = WETTERSEKTOR
FWD_URL = http://wettersektor.de/getwett.php
FWD_SID = username
FWD_PWD = password
FWD_INTERVAL = 60
FWD_ENABLE = True
```

```
#WSWin-Export
[Forward-23]
FWD_CMT = WSWin forward
FWD_TYPE = WSWIN
FWD_URL = /opt/loxberry/log/plugins/foshkplugin/ # specify only the path here - the file name is always wswin.csv
FWD_INTERVAL = 60
FWD_ENABLE = True
```

```
#InfluxDB metric export (use INFLUXIMP for imperial values)
[Forward-24]
FWD_CMT = InfluxDB metric forward
FWD_TYPE = INFLUXMET
FWD_URL = http://server:port@database # use https for SSL connection
FWD_SID = username
FWD_PWD = password
FWD_INTERVAL = 60
FWD_ENABLE = True
```

```
#send to Windy in WU format
[Forward-25]
FWD_CMT = Windy
FWD_TYPE = WU
FWD_URL = https://stations.windy.com/pws/update/[windyAPIkey]?
FWD_INTERVAL = 300
FWD_ENABLE = True
```

Configuration-file:

```
[Config]
LOX_IP =                        # Loxone-IP or broadcast-address to send
data to
LOX_PORT = 12340                # Loxone-Port - port to send data to
LB_IP =                         # Loxberry-IP
LBU_PORT = 12341                # Loxberry-Port - port to receive UDP-
datagrams
LBH_PORT =                      # Loxberry-Port - port to receive HTML-in &
out
LOX_TIME = True                 # adjust time base to 01.01.2009 (Loxone
only)
USE_METRIC = True               # use metric instead of imperial values
IGNORE_EMPTY = True             # do not send -9999 or empty values
UDP_ENABLE = True               # set to False to disable UDP-sending
UDP_IGNORE =                    # comma-separated list of fields to not send
via UDP
UDP_MAXLEN =                    # defines the length of a UDP datagram
before deviding to several datagrams (default 2000)
UDP_STATRESEND = 0              # cycle sending of warnings via UDP in
seconds
LANGUAGE = DE                   # remove or adjust to EN to use english
output for wprogtxt and wnowtxt (or NL/SK/FR/ES)
AUTH_PWD =                      # if set, only incoming & outgoing http-
requests containing this passphrase will be accepted
DEF_SID =                       # override the default identifier for
outgoing UDP datagrams (default: FOSHKweather)
REBOOT_ENABLE = False           # enable remote reboot of weather station
via UDP and http (default: False - danger!)
RESTART_ENABLE = False          # enable restarting FOSHKplugin via UDP and
http (default: False - danger!)

[Weatherstation]
WS_IP =                         # IP-address of weather station
WS_PORT =                       # UDP-port of weather station
WS_INTERVAL = 60                # weather station will send data every n
```

```
seconds (16..3600)

[Export]
EVAL_VALUES = True              # calculate some extra values on base of
current data (dew point, windchill, heatIndex, feelsliketemp, AQI, ...)
ADD_ITEMS =                     # additional fixed strings to append to
every raw-data-line
OUT_TEMP =                      # fake the temperature-value for outdoor-
sensor with value of specific indoor-sensor e.g. temp1f
OUT_HUM =                       # fake the humidity-value for outdoor-sensor
with value of specific indoor-sensor e.g. humidity1
OUT_TIME = False                # exchange incoming time string with time of
receiving
FIX_LIGHTNING = True            # use last known lightning data as raw data
in case of empty values
UDP_MINMAX = True               # send min/max values and their occurence
via UDP

[Forward]
FWD_ENABLE = True               # to deactivate this forward temporarily
just set to False instead of deleting the URL (default: True)
FWD_CMT = This is a permanent comment field for notes on this forward
FWD_URL =                       # URL of destination
FWD_INTERVAL =                  # interval in seconds in which lines will be
forwarded
FWD_IGNORE =                    # comma-separated list of fields to not
forward
FWD_TYPE =                      #
WU/UDP/LD/RAW/EW/RAWEW/RAWUDP/AMB/RAWAMB/WC/MT/AWEKAS/WETTERCOM/WEATHER365/R
EALTIMETXT/CLIENTRAWTXT/CSVFILE/TXTFILE/WETTERSEKTOR/MQTTMET/MQTTIMP/RAWTEXT
                                # WU: WU-format
                                # UDP: UDP-String will be forwarded
(default)
                                # LD: PM2.5 luftdaten.info
                                # EW: Ecowitt-format
                                # RAWEW: Ecowitt untouched
                                # RAW: as input
                                # RAWUDP: RAW via UDP
                                # AMB: Ambient-format
                                # RAWAMB: Ambient untouched
                                # WC: Weathercloud-format
                                # MT: Meteotemplate-format
                                # AWEKAS: Awekas API format
                                # WETTERCOM: wetter.com API format
                                # WEATHER365: weather365.net API format
                                # REALTIMETXT: export as realtime.txt
                                # CLIENTRAWTXT: export as clientraw.txt
                                # CSVFILE: export every single record via
http(s)/POST, ftp(s) or save as a file in the filesystem
                                # TXTFILE: export every single record via
http(s)/POST, ftp(s) or save as a file in the filesystem
```

```
                                # WETTERSEKTOR: Wettersektor-format
                                # MQTTMET: metric values to MQTT
                                # MQTTIMP: imperial values to MQTT
                                # RAWTEXT: export every single imperial
record via http(s)/POST, ftp(s) or save as a file in the filesystem
FWD_SID =                       # username for forward if necessary
(SensorID for luftdaten.info)
FWD_PWD =                       # password for this forward if necessary
FWD_STATUS = False              # FWD_TYPE=UDP only: if set to True attach
current status on each outgoing datagram (default: False)
FWD_MQTTCYCLE = 0               # FWD_TYPE MQTTMET/MQTTIMP only: time in
minutes to send the complete record while otherwise only the changes are
sent (default: 0 - send everytime the complete record)
FWD_EXEC =                      # external script to be started immediately
before sending, last line of the script's output is used as the new outstr
for sending
FWD_WARNINT = 10                # threshold of unsuccessful forward attempts
before warning
FWD_QUEUE =                     # type of file to save to if forward target
can not be connected: NONE, INFLUX, AWEKAS, CMX, EW
FWD_QDIR =                      # dircetory to save the queued file(s)

# you additionally can use Forward-1..99
[Forward-1]
FWD_ENABLE = True               # to deactivate this forward temporarily
just set to False instead of deleting the URL (default: True)
FWD_CMT = This is a permanent comment field for notes on this forward
FWD_URL =                       # URL of destination
FWD_INTERVAL =                  # interval in seconds in which lines will be
forwarded
FWD_IGNORE =                    # comma-separated list of fields to not
forward
FWD_TYPE =                      #
WU/UDP/LD/RAW/EW/RAWEW/RAWUDP/AMB/RAWAMB/WC/MT/AWEKAS/WETTERCOM/WEATHER365/R
EALTIMETXT/CLIENTRAWTXT/CSVFILE/TXTFILE/WETTERSEKTOR/MQTTMET/MQTTIMP
FWD_SID =                       # username for forward if necessary
(SensorID for luftdaten.info)
FWD_PWD =                       # password for this forward if necessary
FWD_STATUS = False              # FWD_TYPE=UDP only: if set to True attach
current status on each outgoing datagram (default: False)
FWD_MQTTCYCLE = 0               # FWD_TYPE MQTTMET/MQTTIMP only: time in
minutes to send the complete record while otherwise only the changes are
sent (default: 0 - send everytime the complete record)
FWD_EXEC =                      # external script to be started immediately
before sending, last line of the script's output is used as the new outstr
for sending
FWD_WARNINT = 10                # threshold of unsuccessful forward attempts
before warning
FWD_QUEUE =                     # type of file to save to if forward target
can not be connected: NONE, INFLUX, AWEKAS, CMX, EW
```

```
FWD_QDIR =                      # dircetory to save the queued file(s)

[CSV]
CSV_NAME =                      # file name for csv-file (always metric!)
CSV_FIELDS =                    # fields in desired order, separated with ;
or ,
CSV_INTERVAL =                  # interval in seconds in which lines are
written to the csv-file
CSV_DAYFILE =                   # file name for CSV-dayfile with min/max
values of each last day (will be appended every day after midnight)

[Warning]
WSDOG_WARNING = True            # warn if weather station did not report
data within n send-intervals
WSDOG_INTERVAL = 3              # checking interval for WSDOG_WARNING
WSDOG_RESTART = 0               # automatically restart FOSHKplugin on n
send-intervals without data from weatherstation
SENSOR_WARNING = False          # warn on missing sensor data
SENSOR_MANDATORY = wh65batt     # a comma-separated list of all mandatory
sensors
BATTERY_WARNING = True          # warn if battery level of known sensors is
critical
STORM_WARNING = True            # activate storm warning based on a change
in air pressure
STORM_WARNDIFF = 1.75           # change of air pressure in hPa within one
hour to get warning state
STORM_WARNDIFF3H = 3.75         # change of air pressure in hPa within three
hours to get warning state
STORM_EXPIRE = 60               # minutes, warning will stay active since
last over- and under-range indication
TSTORM_WARNING = True           # enable thunderstorm warning (needs WH57
lightning sensor)
TSTORM_WARNCOUNT= 1             # warn if more than n lightnings were
detected
TSTORM_WARNDIST = 30            # warn only if lightning is closer than n km
TSTORM_EXPIRE = 15              # minutes, warning will stay active since
last lightning
LEAKAGE_WARNING = False         # warn if leakage detected on any WH55
CO2_WARNING = False             # warn if co2 ppm exceeds CO2_WARNLEVEL
CO2_WARNLEVEL = 1000            # ppm threshold to trigger the co2 warning
INTVL_WARNING = False           # warn if receive interval not correspond to
send interval
INTVL_PCT = 10                  # percentage value of the max. permissible
exceeding of the send interval (default: 10)
REBOOT_WARNING = True           # check if current station runtime < last
runtime and warn (log, pushover)
FWD_WARNING = True              # warn after n (defined by FWD_WARNINT)
consecutive failed forward attempts (default: true)
FWD_WARNINT = 10                # Number of failed forward attempts until a
warning occurs (default: 10)
```

```
[Logging]
logfile = REPLACEFOSHKPLUGINLOGDIR/log-foshkplugin.log      # default log - all start/stop/warn/error messages
rawfile = REPLACEFOSHKPLUGINLOGDIR/raw-foshkplugin.log      # logs raw messages coming from weather station only
sndfile = REPLACEFOSHKPLUGINLOGDIR/snd-foshkplugin.log      # logs outgoing messages from plugin (CSV, UDP, FWD)
IGNORE_LOG =                    # a comma-separated list of (sub)strings which will prevent logging a line containing these strings in standard-log
BUT_PRINT = True                # respect IGNORE_LOG also for outputs on the console (default: True)
LOG_LEVEL = ALL                 # with each log level fewer messages will be logged: ALL, INFO, WARNING, ERROR - a lower level includes the higher ones

[Update]
UPD_CHECK = True                # enable/disable firmware-update-check for weatherstation
UPD_INTERVAL = 86400            # interval in seconds of checking for firmware-updates

[Pushover]
PO_ENABLE = False               # enable/disable push notification via Pushover (default: False)
PO_URL =                        # keep empty to use the standard API-URL
PO_TOKEN =                      # generated API-Token from "Your Applications" at Pushover-Dashboard
PO_USER =                       # the user key shown at "Your User Key" at Pushover-Dashboard
PO_CUSTOMWARNING = True         # enable/disable user-defined push messages when values meet a definable condition (100 rules - default: True)
PO_CUSTOM = @tempc <= 2.5,Current temperature (@value°C) is equal or below 2.5°C!,False,3600
PO_CUSTOM1 = @tempf < 32,Current temperature (@value°F) is below 32°F!,False,3600

[Coordinates]
# coordinates are only needed for calculating cloudbase or export to Awekas-API, clientraw.txt, Weather365.net
ALT =                           # altitude in m e.g. 53
LAT =                           # latitude in dec. grad e.g. 52.668759; North of the equator has no sign. South of the equator has a - sign.
LON =                           # longitude in dec. grad e.g. 13.266274; for longitudes left of Greenwich a - sign is needed.

[Sunduration]
SUN_CALC = False                # enable for better sunhours calculation (LAT, LON needed), disable to use static threshold of 120W/m²
SUN_MIN = 0                     # from this value (W/m²) calculation starts
SUN_COEF = 0.8                  # adjustment factor also depends on the location
```

```
SUNSHINE_HOLD = 0                # Hold time in seconds for value sunshine,
this time continues to be output sunshine = True, even if there is no
sunshine (default: 0)
```

Attention!
After changes in the [Weatherstation] area, the new settings must be transferred to the weather station!
By calling ./foshkplugin.py -writeWSconfig, the weather station-specific values from config file are transmitted to the weather station and activated.

datapoints:
The names of the outgoing data points depend on the selected output system (metric or imperial). If USE_METRIC is active, the following data points are output to all configured exports (but not for format-specific forwards):

for Gateway DP1500/GW1000: humidityin tempinc	
soil moisture sensors DP100/WH51: soilbatt1..8 (battery in volt) soilmoisture1..8	**for water sensor WH55:** leak_ch1..4 (1=Alarm, 0=ok) leakbatt1..4 (Battery-Status; 5 = max)
for Multi-Temp/Hum-sensors DP50/WH31: batt1..8 (battery-Status; 1 = Alarm, 0 = ok) humidity1..8 temp1..8c	**for 2- or 3-wing outdoor sensor WH3000SE All-In-One or HP1000SE All-In-One (WH65):** baromabshpa (abs. air pressure in hPa) baromhpa baromrelhpa dailyrainmm dewptc eventrainmm
for PM-sensors DP200/WH41/WH43: pm25_avg_24h_ch1..4 pm25_ch1..4 pm25batt1..4 (Battery-Status; 5 = max) pm25_AQI_ch1..4 pm25_AQI_avg_24h_ch1..4 pm25_AQIlvl_ch1..4 (Level 1..6) pm25_AQIlvl_avg_24h_ch1..4 (Level 1..6)	feelslikec heatindexc hourlyrainmm monthlyrainmm humidity loxtime maxdailygust rainratemm solarradiation tempc
Status/Activity/Tracker: running (1 = startet, 0 = stopped) loxtime (Loxone-time) wswarning (weather station does not send data) sensorwarning (mandatory sensor is missing) batterywarning (Battery-warning) stormwarning (Stormwarning) tswarning (Thunderstorm-warning) updatewarning (Firmware-update available) leakwarning (Leakage present)	totalrainmm uv weeklyrainmm wh65batt (1 for warning, 0=ok) windchillc winddir windgustkmh windspeedkmh yearlyrainmm ptrend1 (air pressure-trend 1h: 2=rapid rising,1=rising, 0=equal,-1=falling,-2 rapid falling) pchange1 (air pressure change in 1h in hPa)
for lightning sensor WH57/DP60: lightning (distance last lightning in km) lightning_time (time of last lightning Unixtime) lightning_loxtime (time of last lightning Loxone-time) lightning_num (count of lightnings) wh57batt (Battery-Status; 5 = max)	ptrend3 (air pressure-trend 3h: 2=rapid rising,1=rising, 0=equal,-1=falling,-2 rapid falling) pchange3 (air pressure change in 3h in hPa) wproglvl (weather prognose level) wprogtxt (weather prognose text) wnowlvl (current weather level) wnowtxt (current weather text)
for soil/water-temp-sensor WN34: tf_chNc (temperature in °C; N=1..8) tf_battN (battery; N=1..8)	

As output via UDP you will get this:

```
SID=FOSHKweather dateutc=2020-06-01+13:35:54 loxtime=360257754 tempinc=27.0
humidityin=30 baromrelhpa=1022.59 baromabshpa=1017.51 tempc=25.1 humidity=30
winddir=273 windspeedkmh=1.43 windgustkmh=7.19 maxdailygust=18.36 solarradiation=677.62
uv=5 rainratemm=0.0 eventrainmm=0.0 hourlyrainmm=0.0 dailyrainmm=0.0 weeklyrainmm=0.0
monthlyrainmm=0.0 yearlyrainmm=206.4 totalrainmm=206.4 temp2c=23.3 humidity2=40
temp3c=24.8 humidity3=34 soilmoisture1=34 soilmoisture2=37 soilmoisture3=41 soilmoisture4=52
pm25_ch1=11.0 pm25_avg_24h_ch1=9.8 lightning_num=0 leak_ch1=0 wh65batt=0 batt2=0
batt3=0 soilbatt1=1.6 soilbatt2=1.6 soilbatt3=1.9 soilbatt4=1.9 pm25batt1=5 wh57batt=5
leakbatt1=5 dewptc=6.3 windchillc=25.1 feelslikec=25.1 heatindexc=24.4 pm25_AQI_ch1=46
pm25_AQIlvl_ch1=1 pm25_AQI_avg_24h_ch1=41 pm25_AQIlvl_avg_24h_ch1=1 ptrend1=-1
pchange1=-0.3 wnowlvl=3 wnowtxt=sonnig ptrend3=-1 pchange3=-1.42 wproglvl=3
wprogtxt="baldiger Regen"
```

where you can easily pickup the required fields for further processing.
Activity and Tracker messages are event-based but also include the SID-token to state these messages are coming from FOSHKplugin:

SID=FOSHKweather stormwarning=1 time=351042104
SID=FOSHKweather stormwarning=0 time=351042135
SID=FOSHKweather tswarning=1 time=359550337
SID=FOSHKweather tswarning=0 time=359551236 start=359550337 end=359551236 last=359550330
SID=FOSHKweather wswarning=1 last=360086463 time=360086560
SID=FOSHKweather wswarning=0 last=360086567 time=360086590
SID=FOSHKweather sensorwarning=1 missed=pm25batt1 time=360169470
SID=FOSHKweather sensorwarning=0 back=pm25batt1 time=360170059

If LOX_TIME is deactivated, the Unixtime appears for all the given times instead of Loxone-time.

There are a some datapoints through which the plugin can be controlled via UDP:

System.reboot	restart the GW1000/DP1500
Plugin.shutdown	shutdown FOSHKplugin - if started as a system service (systemd) it will be restarted some seconds later
Plugin.getstatus	requests the current status values from the plugin (running, wswarning, sensorwarning, ...)
Plugin.getminmax	request the current min/max values from the plugin
Plugin.debug=enable	enable debug mode for some more information in log file
Plugin.debug=disable	disable debug mode
Plugin.pushover=enable	activate Pushover warnings
Plugin.pushover=disable	disable Pushover warnings
Plugin.customwarning=enable	activate custom warnings
Plugin.customwarning=disable	disable custom warnings
Plugin.leakwarning=enable	activate leak warning
Plugin.leakwarning=disable	disable leak warning
Plugin.co2warning=enable	activate co2 warning
Plugin.co2warning=disable	disable co2 warning
Plugin.intvlwarning=enable	activate interval warning
Plugin.intvlwarning=disable	disable interval warning

Plugin.rebootwarning=enable	activate reboot warning
Plugin.rebootwarning=disable	disable reboot warning
Plugin.fwdwarning=enable	activate FWD (forward) warning
Plugin.fwdwarning=disable	disable FWD (forward) warning

By sending a string "SID=FOSHKplugin,System.reboot" the plugin causes the GW1000 to restart. If you send the string "SID=FOSHKplugin,Plugin.getstatus", the plugin replies with the current status values for wswarning, sensorwarning, stormwarning, ...

legal notice:
I do not assume any guarantees regarding the use of this software - use is at your own risk.
Never make decisions that can lead to personal injury or property damage on the basis of this software.
Warnings generated by the program (e.g. storm or thunderstorm) may occur. However, the absence of these warnings does not imply that these things are not possible.

There is a more extensive documentation, although more Loxone-specific and currently in german only: https://foshkplugin.phantasoft.de/
However, the Google translator should be helpful:
https://wiki-loxberry-de.translate.goog/plugins/foshkplugin/start?_x_tr_sl=de&_x_tr_tl=en&_x_tr_hl=de&_x_tr_pto=wapp

Recipes:

change sending interval to a shorter interval than 16 seconds

change sending interval to a shorter interval than 16 seconds

```
# open a console on your Linux-machine
# make sure you have Python & the libraries we need
sudo apt-get -y install –no-upgrade python3 python3-pip
pip3 install requests

# create a new dir somewhere
sudo mkdir /opt/FOSHKplugin

# change into this new dir
cd /opt/FOSHKplugin

# get current version of FOSHKplugin
wget -N http://foshkplugin.phantasoft.de/files/generic-FOSHKplugin.zip

# unzip the file
unzip generic-FOSHKplugin.zip

# if you already know the ip address of the device you want to change you can skip next 3 steps
# the WS_PORT should always be 45000 on FOSHK-devices

# get ip address of weather station –> you will need this later as WS_IP
# if you have more than one weather station you may scan for all devices with:
```

```
# ./foshkplugin.py -scanws
./foshkplugin.py -getwsip

# get port of weather station –> you will need this later as WS_PORT
./foshkplugin.py -getwsport

# get current sending interval - not needed; just for info or check
./foshkplugin.py -getwsinterval

# now set the desired sending interval to 5 seconds - where WS_IP is the ip address of your GW1000 and WS_PORT the command port (probably 45000)
# example: the IP address of your GW1000 is 192.168.0.1, interval should be 5: ./foshkplugin.py -setWSinterval 192.168.0.1 45000 5
./foshkplugin.py -setWSinterval WS_IP WS_PORT 5
```

---

redistribute the Ecowitt stream of a GW1000 to several weewx

redistribute the Ecowitt stream of a GW1000 to several weewx

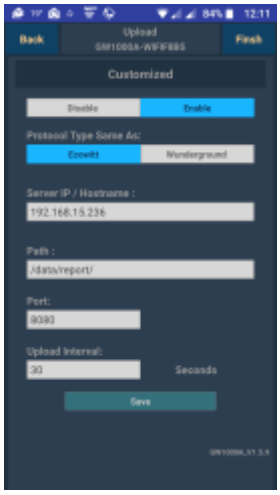
The GW1000 knows exactly **ONE** destination for a custom server upload. But if you want to feed several instances of weewx with the data from the GW1000, this also requires several GW1000.
Or you can simply use FOSHKplugin with just one GW1000 and let FOSHKpugin redistribute the incoming stream as you like!

FOSHKplugin knows 100 forwarding destinations - called Forwards.
A Forward-block must be created in the config file for each forward, in which the target, type and other conditions can be configured:

```
[Forward-n]
FWD_URL =           # URL of destination
FWD_INTERVAL =      # interval in seconds in which lines will be forwarded
FWD_IGNORE = ""     # comma-separated list of fields to not forward
FWD_TYPE = ""       # WU/UDP/LD/RAW/EW/RAWEW/RAWUDP - WU: WU-format; UDP: UDP-String will be forwarded (default); LD: PM2.5 luftdaten.info; EW: Ecowitt; RAWEW: Ecowitt untouched; RAW: as input; RAWUDP: RAW via UDP
FWD_SID = ""        # SensorID for luftdaten.info
```

For the forwarding of the original Ecowitt stream, only the destination (FWD_URL) and the type (FWD_TYPE) have to be defined. The other parameters are optional.

If you want to operate 3 weewx instances with a GW1000, you have to enter the IP address of the FOSHKplugin host under **Server IP / Hostname** and the port number configured there (LBH_PORT) under **Port** via the WS View app.
/data/report/ should be configured as **Path**.



On FOSHKplugin-side you have to create 3 blocks that refer to the respective weewx instances:

```
[Forward-1]
FWD_URL = http://weewx-host1:weewx-port1          # URL of destination
FWD_TYPE = RAWEW                                  # RAWEW: Ecowitt untouched

[Forward-2]
FWD_URL = http://weewx-host1:weewx-port2          # URL of destination
FWD_TYPE = RAWEW                                  # RAWEW: Ecowitt untouched

[Forward-3]
FWD_URL = http://weewx-host1:weewx-port3          # URL of destination
FWD_TYPE = RAWEW                                  # RAWEW: Ecowitt untouched
```

Don't forget to restart the systemd service after making changes to the config file: service foshkplugin restart.
As a result, when an Ecowitt string is received by the GW1000, it is forwarded in parallel and unprocessed to these defined forwards.

Example:

current situation:

GW1000-1 with ip address 192.168.1.1 configured in WS View:	Server IP / Hostname: 192.168.1.100	Port: 8001	Path: (none)
GW1000-2 with ip address 192.168.1.2 configured in WS View:	Server IP / Hostname: 192.168.1.100	Port: 8002	Path: (none)
GW1000-3 with ip address 192.168.1.3 configured in WS View:	Server IP / Hostname: 192.168.1.100	Port: 8003	Path: (none)

You install FOSHKplugin on 192.168.1.100 with LBH_PORT 8080 and create these blocks in your foshkplugin.conf:

[Forward-1] FWD_URL = http://192.168.1.100:8001 FWD_TYPE = RAWEW	 # URL of destination # RAWEW: Ecowitt untouched
[Forward-2] FWD_URL = http://192.168.1.100:8002 FWD_TYPE = RAWEW	 # URL of destination # RAWEW: Ecowitt untouched

[Forward-3] FWD_URL = http://192.168.1.100:8003 FWD_TYPE = RAWEW	 # URL of destination # RAWEW: Ecowitt untouched

restart the FOSHKplugin-service with sudo service foshkplugin restart

final situation:
Afterwards you can remove GW1000-2 and GW1000-3 and reconfigure the remaining GW1000-1 with WS View to:

Server IP / Hostname: 192.168.1.100
Port: 8080
Path: /data/report/

Now the GW1000-1 sends the Ecowitt-Stream to FOSHKplugin which redistribute this to all 3 weewx-instances. Just with **ONE** GW1000.

Saving or further processing of incoming data sets (set-wise)

Saving or further processing of incoming data sets (set-wise)

For further processing of the data from the weather station by other programs, it is recommended to use a cron job in which a shell script is started that fetches the last data record from FOSHKplugin as a string:

```
#! /bin/bash
# several data formats available: CSV, UDP, RAW, STRING, JSON, WU
# metric/imperial and separator are selectable
FMT=STRING&crondaemon
IP=192.168.15.236
PORT=8080

URL=http://$IP:$PORT/$FMT
payload=`curl -s $URL`
# do what you want with station data
echo $payload
```

For example, to start this script every 30 seconds, simply set up two entries in cron:

```
* * * * * /opt/FOSHKplugin/cronjob.sh
* * * * * sleep 30; /opt/FOSHKplugin/cronjob.sh
```

Forwarding of the station data to an MQTT broker

Forwarding of the station data to an MQTT broker

With v0.08, the script-controlled way to send the data is no longer necessary - FOSHKplugin can forward the data directly to an MQTT server.
Both the metric and the imperial values (and names) can be transferred in 2 separate forwards. The forward to different brokers - also in different formats is possible - you then have to set up separate forwards.
FWD_EXEC can be used to intercept the data to be sent to the MQTT broker by any script right BEFORE sending. If necessary, the output data could be modified there. The last output line of the script is then taken by FOSHKplugin as the data intended to forward.
Basically, all available values of the weather station, any values calculated by EVAL_VALUES as well as the status messages and the min/max values are transferred to the MQTT broker.
As usual with forwards, individual fields can be excluded from sending via the ignore list FWD_IGNORE.

Metric names and values are sent with FWD_TYPE = MQTTMET and imperial names/values with FWD_TYPE = MQTTIMP.
In FWD_URL the MQTT broker address, the port and the topic are in the format

```
FWD_URL = MQTTipaddress:port@topic-hierarchy%prefix
```

specified.
Port 1883 is set as the standard port. The default topic is FOSHKweather/Keyname. A prefix is optional and would be placed in front of the name of the key.
A possibly required user name and the specification of the password are made via FWD_SID (username) and FWD_PWD (password). FWD_INTERVAL defines how often the data is sent to the MQTT server. If this value remains free, every incoming data packet is sent from the weather station.

Examples:

```
[Forward-41]
FWD_CMT = MQTT-Forward of metric values to LoxBerry
FWD_TYPE = MQTTMET
FWD_ENABLE = True
FWD_URL = 192.168.15.236:1883%metric_
FWD_STATUS = True
FWD_SID = mqttUSERNAME
FWD_PWD = mqttPASSWORD
FWD_IGNORE =
FWD_MQTTCYCLE = 0
FWD_EXEC =
FWD_INTERVAL =

[Forward-42]
FWD_CMT = MQTT-Forward of imperial values to LoxBerry
FWD_TYPE = MQTTIMP
FWD_ENABLE = True
FWD_URL = 192.168.15.237@house/weatherstation/ecowitt/GW1000/%imperial_
FWD_STATUS = True
FWD_SID = mqttUSERNAME
FWD_PWD = mqttPASSWORD
```

```
FWD_IGNORE =
FWD_MQTTCYCLE = 0
FWD_EXEC =
FWD_INTERVAL = 60
```

In the standard setting, all fields are sent to the MQTT broker via MQTT each time data is received from the weather station.
FWD_MQTTCYCLE can be used to specify that only changed data is transmitted via MQTT. The complete data record of the weather station is only transferred in the cycle of the number of minutes specified here.
FWD_MQTTCYCLE = 10 therefore means that the data is completely sent to the MQTT server every 10 minutes. In the intervening period, only the topics whose value / content have changed are transmitted.
This serves to minimize traffic while at the same time ensuring that all data is available at the destination.

Sending FOSHKplugin data to a MQTT-server (deprecated)

Sending FOSHKplugin data (weather station data) to a MQTT-server (deprecated)

Because there was no native MQTT support up to version v0.08, the method described here was the only option of sending the data to an MQTT broker. This is still possible - but no longer necessary with v0.08 with its dedicated MQTT forward.

However, it is possible to publish on an MQTT server using a wrapper script. A cron job, a small bash script and the MQTT CLI are required.

cronjob

```
* * * * * root /opt/FOSHKplugin/FOSHKplugin2mqtt.sh
* * * * * root sleep 30; /opt/FOSHKplugin/FOSHKplugin2mqtt.sh
```

The bash-script FOSHKplugin2mqtt.sh:

FOSHKplugin2mqtt.sh

```
#! /bin/bash
# FOSHKplugin2mqtt.sh
# script to pull data from FOSHKplugin and send them via mqtt
# could be started through cron

# FOSHKplugin-stuff
IP=192.168.15.236                       # address where FOSHKplugin is
running
PORT=8080                               # port on which FOSHKplugin is
running
```

```
FMT=STRING?units=m&crondaemon          # use units=e for imperial values
#FMT=RAW                               # or use RAW-data instead

# MQTT-stuff
MIP=192.168.15.236                     # address of MQTT-server
MPORT=1883                             # MQTT-port (usually 1883)
MUSR=loxberry                          # username for MQTT-server
MPWD=G3he1mesPassword                  # password for MQTT-user

# pull data from FOSHKplugin
URL=http://$IP:$PORT/$FMT              # just create the URL
payload=`curl -s $URL|sed "s/ /%20/g"|sed "s/[&; ]/ /g"`

# parse and send over to MQTT
for record in $payload; do
  set -- `echo $record | tr '=' ' '`
  key=$1
  value=`echo $2|sed "s/%20/ /g"|sed "s/\"//g"`
  mosquitto_pub --topic FOSHKplugin/$key -u $MUSR -P $MPWD -h $MIP -p $MPORT
-m "$value"
  #echo "$key <-- $value"
done
```

mosquitto_pub is used to publish the data - so you have to make sure that this program is available:

```
sudo apt install mosquitto-clients
```

Installation of FOSHKplugin generic version for several PWT instances

Installation of FOSHKplugin generic version for several PWT instances

Personal Weather Tablet (PWT) on an Android tablet is a very nice alternative to a dedicated console. This means that old, disused tablets can still be put to good use. In custom server mode, PWT expects the weather station to deliver the Ecowitt data to the tablet. Since there is only the possibility of defining a single destination on the weather station, FOSHKplugin instead can receive this data and forward it to any other destination.

Required: 24/7 computer system (e.g. Raspberry Pi) with ssh access or local console for installation

1. Create a directory

```
sudo mkdir /opt/FOSHKplugin
```

2. Change to the directory

```
cd /opt/FOSHKplugin
```

3. Download the installation file

```
wget -N http://foshkplugin.phantasoft.de/files/generic-FOSHKplugin.zip
```

4. Extract the installation file

```
unzip generic-FOSHKplugin.zip
```

5. Start the installation script

```
sudo ./generic-FOSHKplugin-install.sh --install
```

6. Initial configuration
In the square brackets there should already be meaningful defaults that can be selected with ENTER. However, you can also enter your own value in order to overwrite these defaults:

+ + + FOSHKplugin + + + ip address of target system to send UDP-messages to []:
Leave blank if no UDP forwarding is required, otherwise enter the IP address of the destination of the UDP messages.

+ + + FOSHKplugin + + + udp port on target system []:
Leave blank if no UDP forwarding is required, otherwise enter the UDP port of the target system.

+ + + FOSHKplugin + + + ip address of local system [192.168.15.237]:
Enter the IP address of the local system, i.e. the device on which FOSHKplugin is to run (please do not use an IPv6 or localhost address).

+ + + FOSHKplugin + + + http port on local system [8080]:
The local port on which the internal http server is started by FOSHKplugin, the default can be accepted with ENTER.

+ + + FOSHKplugin + + + ip address of weather station [192.168.15.215]:
Please enter the IP address of the weather station here. This is generally found automatically.

+ + + FOSHKplugin + + + command port of weather station [45000]:
The command port of the weather station. Here, too, the default 45000 (the default port for FOSHK weather stations) can be accepted with ENTER.

+ + + FOSHKplugin + + + message-interval of weather station [30]:
Enter the desired interval in seconds at which the weather station sends the data to FOSHKplugin. The default of 30 seconds can be accepted - however, other time intervals can also be used. Note that this is also the minimum time interval for forwards.

+ + + FOSHKplugin + + + are these settings ok? (Y/N)
The settings made are accepted with Y. With N it can be configured again if an error is discovered.

+ + + FOSHKplugin + + + do you want to write settings into the config-file? (Y/N)
With Y these settings are written into the config file (foshkplugin.conf).

+ + + FOSHKplugin + + + do you want to write settings into the weather station? (Y/N)
With Y the required data (IP address and port of the target system from the perspective of the weather station, the interval and the data format Ecowitt) are written to the weather station.

+ + + FOSHKplugin + + + do you want to enable and start the service? (Y/N)
With Y, a service (foshkplugin) is installed and started, which is started automatically every time the computer is restarted and which leads to the restart of FOSHKplugin within a few seconds even if the program crashes.

7. advanced configuration
The foshkplugin.conf file can be edited with any editor in order to make further settings. For forward operation, however, only the required forwards need to be entered.

```
vi foshkplugin.conf
```

The data received from the weather station are forwarded to other systems / programs via so-called forwards. A forward block is required for each desired forwarding, in which the target, format, interval and, if necessary, other forward-specific settings are specified.
A forward block is identified by [Forward-n] where n is a sequential number (1-99). This number must not be repeated within the config file.

Example of a forward to an installation of PWT on a device with the IP address 192.168.15.206:

```
[Forward-1]
FWD_CMT = forward for PWT@Pixel 4a
FWD_ENABLE = True
FWD_URL = http://192.168.15.206:8572/data/report/
FWD_INTERVAL = 30
FWD_TYPE = RAWEW
```

Further forwards according to this template are possible:

```
[Forward-2]
FWD_CMT = forward for PWT@Tablet bedroom
FWD_ENABLE = True
FWD_URL = http://192.168.15.216:8572/data/report/
FWD_INTERVAL = 30
FWD_TYPE = RAWEW

[Forward-3]
FWD_CMT = forward for PWT@Tablet next to the front door
FWD_ENABLE = True
FWD_URL = http://192.168.15.226:8572/data/report/
FWD_INTERVAL = 30
FWD_TYPE = RAWEW
```

Updating a current FOSHKplugin generic installation

Updating the generic FOSHKplugin to the current official release should be easily made:

- open an ssh shell to your server (or use local access)

- change to the directory in which FOSHKplugin is running

- start the update process with

```
sudo ./generic-FOSHKplugin-install.sh --update
```

If you want to update to a specific (Beta) version you should append the filename:

```
sudo ./generic-FOSHKplugin-install.sh --update generic-
FOSHKplugin-0.0.8Beta.zip
```

The current settings in the config file are retained during the update; the service (if configured) will be restarted automatically.

Please use the usual updating way via Plugin-Administration for the LoxBerry-version of FOSHKplugin.

---

get push notifications for critical status changes on the smartphone/tablet

get push notifications for critical status changes on the smartphone/tablet

In addition to notification via UDP, availability as status via http and logging in the log file, important status changes (e.g. firmware update, sensor, watchdog, battery, storm and thunderstorm warnings) can also be sent to any mobile device (iOS , Android) as push notification, FOSHKplugin uses the API of Pushover.
Pushover is a one-time purchase app (no subscription!) that listens in the background for incoming messages from the pushover server.
If configured accordingly, FOSHKplugin sends these critical warnings via API call via the Internet to the Pushover cloud service, which then immediately searches for contact with the devices stored there and delivers the message immediately.
In the mobile device itself, these push notifications then arrive - depending on the setting - with or without sound and/or vibration or silently and are displayed both on the lock screen and in the notification bar. If available and configured accordingly, the notification LED also lights up or flashes.
With adjustable time schedules, you can specify the time periods during which the messages are sent

silently within Pushover.
Besides the one-off purchase price for the app, there are no other hidden costs. At least if you stay below the free 7500 (!) notifications per month.
I've experimented with it for a few days now and I'm thrilled! The delivery takes place reliably and promptly - within a few (here 1-2) seconds (provided that the Internet is available).

Manual:

1. get the app Pushover from the respective store (Android, iOS)
2. Start the Pushover app and assign credentials
3. Log in via web browser: https://pushover.net/login
4. Make a note of the key under "Your User Key", this must be specified for PO_USER in the FOSHKplugin config file foshkplugin.conf
5. Generate an API token for FOSHKplugin under "Your Applications" (Create an Application / API Token) - the key specified under API Token/Key must be specified under PO_TOKEN in the FOSHKplugin config file - as app notification icon you could use this
6. Adjust config foshkplugin.conf: Pushover\PO_ENABLE=True Pushover\PO_USER="Your User Key" and Pushover\PO_TOKEN=API-TOKEN:

[Pushover]
PO_ENABLE = True
PO_USER = userkey
PO_TOKEN = token
It is not necessary to enter a URL under PO_URL. This setting would overwrite the internally predefined pushover URL.

1. Restart FOSHKplugin

From now on there should be a push notification for all important status changes.
The sending of push messages is activated with the switch PO_ENABLE = True in the config file. Sending is deactivated by default (False).
Push notifications from FOSHKplugin can also be activated and deactivated during runtime, provided the correct credentials are stored in the config file.
This is done via http via any web browser by calling up the page http://ipaddress:port/FOSHKplugin/pushover=enable (activate) or http://ipaddress:port/FOSHKplugin/pushover=disable (deactivate). The port is the port specified in the config file under LBH_PORT.
This is also possible via the UDP interface of FOSHKplugin: sending "Plugin.pushover=enable" to the IP address of the host and the port on which FOSHKplugin is running (LBU_PORT in the config file) activates the sending; "Plugin.pushover=disable" deactivates this.
Any errors when sending push notifications as well as activating/deactivating during runtime are logged in the standard log file.

You'll get Pushover-notifications (if configured) in case of:
SENSOR_WARNING = True and a key (any key that is normally contained in the output string of the weather station, such as wh65batt, leakbatt1, soilbatt1, pm25batt1 ...) given in SENSOR_MANDATORY is missed in the incoming string from the weatherstation:
<WARNING> missing data for mandatory sensor [key]
<OK> mandatory data for sensor [key] is back again

BATTERY_WARNING = True and the battery level of all known sensors is below a defined threshold:
<WARNING> battery level for sensor(s) [key] is critical - please swap battery

<OK> battery level for all sensors is ok again

STORM_WARNING = True and air pressure is risen/dropped more than value given in STORM_WARNDIFF within one hour or more than value given in STORM_WARNDIFF3H within three hours:
<WARNING> possible storm - air pressure has risen/dropped more than [threshold] hPa within [count] hours! ([time/pressure before] -> [time/pressure now] diff: [difference] hPa)
<OK> storm warning cancelled after [duration] minutes ([time/pressure before] -> [time/pressure now] diff: [difference] hPa)

TSTORM_WARNING = True and WH57 present and at leas TSTORM_WARNCOUNT lighnings within TSTORM_WARNDIST km were detected:
<WARNING> thunderstorm recognized (start=[warntime])
<OK> thunderstorm warning cancelled after [duration] minutes (start=[warntime] end=[now] last=[last lightning time] lcount=[#lightnings] ldmin=[min. distance] ldmax=[max. distance] ldavg=[avg. distance])

WSDOG_WARNING = True and there were WSDOG_INTERVAL intervals no data received from weatherstation:
<WARNING> weather station has not reported data for more than [count] seconds (WSDOG_INTERVAL send-intervals)
<OK> weather station has reported data again

CO2_WARNING = True and the CO2 value sent by WH45 is above the limit set as CO2_WARNLEVEL:
<WARNING> CO2 sensor reported a value higher than threshold: CO2/[CO2_WARNLEVEL] !seconds (WSDOG_INTERVAL send-intervals)
<OK> CO2 value is ok now (CO2/[CO2_WARNLEVEL] - CO2 warning cancelled!)

LEAKAGE_WARNING = True and one of your leakage sensors WH55 is warning:
<WARNING> leakage reported for sensor(s) #nr!)
<OK> leakage remedied - leakage warning for all sensors cancelled!)

WSDOG_RESTART > WSDOG_INTERVAL and there's still no data from weatherstation:
<WARNING> weather station has not reported data for more than [count] seconds (WSDOG_INTERVAL send-intervals) - restarting

after a start of FOSHKplugin and afterwards every UPD_INTERVAL seconds if there's a newer firmware available for your weatherstation:
<WARNING> firmware update for [model] available - current: [current version] avail: [remote version] use the app [app name] to update!

Operation of multiple FOSHKplugin instances on one host

Operation of multiple FOSHKplugin instances on one host

In some constellations it can make sense to operate several instances of FOSHKplugin in parallel - for example, to process data from several GW1000/DP1500/HP2551C.
Basically this is possible, but it does require a few points to be observed during the installation and, if

necessary, a few adjustments to the configuration file foshkplugin.conf.
In any case, each instance must be installed in its own directory!

Installation:
The http port (LBH_PORT) and the port for incoming UDP messages (LBU_PORT) must not be used more than once on a host. Therefore, a different http port and a different UDP port must be specified for each instance.
The installation routine generic-FOSHKplugin-install.sh automatically ensures that ports are not assigned twice.

By default, a service with the name foshkplugin is created, which can be started and stopped and which starts again automatically in the event of an unscheduled termination.
When running the installation script generic-FOSHKplugin-install.sh, however, a different name can be defined for the service.
Important: a different name must be selected for the service for each instance on the same host in order to be able to operate these different services in parallel!
I recommend naming all FOSHKplugin services with "foshkplugin" - followed by a serial number or a more specific description: for example foshkplugin-GW1 or foshkplugin-Location1.

Adjustments:
If FOSHKplugin is also used to send UDP messages, it may be necessary to change the identifier for these messages so that the target system can assign the incoming messages.
Every outgoing UDP message contains a "SID = FOSHKweather" as an indicator of the data source. If you want to change this identifier, you can specify a different identifier in the config file under Config\DEF_SID.
When the UDP messages are parsed on the processing side, this can then be used as a trigger for further processing.
Alternatively, a distinction can also be made on the basis of the UDP port number (LOX_PORT).

This adjustment is not required for pure forward operation without UDP transmission.

Upload to Ambient Weather

Upload to Ambient Weather

Ambient Weather has a very modern web interface, a nice app, a connection to IFTTT, Amazon Alexa and Google Assistant and can be queried via API interface.
Access to this service requires that you also deliver your weather data there. This is possible with weather stations from Ambient Weather as well as with devices from third-party manufacturers.
A special license is required for operation with devices from third-party manufacturers: VW-ANET. This license is purchased once, is bound to a MAC address and can be used with FOSHKplugin to send data from an Ecowitt station to Ambient Weather.

To connect a GW1000 from Ecowitt (or Froggit DP1500) to Ambient Weather, the following steps are necessary:

1. Registration with Ambient Weather - create an account at https://ambientweather.net/signin

2. Purchase of the VW ANET license stating the MAC address of the GW1000 / DP1500 via https://www.ambientweather.com/amwevwamweac.html

3. Configuration of FOSHKplugin:
   A new forward must be created in the config file:

```
[Forward-21]
FWD_TYPE = RAWAMB
FWD_CMT = Forward for Ambient Weather
FWD_URL = https://api.ambientweather.net/endpoint?
FWD_ENABLE = True
FWD_STATUS = False
FWD_INTERVAL = 180
```

   The device is authenticated via the PASSKEY sent through the GW1000/DP1500 automatically, but can be adjusted in the config file with FWD_SID = [PASSKEY] if necessary.

From now on, FOSHKplugin also sends the incoming data from the GW1000/DP1500 to Ambient Weather.
No additional hardware or software is required.

Set up a forward to PWSDashboard

Set up a forward to PWSDashboard

PWSDashboard is a website template that can be set up very quickly. It visualizes many sensors from the world of fine offset in an extremely beautiful way.
Since PWSDashboard receives data in Ecowitt format, FOSHKplugin actually only needs to set up a forward in Ecowitt format (FWD_TYPE EW or RAWEW):

```
[Forward-11]
FWD_URL = http://192.168.15.236:80/pwsWDxx/data/report/
FWD_ENABLE = True
FWD_TYPE = EW
FWD_CMT = PWSDashboard
FWD_INTERVAL = 30
```

The variable $passkey1 in the index.php in the web server directory pwsWDxx/data/report still has to be filled with the value transmitted under PASSKEY.

```
$passkey1 = '000102030405060708090A0B0C0D0E0F'; // if PASSKEY is known enter
it here
```

The easiest way to find the PASSKEY is in the FOSHKplugin-RAW-log file or copy it from the PWSDashboard log file pwsWDxx/data/report/ecco_stats.txt:

```
Fri, 10 Jul 2020 13:13:09 +0000 = Ecowitt data: 50
000102030405060708090A0B0C0D0E0F needs to be set in index.php
```

Also note the documentation from PWSDashboard for integration via the Ecowitt protocol.

Saving lightning values for the GW1000/DP1500 (WH2650/WH2600Pro)

Saving lightning values for the GW1000/DP1500 (WH2650/WH2600Pro)

With every restart and apparently after a certain time, the GW1000/DP1500 (probably also the WH2650/WH2600Pro) loses the values for the time of the last lightning (lightning_time) and its distance (lightning).
From this point on only empty values are sent instead of the actual values.
In the Ecowitt output string it looks like this:

```
[...]&lightning_time=&lightning_num=0&lightning=&[...]
```

PWS Dashboard or Personal Weather Tablet (PWT) can no longer display lightning data because they do not buffer these values.

With the option Export\FIX_LIGHTNING (enabled by default) FOSHKplugin writes these values with every change that does not contain "" as a value into the config-file as:

foshkplugin.conf

```
[Status]
last_lightning_time = 1610556044
last_lightning = 27
```

and uses these saved values when empty values are received from the weather station for all forwards and export to UDP/CSV/...

When the device is started up for the first time, these values are of course not known to FOSHKplugin. Therefore these values can be entered manually in the config file.
Ecowitt declares the time of the last lightning as Unixtime (UTC). The indication of the distance is saved in kilometers.
A service such as https://www.unixtime.de/ is recommended to calculate the corresponding Unixtime yourself. But you can also just look in the raw-log file and transfer the values.

---

modify outgoing data line (exec)

modify outgoing data line (exec)

The FWD_EXEC function offers the option of editing, exchanging, deleting or adding further data for all data outgoing for this forward before it is actually sent.
In the config file, the line

```
FWD_EXEC = /path/to/script
```

must be entered in the corresponding forward block. Keep in mind to restart FOSHKplugin after any modification of the config file.

The output line generated for this forward is transferred to this script as a parameter.

With a shell script or another program, these data can be changed as required:

```
#!/bin/bash
# add additional data to outgoing data line for e.g. Ecowitt-upload
instr="$@"
echo "$instr"+"&tf_ch1=71.1&tf_batt1=1.1"
```

If you want to integrate the sunhours value from the independent BL-SunRecorder to the output data of a forward, just put the line ***FWD_EXEC = /path/to/addSR.sh*** in the forward section and place a script like:

```
#!/bin/bash
# addSR.sh - gather sunhours from BL Sun Recorder and append the value as
sunhours to the output string
# execute this script with FWD_EXEC = /path/to/addSR.sh in the specific
forward section
instr="$@"
sunhours=`head -1 /some/where/SRsunshine.dat`
if [ ! -z "$sunhours" ]; then srstring="sunhours=$sunhours"; fi
echo "$instr"$srstring
```

to /path/to/.

As in the example above, values can be added from any source - also (with curl) via http. Existing assignments can be changed, exchanged or deleted.

The last output of the script is then taken over by FOSHKplugin and sent to the destination specified under FWD_URL.

Attention!
The user in whose context FOSHKplugin is running requires start permissions for the corresponding script.
Any errors when starting the script are recorded in the log file.
There is a timeout when processing the data! If there is no response from the started script within 10 seconds, FOSHKplugin sends the original string to the configured target instead!

Export of a Cumulus export file realtime.txt or clientraw.txt (Weather Display)

Export of a Cumulus export file realtime.txt or clientraw.txt (Weather Display)

From version v0.08 FOSHKplugin supports forward as well as the storage or the retrieval of a realtime.txt and clientraw.txt. The pull-method can be controlled via cron by downloading from the URL http://ipaddress:port/realtime.txt:

curl -s -o realtime.txt http://192.168.15.236:8080/realtime.txt
or
wget -Nq http://192.168.15.236:8080/realtime.txt [-O /path/to/the/file/realtime.txt]

The forward supports both local storage in the file system and sending to a remote server via http(s)/POST or ftp(s). The form of storage is specified by the FWD_URL in the corresponding forward block.
If FWD_URL is specified as a path name, the file is stored in the specified interval (FWD_INTERVAL) at the specified location (FWD_URL) in the file system:

```
[Forward-31]
FWD_CMT = save as local realtime.txt
FWD_TYPE = REALTIMETXT
FWD_ENABLE = True
FWD_INTERVAL = 30
FWD_URL = /srv/oli-ubuntu2/loxberry-test/

[Forward-32]
FWD_CMT = save as local clientraw.txt
FWD_TYPE = CLIENTRAWTXT
FWD_ENABLE = True
FWD_INTERVAL = 30
FWD_URL = /srv/oli-ubuntu2/loxberry-test/
```

The owner of the generated realtime.txt and clientraw.txt corresponds to the user context in which FOSHKplugin is started. This user therefore needs the appropriate write rights in the target directory.

To save the realtime.txt/clientraw.txt file on a remote server, I recommend sending it via http(s):

```
[Forward-33]
FWD_CMT = save via http/POST as remote realtime.txt
FWD_TYPE = REALTIMETXT
FWD_ENABLE = True
FWD_INTERVAL = 30
FWD_SID = user
FWD_PWD = password
FWD_URL = http://wetter.phantasoft.de/files/upload.php

[Forward-34]
FWD_CMT = save via http/POST as remote clientraw.txt
FWD_TYPE = CLIENTRAWTXT
```

```
FWD_ENABLE = True
FWD_INTERVAL = 30
FWD_SID = user
FWD_PWD = password
FWD_URL = http://wetter.phantasoft.de/files/upload.php
```

On the server side, however, a php file upload.php is required to receive the incoming file and store it in the web server's file system:

upload.php

```
<?php
# set user & password to allow upload
$known_user = "user";
$known_password = "password";

$content = $_POST['content'];
$filename = $_POST['filename'];
$user = $_POST['user'];
$password = $_POST['password'];

if ($known_user == $user and $known_password == $password) {
  $file = $filename;
  $Saved_File = fopen($file, 'w');
  fwrite($Saved_File, $content);
  fclose($Saved_File);
  echo "ok";
  http_response_code(200);
} else {
  echo "unauthorized access!";
  http_response_code(401);
}
?>
```

Alternatively, you can also upload via ftp or ftps. For this purpose, the complete URI is specified in FWD_URL and the credentials are specified via FWD_SID (username) and FWD_PWD (password):

```
[Forward-35]
FWD_CMT = save via ftp as remote realtime.txt
FWD_TYPE = REALTIMETXT
FWD_ENABLE = True
FWD_INTERVAL = 30
FWD_SID = username
FWD_PWD = password
FWD_URL =
ftp://foshkplugin.phantasoft.de/httpdocs/foshkplugin.phantasoft.de/temp/

[Forward-36]
FWD_CMT = save via ftp as remote clientraw.txt
FWD_TYPE = CLIENTRAWTXT
```

```
FWD_ENABLE = True
FWD_INTERVAL = 30
FWD_SID = username
FWD_PWD = password
FWD_URL =
ftp://foshkplugin.phantasoft.de/httpdocs/foshkplugin.phantasoft.de/temp/
```

If the transport is to be TLS-secured (ftps), instead of ftp:// ftps:// must be used in the FWD URL.

It should be noted that not all fields in the realtime.txt and clientraw.txt are filled by FOSHKplugin.

Saving of daily values in a separate CSV file

Saving of daily values in a separate CSV file

For some evaluations it can be helpful to collect the minimum and maximum values of a day as well as other statistical values on a daily basis - together with the time of the respective occurrence.
This statistical data includes, for example, the time and amount of the heaviest rain during the day or the number of lightning strikes. But the minimum and maximum values with the respective time of the day could also be of interest for all temperature values.
FOSHKplugin has a new function for this in v0.08 that creates such a CSV file.
During the first transmission of the day by the weather station (this does not have to be exactly 00:00 but depends on the actual transmission interval of the station), the values and calculations of the last transmission (thus the last of the previous day) are written to the CSV file.
The CSV file only contains metric data and its structure (with semicolon as separator and comma as decimal point) can be processed with Excel (at least in the German version) without any problems.

In the configuration file foshkplugin.conf, a name for the file to be generated must be entered in the [CSV] area under CSV_DAYFILE:

```
[CSV]
CSV_DAYFILE = /path/to/dayfile.csv
```

If a new version of FOSHKplugin results in a new structure (e.g. because new fields have been added or the sorting has been changed), the previous file /path/to/anyfile.csv is automatically renamed to /path/to/anyfile-YYMMDDHHMMSS.csv. New data will then continue to be written to the file configured under CSV_DAYFILE - but with an updated header line.

Attention!
The output file contains columns for all currently available (and announced) sensors. So it is not abnormal that some columns (sensors) do not contain readings. The goal was to create a CSV file that can be used for as long as possible. Therefore, subsequently added sensors had to be integrated in advance.

Background:
For a school project, my son was supposed to record the daily high and low temperatures over a period of one week.
With the data I had for all transimissions, it was difficult (laborious) to convert it to the individual day.
With WSWin I could at least collect the data. But even there, unfortunately, only by hand per day and not as an overview per week (at least I did not find it).
Then I built this function in. If someone can still use that: all the better!

Exporting the weather data to WSWin

Exporting the weather data to WSWin

WSWin is an excellent weather data processing and visualization program from Werner Krenn, which currently lacks the direct support of Fine Offset WIFI stations.
However, it offers ways to import the data from the these stations - both manually and automatically.
WSWin's own file monitoring can be used for the automatic import. This is based on checking a file for changes in a directory to be defined and, if necessary, importing it.
FOSHKplugin offer with FWD_TYPE=WSWIN the creation of such a file - the manual creation/configuration of an X-CSV is not necessary - the values of this file are automatically imported and linked with the correct sensor IDs in WSWin.
A corresponding forward is recommended to generate the file for WSWin:

```
[Forward-36]
FWD_TYPE = WSWIN
FWD_URL = /opt/loxberry/log/plugins/foshkplugin/
FWD_INTERVAL = 60
FWD_CMT = WSWin forward
```

Since the minimum interval for WSWin is 60 seconds, the send interval under FWD_INTERVAL should also be set to at least 60 seconds. The path specified under FWD_URL should be a path that the Windows computer can access (via Samba). Alternatively, the file storage location may also be called up via curl/wget (e.g. via a batch file) if the storage location specified as FWD_URL can be reached via a web browser. The file name is always wswin.csv.

Attention!
The file name is not to be entered here, but the local directory in which the wswin.csv file is stored. The last character of the FWD_URL should therefore be a "/" (slash).

Restarting FOSHKplugin after configuration changes

Restarting FOSHKplugin after configuration changes

Changes to the configuration require the plugin to be restarted. In the Loxone version there is a "Restart" button that restarts the service.
Since the generic version currently does not have a web interface for operation, there are various other ways to restart the plugin:

From the console or via ssh - depending on the Linux distribution used:

```
sudo service foshkplugin restart
```

or

```
sudo systemctl restart foshhkplugin
```

In addition, FOSHKplugin can also be restarted via http/webbrowser or UDP if the RESTART_ENABLE switch in the [Config] area of the config file has been set to True.
Danger!
If the switch is active, the service can be stopped (and automatically restarted by systemd) without further authentication. From anybody who is able to reach your system.
Also important:
If FOSHKplugin is not started as a service but as a program, there is NO automatic restart of FOSHKplugin - the program is only terminated.

FOSHKplugin is restarted via the web browser with

http://ipaddress:port/FOSHKplugin/restartPlugin

where ipaddress is the hostname or IP address of the system running FOSHKplugin and port is the port configured as Config\LBH_PORT.

The string "SID = FOSHKplugin, Plugin.shutdown" must be sent via UDP to the port configured under LBU_PORT in order to terminate FOSHKplugin. It is then automatically restarted as a service by sytemd.

The upper and lower case is to be observed in all cases!
Please activate this switch only if you can ensure that no stranger has access to your FOSHKplugin installation.

Supplying a database with data from the weather station (pull method)

Supplying a database with data from the weather station (pull method)

Since FOSHKplugin also answers HTTP requests from clients, the data can also be called up by FOSHKplugin in various formats.
Telegraf can retrieve this data at a specified interval and write it to various database systems. I use an InfluxDB database for this.
JSON is recommended as the import format. If the min/max data is also to be transferred in JSON, the optional parameter &minmax should be added to the URL.
If the status values are also of interest to the database, a &status can also be passed as parameter.
With parameter &bool, the Boolean values are saved in JSON as True/False instead of 0/1.

A few settings are required in telegraf.conf:

```
[[outputs.influxdb]]
  urls = ["http://127.0.0.1:8086"]                                    # address
of InfluxDB server and port
  database = "weatherstation"

[[inputs.http]]
  urls = [
     "http://192.168.15.237:8080/JSON?minmax&status&bool"             # url of
host, running FOSHKplugin
  ]
  method = "GET"
  timeout = "5s"
  data_format = "json"
  name_override = "weatherdata"
  tag_keys = [
    "PASSKEY"
  ]
  interval = "30s"                                                    # interval
to gather all the data from FOSHKplugin
```

Once the data is in the InfluxDB database, it can also be visualized very nicely with Grafana or Chronograf, for example.

versions in use while testing export to InfluxDB:
InfluxDB v1.6.4 (git: unknown unknown)
Version 7.5.5 (commit: b5190ee547, branch: HEAD)
Telegraf 1.18.2 (git: HEAD a6143722)

---

native connection to an InfluxDB database (push method)

native connection to an InfluxDB database (push method)

With v0.08, FOSHKplugin can forward the weather station data directly to an InfluxDB database.
This requires a forward in the format INFLUXMET or INFLUXIMP - where MET stands for the metric and IMP for the data in the imperial format. The forward format is to be defined as usual with FWD_TYPE in the config.
The server address and database are specified here in the FWD_URL; If the InfluxDB server does not run under the standard port 8086, the correct port can be specified with a colon after the address. The name of the database is given after the @ sign. If the (existing) database requires authentication, the username and password must be entered as FWD_SID and FWD_PWD. If the connection to the database is to be made via SSL, an https:// must be entered instead of http://.
It is not necessary to create a database in advance. FOSHKplugin adds the data to an existing database and - if necessary - creates a new one with the name specified after @ if the database does not exist.
The min/max values are always part of the data transmission. If FWD_STATUS is set to True, the status values are also transferred in Boolean format True/False.

This is what a corresponding forward looks like:

```
[Forward-3]
FWD_TYPE = INFLUXMET
FWD_CMT = InfluxDB-Forward of metric values
FWD_URL = http://192.168.15.237:8086@TestData
FWD_SID = DBusername
FWD_PWD = DBpassword
FWD_STATUS = True
FWD_EXEC =
FWD_IGNORE =
```

If the data is contained in the InfluxDB database, graphical evaluations can be carried out very easily using Grafana or Chronograf.
As an experiment I realized this with Grafana.

versions in use while testing export to InfluxDB:
InfluxDB v1.6.4 (git: unknown unknown)
Version 7.5.5 (commit: b5190ee547, branch: HEAD)
Telegraf 1.18.2 (git: HEAD a6143722)

Log file handling

Log file handling

Log files are very useful for commissioning and in the event of problems.
All messages that are useful for assessing the status or troubleshooting are logged. Also the forward number is logged in order to be able to find faulty blocks in the config file more easily.
However, you should consider whether it is really sensible to keep a permanent log. If an SD card is used as the storage medium, the SD card will eventually be written down - an SD card can only tolerate a defined number of write operations.
Especially the export log - if a very short interval is set - can quickly become very large, because for every message coming from the weather station - depending on the configuration - an entry for UDP, each forwarding (FWD) and CSV is also generated.

We distinguish 3 different log files with different content:

logfile
Contains status messages when the FOSHKplugin is started and stopped, shows the parameters currently in use and logs error messages while running and incoming requests that do not come from the weather station itself. In normal operation in the internal local network, this log should not be particularly large.

rawfile
Logs every (!) incoming line from the weather station. Depending on the selected transmission interval, this file can become very large very quickly - with a transmission interval of 16 seconds, 5400 lines can be stored per day, which, depending on the number of connected sensors, contain 1000 or more characters - that results in approx. 5MByte per day.
You should therefore urgently consider whether this logging really makes sense. I do this - simply to

be able to find an explanation for a strange behavior in retrospect. However, my storage location for the log files is not an SD card but a file server connected via nfs.

sndfile
The fastest growing log file with the largest space requirement - also mentioned as Export-log.
Every (!) line used in a forward is logged here. Each incoming line from the weather station is multiplied by a factor of ten, depending on the number of forwards.
With (currently possible) 100 forwards, 100 * 5MByte per day can accrue in the log level ALL. (That's 500MByte per day!)
I actually write that down here - in order to be able to examine the chain from weather station - FOSHKplugin - send destination for any problems in the event of an error. But as already mentioned above, my logging target is not an SD card but an uncritical hard drive in a server system.
For the normal user, it should not be necessary to write the messages that have been sent successfully.

There are a few options for creating and adjusting log files in the FOSHKplugin.
One can configure the log level (ALL, INFO, WARNING, ERROR), filter out lines with certain strings (LOG_IGNORE), completely (LOG_ENABLE = False) or partially deactivate logging (remove filenames for logfile, rawfile or sndfile).

The sharpest sword for limiting log expenditure is the log level via Logging\LOG_LEVEL in the config file.
You can fine-tune the messages depending on the importance:

- **ALL**, as before, all lines are logged
- **INFO** - all lines except ERROR, WARNING, INFO and OK are hidden
- **WARNING** - all lines except ERROR and WARNING and OK are hidden
- **ERROR** - only lines with ERROR and OK are output

For reasons of compatibility, **ALL** is preset - but I recommend **LOG_LEVEL = INFO** - so everything that was **NOT** successful is logged

Another approach is to hide recurring log lines in the standard log (logfile), for example for recurring queries via cron.
With Logging\IGNORE_LOG, lines can be excluded from the logging in the standard log. All lines that contain one of the strings specified there (comma-separated list of search words/strings) are not logged.

Individual log files can be deactivated by leaving their log file names empty:

```
[Logging]
logfile = /opt/FOSHKplugin/log-foshkplugin.log
#rawfile = /opt/FOSHKplugin/raw-foshkplugin.log
sndfile = /opt/FOSHKplugin/snd-foshkplugin.log
```

deactivates the output of the raw log file - so the incoming lines from the weather station are not logged at all. However, the standard log foshkplugin.log and all outgoing lines are still logged.
So, to deactivate a specific log file, simply remove the name of the respective file.
From v0.08 it is possible to switch logging on and off globally via the switch in the config file Logging\LOG_ENABLE = True/False (default: True):

```
[Logging]
LOG_ENABLE = False
LOG_LEVEL = INFO
IGNORE_LOG = crondaemon
logfile = /opt/FOSHKplugin/log-foshkplugin.log
rawfile = /opt/FOSHKplugin/raw-foshkplugin.log
sndfile = /opt/FOSHKplugin/snd-foshkplugin.log
```

The names configured for logfile, rawfile and sndfile are retained. A change to True and a subsequent restart of FOSHKplugin then writes the originally configured log files again.

LOG_LEVEL can also be adjusted during operation via http://ipadress:port/FOSHKplugin/loglevel=[ALL,INFO,WARNING,ERROR] - however, the value set in the config file applies again after a restart

One final piece of advice on logging concerns the use of logrotate.
As the name of the program suggests, logrotate rotates the log files and shortens them if they exceed a specified size or age. It is part of the Linux distribution and can also be used for shortening or collecting and packing log files from FOSHKplugin.
Instructions for this should be found on the Internet. A first attempt could be this:

/etc/logrotate.d/foshkplugin

```
/opt/FOSHKplugin/*.log {
    daily
    rotate 7
    missingok
    dateext
    copytruncate
    compress
}
```

Just create the file /etc/logrotate.de/foshkplugin as user root and put the content in. Afterwards restart the logrotate service as root to make sure the new settings are in use:

```
sudo service logrotate restart
```

Submit station data to any number of Personal Weather Tablet (PWT) instances with ONE forward

Submit station data to any number of Personal Weather Tablet (PWT) instances with ONE forward

Previously, you had to generate a separate forward for each device on which Personal Weather Tablet (PWT) was to receive the data. I had described this in the recipe "Installation of FOSHKplugin generic version for several PWT instances here.
However, this is quite time-consuming, especially with changing IP addresses (DHCP). And it generates useless traffic even if the corresponding target device is not switched on at all.

Therefore the new version v1.4.1 of PWT now offers a new connection type: the "**UDP broadcast listener**".
If PWT is operated in UDP broadcast listener mode, it listens to incoming data in Ecowwitt format on a configurable UDP port.

FOSHKplugin can already forward incoming Ecowitt data via UDP. A forward like this is required for this:

```
[Forward-46]
FWD_TYPE = RAWUDP
FWD_URL = 192.168.15.255:12350     # use broadcast address:port and
configure PWT to the same UDP port
FWD_CMT = send the Ecowitt stream via UDP broadcast to multiple PWT
instances to broadcastaddress:port
```

This sends the data in Ecowitt format coming from the weather station via UDP to a port (here as an example 12350) to a broadcast address (here 192.168.15.255) at which any number of clients can listen (not only PWT!) and process the incoming data.
I use it to operate 7 or 8 devices in parallel here and thus reduce the data transfer considerably.

From FOSHKplugin v0.09 on there is another forward type EWUDP, which receives incoming data in Ambient Weather, WU and Ecowitt format, if necessary, converts it to Ecowitt format and then sends this data via UDP.
PWT picks up the data and displays them accordingly - at the same time, on all devices listening on this port.
This means that Ambient Weather devices which transmits in AW format and stations that only can transmit data in WU format (e.g. WH600x) can then feed Personal Weather Tablet with data in this way.

I think this new feature of PWT can be very useful for some.
But of course the "old" way via dedicated forwards in FOSHKplugin and connection type "Custom Server" on PWT is still available.

Remapping keys/values

Remapping keys/values

Some forward targets only support a selection of sensors, e.g. Ambient Weather only supports one internal/external PM2.5 sensor or Awekas or WSWin only support 4 soil moisture sensors.
However, FOSHKplugin always transmits logically continuously - i.e. starts with the first sensor and sends the maximum number of channels valid for the forward.
A corresponding assignment or selection can be made with FWD_REMAP.

Example:
Ambient Weather only supports one internal and one external PM2.5 sensor. In the Ecowitt format, however, there are four PM2.5 sensors - no distinction between internal and external.
FOSHKplugin sends the PM2.5 sensor of channel 1 as an outdoor and channel 2 as an indoor sensor to

Ambient Weather.
If you want to use the PM2.5 sensor of channel 3 in Ambient Weather as an indoor sensor, you can use the remap rule:

```
FWD_REMAP = @pm25_ch1=@pm25_ch3,@pm25_avg_24h_ch1=@pm25_avg_24h_ch3
```

which convert the data of the 3rd channel to the 1st channel, whereby the 3rd channel is used as a PM2.5 outdoor sensor for Ambient Weather.
This can be combined as desired - so if you also want to use the data from the 4th channel as a PM2.5 indoor sensor for Ambient Weather, you can use the line

```
FWD_REMAP =
@pm25_ch1=@pm25_ch3,@pm25_avg_24h_ch1=@pm25_avg_24h_ch3,@pm25_ch2=@pm25_ch4,
@pm25_avg_24h_ch2=@pm25_avg_24h_ch4
```

to make this happen.
The assignment is always made with target = origin - whereby the origin can be a static string or a static numerical value or the content of an existing field.
If you want to use the content of a field, an @ must be placed at the beginning of the original field. A

```
FWD_REMAP = @pm25_ch1=pm25_ch3
```

would only assign the string "pm25_ch3" to the pm25_ch1 field. The @ classifies the content of an existing field.
The same goes for the target - the assignment

```
FWD_REMAP = pm25_ch1=@pm25_ch3
```

basically does the same job as

```
FWD_REMAP = @pm25_ch1=@pm25_ch3
```

however, in the second case, if the pm25_ch1 field does not exist, an error message is issued in the log; while in the first case an additional field pm25_ch1 is generated. So the @ in the target is used to check the function.

There are two critical points to note about the remap function:
1. Remapping does not take place globally but is always related to the respective forward.
2. The field names to be specified do NOT correspond to the key names of the target system but to the internal variable names of FOSHKplugin. Depending on the measurement system (metric/imperial) required for the target, these names can be different.

Another example:
With the WN34, Ecowitt offers a total of 8 temperature sensors that can be used for temperature measurements in the ground (WN34S) or in liquids (WN34L).
In addition to the actual interior temperature sensor, WSWin knows 6 other T/H sensors and 4 soil temperature sensors.
FOSHKplugin uses the first 6 WH31 to fill the 6 T/H sensors possible in WSWin and only uses channels 1 to 4 for the floor temperature sensors (channels 5..8 are discarded).
The user may wish to use the temperature and humidity values of an existing WH45 instead of the WH31 channel 4.

A remapping of the data of the WH45 on channel 4 of the additional sensors should then look like this:

```
FWD_REMAP = @temp4c=@tc_co2,@humidity4=@humi_co2
```

If you want to transfer channels 5-8 to WSWin instead of channels 1-4 of the possibly existing 8 WN34, a remap definition like this could be made:

```
FWD_REMAP =
@tf_ch1c=@tf_ch5c,@tf_ch2c=@tf_ch6c,@tf_ch3c=@tf_ch7c,@tf_ch4c=@tf_ch8c
```

Last example:
When forwarding to air data, FOSHKplugin always uses channel 1 of the existing PM2.5 sensors. If you want to use channel 3 instead, this can be done by remapping the values:

```
FWD_REMAP = @pm25_ch1=@pm25_ch3
```

This assigns the content of the pm25_ch3 key to the pm25_ch1 key (which the will be sent to Luftdaten.info).

One final note:
It is also possible to remove individual keys via the remap function. To do this, enter a simple minus sign as the source value:

```
FWD_REMAP = pm25_ch1=-
```

This removes the key pm25_ch1 from the input data for this forward.

The remap function is available for all forwards.

Sending weather station data via APRS/CWOP

Sending weather station data via APRS/CWOP

With v0.09 of FOSHKplugin it is possible to set up a forward to CWOP via the type APRS. This requires a forward definition such as:

```
[Forward-23]
FWD_TYPE = APRS
FWD_URL = cwop.aprs.net:14580
FWD_SID = CWOP-ID
FWD_PWD =
FWD_ENABLE = True
FWD_INTERVAL = 300
```

The CW number, i.e. the identifier of the station, is to be named with FWD_SID. If a password is required for sending, this can be specified as FWD_PWD. Please also note these instructions regarding the need for a password and the server to be used..
Ham radio operators may send weather data without registering on CWOP. Just use rotate.aprs2.net

as the server in FWD_URL and your own APRS credentials in FWD_SID and FWD_PWD.
If the standard port (14580) is used, the port can be omitted from the FWD_URL. Since the transmission does not take place via http, of course **no** http:// may be prefixed.
The minimal interval in FWD_INTERVAL should not be less than 300 (5 minutes).

It is also necessary to specify the decimal coordinates of the location in the Coordinates section:

```
[Coordinates]
ALT = 53          # not needed for APRS/CWOP
LAT = 52.669481   # use negative coordinates for southern hemispehere
LON = 13.266531   # use negative coordinates for west of the prime meridian
```

as these coordinates are an integral part of the data set to be transmitted to CWOP.

The values for winddir, windspeedmph, windgustmph, tempf, hourlyrainin, solarradiation, dailyrainin, humidity and baromrelin are transferred - each (hopefully) in the correct form.
As usual, the fields to be transferred can be remapped with the remap function FWD_REMAP (e.g. to use a WH31 instead of the WH32) and a script can still change or otherwise output the data via FWD_EXEC.

Push notifications when limit values are fallen below or exceeded (getvalue) - deprecated

Push notifications when limit values are fallen below or exceeded (getvalue) - deprecated

Remark:
With v0.10 FOSHKplugin is able to send custom push notifications - see here. The path via this script solution should therefore no longer be necessary. However, since it still works, I'll leave this text in here and just mark it as "deprecated".

Although FOSHKplugin ~~cannot (yet) handle user-specific push notifications~~ (I'm still thinking about how to implement a configuration option), any data manipulation is possible with the Exec function. So you could just start a small script that checks the output data of the weather station for exceeding/falling below threshold values before the actual transmission and then sends a push notification if necessary.

However, it probably makes more sense to use the individual queries (getvalue) straight away. With getvalue, any values can be queried from FOSHKplugin - not only the original values of the weather station but also the values converted into the metric system, the min/max values, the calculated values and every status reported by FOSHKplugin.
The retrieval takes place via http (e.g. with curl) and follows the scheme
http://ipaddress:port/getvalue?key=keyname
The possible keynames can be displayed via http://ipaddress:port/minmax&status&units=m (metric) and http://ipaddress:port/minmax&status&units=e (imperial).

If I wanted to be warned when the temperature falls below a certain limit, I would simply install a cron job that queries the corresponding temperature at intervals and warns if the temperature falls below

it with email, SMS, push notification or Alexa voice output.

Here is an example for the outside temperature:

crontab

```
*/5 * * * * root /usr/local/bin/checkTemp.sh
```

And the actual bash script checkTemp.sh, which is started every 5 minutes via cron. Of course, you can change the interval in the crontab as you like.

checkTemp.sh

```
#!/bin/bash

# Pushover settings
po_token='YOURPUSHOVERTOKEN'
                # Pushover API token
po_user='YOURPUSHOVERUSER'
                # Pushover user name
po_title='checkTemp.sh-Warning'
               # Pushover message title
po_message=''
               # Pushover message - will be filled later

# thresholds
temp_min_thrs=5.5
# temperature min threshold
temp_max_thrs=25.5
# temperature max threshold

# main
tempc=`curl -s http://192.168.15.236:8080/getvalue?key=tempc`
# retrieve temperature in °C (use tempf for °F)
if [[ ! -z "$tempc" ]]; then
                # make sure of value
  # outdoor temperature min
  if (( $(echo "$tempc < $temp_min_thrs" |bc -l) )); then
# if current temp < min threshold
    po_message="outdoor temperature (${tempc}°C) is below
${temp_min_thrs}°C!"
    #echo $po_message
               # remove for cron
    curl -s -o /dev/null --form-string "token=${po_token}" --form-string
"user=${po_user}" --form-string "title=${po_title}" --form-string
"message=$po_message" https://api.pushover.net/1/messages.json
  # outdoor temperature max
  elif (( $(echo "$tempc > $temp_max_thrs" |bc -l) )); then
# if current temp > max threshold
    po_message="outdoor temperature (${tempc}°C) is higer than
${temp_max_thrs}°C!"
```

```
    #echo $po_message
                # remove for cron
    curl -s -o /dev/null --form-string "token=${po_token}" --form-string
"user=${po_user}" --form-string "title=${po_title}" --form-string
"message=$po_message" https://api.pushover.net/1/messages.json
  fi
fi
```

What is still missing here is a routine to avoid permanent push notifications - otherwise a corresponding message would be issued every 5 minutes if the temperature value is outside the threshold values.

---

problem solving

problem solving

There are a few well-known problems that occur here and there, and I would like to briefly describe their solutions here:

Most problems are caused by the rather complicated rights management in Linux. Especially with manual changes to the config file or with updates, the access permissions to the corresponding files can be changed or receive the wrong owner.

For understanding:

Only the owner of a file may access a file. He therefore needs access rights that are derived from the ownership. Of course, other users may also access files - provided they have the appropriate permissions. If the first installation takes place in the context of user 1, the start as well as any updates or changes in the config file must be carried out by exactly this user, unless the group rights are changed.

Sometimes, however, one cannot remember in which context the installation originally took place or forgets to prefix the required "*sudo -u username*" when editing or updating. The owner as well as the file permissions can be shown at any time from the console or via ssh with a

```
ls -lah /opt/FOSHKplugin/
```

Example:

```
root@test:/opt/FOSHKplugin# ls -lah /opt/FOSHKplugin/
total 516K
drwxr-xr-x  2 root root 4.0K Feb 11 12:32 .
drwxr-xr-x 11 root root 4.0K Feb 11 12:31 ..
-rw-rw-rw-  1 root root  12K Jan 22 00:09 foshkplugin.conf
-rw-r--r--  1 root root  12K Jan 22 00:09 foshkplugin.conf.orig
-rw-r--r--  1 root root 2.4K Jan 22 00:09 foshkplugin.png
-rwxr-xr-x  1 root root 326K Jan 22 00:09 foshkplugin.py
-rw-r--r--  1 root root  408 Jan 22 00:09 foshkplugin.service
-rw-r--r--  1 root root  95K Jan 22 09:28 generic-FOSHKplugin-0.0.9Beta.zip
```

```
-rwxr-xr-x  1 root root  13K Jan 22 00:09 generic-FOSHKplugin-install.sh
-rw-r--r--  1 root root  35K Jan 22 00:09 README.md
```

The same owner (root:root in this example) should be entered here for all files.

Of course, you can also change the owner of the files with *chown* or adjust the file rights with *chmod*. However, I recommend using the integrated repair function. A

```
/opt/FOSHKplugin/generic-FOSHKplugin-install.sh -repair
```

is always a good way to adjust the user rights. This repair also automatically installs any necessary but missing libraries. Especially after an update of FOSHKplugin (before v0.09), additional libraries may be required that need to be installed - the repair function automates this.

In rare cases, the FOSHKplugin config file may become corrupted and FOSHKplugin will no longer start. With every successful start FOSHKplugin creates a backup copy of the current (working) configuration, which can easily be reverted to in this case:

A

```
sudo cp -p /opt/FOSHKplugin/foshkplugin.conf.foshkbackup
/opt/FOSHKplugin/foshkplugin.conf
```

copies the last working version of the config file over the (broken) file actually in use.

Also a common reason for support requests:

Changes to the config file only become active AFTER a restart of FOSHKplugin! So you can make any changes in the config file - but they will only become active after FOSHKplugin has been restarted. Restarting FOSHKplugin is of course done by restarting the underlying operating system. However, it is easier and faster to restart FOSHKplugin alone. If FOSHKplugin was set up as a service, in most cases this service can be restarted by a

```
sudo service foshkplugin restart
```

to restart the service. When the service is started, the current config file is read in and processed. If FOSHKplugin runs in the foreground and not as a service, a CTRL-C in the programme window or (if it runs in the background) a killall FOSHKplugin.py with a subsequent restart of the programme is sufficient.

In general, it is a good idea to run FOSHKplugin in the foreground in case of unclear problems. All you have to do is stop the corresponding service:

```
sudo service foshkplugin stop
```

and then start FOSHKplugin like a normal program from the command line:

```
sudo -u username /opt/FOSHKplugin/foshkplugin.py
```

Any error messages are output directly in plain text. These can then be copied and sent to me by e-mail for further analysis.

FOSHKplugin supports countless services and various weather stations - I cannot test them all to the same extent. I also don't have all the supported weather stations and sensors here myself (although I try) and with some (such as devices from Ambient Weather) it is almost impossible for me to test them in conjunction with FOSHKplugin (here in Europe 868MHz are mandatory). Therefore, it can happen that I myself do not notice any program errors because the constellation of the user is possibly different than here. Therefore, **I ask all users who find a bug to report it to me** - if possible in connection with corresponding log files.

Set up a forward to CumulusMX (CMX)

Set up a forward to CumulusMX (CMX)

CMX is a free, very nice and comprehensive software for visualising and storing the data of a local weather station. It supports many forwarding options and interfaces to other programmes and presents the data - web-based - clearly in a dashboard; offers visualised instruments and enables long-term archiving of the data. The integration of local weather data into own websites is also supported.
The user does not have to configure or adjust much himself - most of it works out of the box.
On a 24/7 running computer, this can be THE central weather station app.
More information and the link to download the CMX package can be found here.

CMX does not need to be installed for operation under Windows - it is sufficient to unpack the ZIP file in a folder and start the CumulusMX.exe there. The further configuration is then done via web browser using the link http://ipaddress:8998 where ipaddress corresponds to the address of the computer on which CMX was started.

In CMX you only have to go through the configuration wizard from the settings menu (Settings/Config Wizard) and enter all mandatory values (e.g. lat/lon/alt/...) and select HTTP Upload (Ecowitt) as station type.

After restarting CMX (just press CTRL-C in the corresponding Dos window and restart CumulusMX.exe) CMX should be able to receive data from the weather station via FOSHKplugin (or directly from the station) and wait for incoming data.

Since CMX receives data in Ecowitt format, you only have to set up one (more) forward in Ecowitt format in the config file of FOSHKplugin foshkplugin.conf (FWD_TYPE EW or RAWEW):

```
[Forward-77]
FWD_TYPE = EW
FWD_CMT = CMX
FWD_URL = http://192.168.15.252:8998/station/ecowitt/
```

The IP address should be the address of the computer running CMX (in this example 192.168.15.252); the port is usually 8998 and the path should be /station/ecowitt.

Don't forget to restart the FOSHKplugin service afterwards, as all newly created forwards will only be activated when FOSHKplugin is started:

```
sudo service foshkplugin restart
```

In this way CMX should receive all data from the local weather station - with FOSHKplugin as middleman.

---

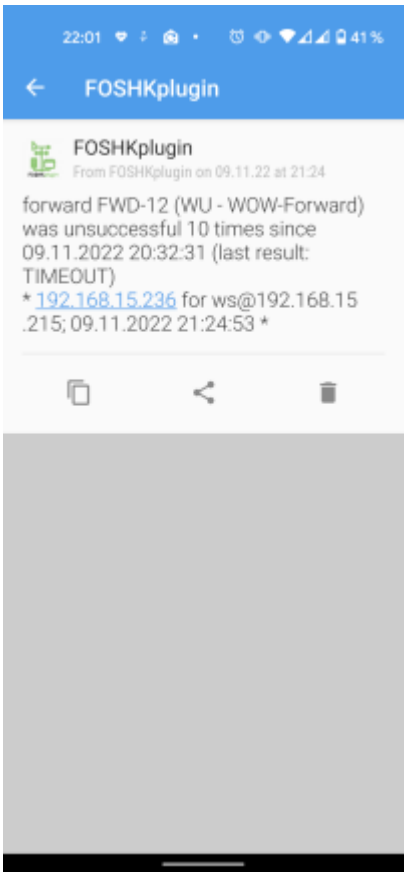
be warned in case of failed forwards (from v0.10)

be warned in case of failed forwards (from v0.10)

If you have configured dozens of forwards that are inconspicuously served by FOSHKplugin, it is quite interesting and sometimes also important to be informed in case of any (longer) failures. Because sometimes there are simply problems with the internet connection or with the weather service provider itself.

FOSHKplugin writes a line to the log file snd-foshkplugin.log for each failed forward. But who looks there regularly and without reason?

In order to still receive information relatively promptly if a forward destination is not reachable for a longer period of time, as of v0.10 there is the possibility to be informed via push notification (Pushover). After a configurable number of successive unsuccessful attempts, a push message is sent:




```
FOSHKplugin
From FOSHKplugin on 2022-11-11 at 17:24
forward FWD-53 (INFLUX2MET - InfluxDB 2.x-Forward of metric values to
LoxBerry) was unsuccessful 15 times since 11.11.2022 17:16:40
(last result: : Failed to establish a new connection: [Errno 113] No route
to host)
* 192.168.15.237 for ws@192.168.15.221; 11.11.2022 17:24:53 *
```

Included are the forward designation (Forward-nn), the forward type (FWD_TYPE) and any forward comment (FWD_CMT) - together with the error message and the time of the last successful

transmission.

This message is sent exactly ONCE after the number of failed attempts specified in the config file under Warning\FWD_WARNINT. The default value is 10.
With each successful attempt, the internal counter is reset - so you really only get a warning if the specified number of failed attempts occur after each other.
After a push message, another message is only generated for this forward if at least one forward was successful before and thus the internal counter was reset.
This function can be deactivated via Warning\FWD_WARNING = False in the config file.
As usual, however, this can also be deactivated at runtime via http (***http://ipaddress:portnumber/FOSHKplugin/fwdwarning=disable***) or via UDP (Plugin.fwdwarning=disable) or activated via enable.

Beside the global variable Warning\FWD_WARNINT that specifies the number of failed forwards before a warning is issued via push notification, an individual value can be configured in each forward with FWD_WARNINT. This setting overrides the global default for that specific forward and allows the warning to be disabled individually by setting FWD_WARNING = 0.
This way, the warning behaviour can optionally be configured separately for each forward.

Counting starts whenever FOSHKplugin has been started - the former state will not be saved.

In addition, there is also a simple html page at ***http://ipaddress:portnumner/FOSHKplugin/fwdstat*** that shows the status of all activated forwards. There you can see at a glance whether and since when a forward has been hanging and, if so, with what error message.
On this forward status page, different font colours show at a glance the status of the forwards: black: everything is ok, orange: the last forward was unsuccessful and red: the number of successive unsuccessful attempts exceeds the defined warning limit:

FOSHKplugin v0.10 fwdstat

forward statistics at 11/11/2022 13:51:44 (warn threshold globally set to 15 or forward specific attempts - see (int) in errcount row):

forward	type	url	last attempt	last ok	last state	err count (int
FWD-01	WETTERSEKTOR	http://wetter.phantasoft.de/files/sendfile.php	11/11/2022 13:51:32		401	1 (0)
FWD-02	MQTTIMP	192.168.15.236@OliTestIMP	11/11/2022 13:51:32	11/11/2022 13:51:33	OK	0 (15)
FWD-06	WSWIN	http://wetter.phantasoft.de/files/sendfile.php	11/11/2022 13:51:32	11/11/2022 13:51:32	OK	0 (15)
FWD-07	WSWIN	ftps://wetter.phantasoft.de/httpdocs/wetter.phantasoft.de/files/wswin-growing.csv	11/11/2022 13:51:32	11/11/2022 13:51:33	OK	0 (15)
FWD-08	TXTFILE	ftps://wetter.phantasoft.de/httpdocs/wetter.phantasoft.de/files/237-txtfile.txt	11/11/2022 13:51:32	11/11/2022 13:51:33	OK	0 (15)
FWD-09	RAWTEXT	ftps://wetter.phantasoft.de/httpdocs/wetter.phantasoft.de/files/237-rawtext.txt	11/11/2022 13:51:32	11/11/2022 13:51:33	OK	0 (15)
FWD-10	WETTERCOM	http://192.168.15.100/wettercom/	11/11/2022 13:51:32		404	1 (15)
FWD-11	WETTERSEKTOR	http://192.168.15.100/wettersektor/	11/11/2022 13:51:32		404	1 (15)
FWD-12	RAWUDP	192.168.15.255:12360	11/11/2022 13:51:32	11/11/2022 13:51:32	OK	0 (15)
FWD-13	INFLUXMET	http://192.168.15.237@WeatherMET	11/11/2022 13:51:32	11/11/2022 13:51:33	OK	0 (15)
FWD-14	INFLUXIMP	http://192.168.15.237@WeatherIMP	11/11/2022 13:51:32	11/11/2022 13:51:33	OK	0 (15)
FWD-15	WU	http://192.168.15.100/WU-upload.php?	11/11/2022 13:51:32		404	1 (15)
FWD-16	WU	http://192.168.15.100/AMB-upload.php?	11/11/2022 13:51:32		404	1 (15)
FWD-17	MT	http://192.168.15.100/MT?	11/11/2022 13:51:32		404	1 (15)
FWD-18	WC	http://192.168.15.100/MT?	11/11/2022 13:51:32		404	1 (15)
FWD-19	AWEKAS	http://192.168.15.100/AWEKAS?	11/11/2022 13:51:33		404	1 (15)
FWD-20	WEATHER365	http://192.168.15.100/weather365/	11/11/2022 13:51:33		404	1 (15)
FWD-22	EWUDP	192.168.15.255:12361	11/11/2022 13:51:33	11/11/2022 13:51:33	OK	0 (15)
FWD-23	APRS	cwop.aprs.net:14580	11/11/2022 13:51:33	11/11/2022 13:51:33	OK	0 (15)

Adjust the output format for date and time (from v0.10)

Adjust the output format for date and time (from v0.10)

The formatting of date and time in all outputs - such as CSV, LOG or even in the push notification - can now be adapted via the optional setting Config\DT_FORMAT in the Config file.
This can then be used to determine whether this should be output as 10.11.2022 17:34:20 or

10/11/2022 17:34:20 or 11/10/2022 05:34:20 pm or whatever.

Here, the internal possibilities of the time library in Python are simply used - for dd.mm.yyyy hh:mm:ss (as usual for Germany), one simply enters in the config file under [Config]:

```
[Config]
DT_FORMAT = %d.%m.%Y %H:%M:%S
```

However, this is also the default that is used with the existing configuration. For UK/US users, these variables must be swapped somewhat.

Relevant directives are:

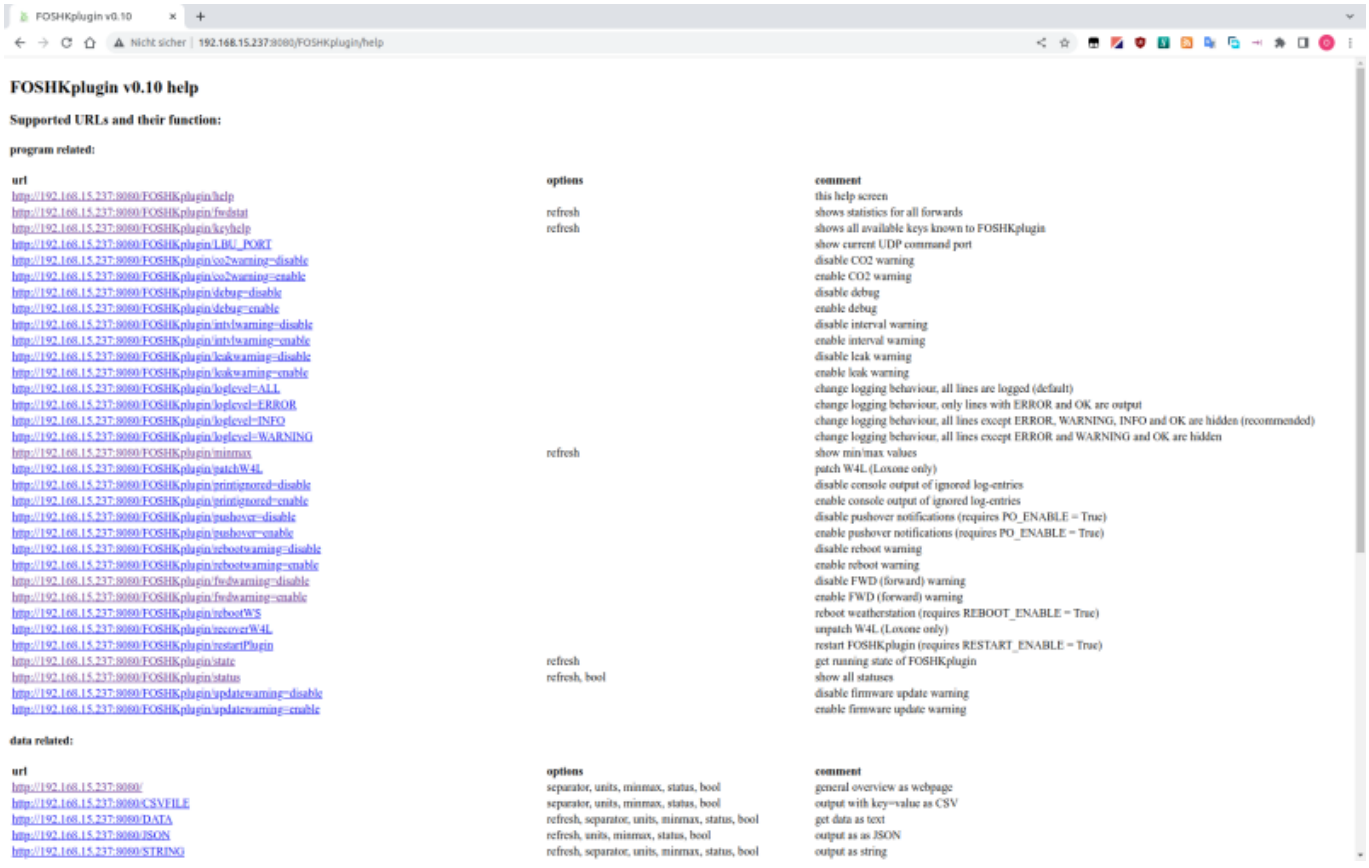
```
%d  Day of the month as a decimal number [01,31].
%m  Month as a decimal number [01,12].
%y  Year without century as a decimal number [00,99].
%Y  Year with century as a decimal number.

%H  Hour (24-hour clock) as a decimal number [00,23].
%I  Hour (12-hour clock) as a decimal number [01,12].
%p  Locale’s equivalent of either AM or PM.
%M  Minute as a decimal number [00,59].
%S  Second as a decimal number [00,61].
```

---

get help from FOSHKplugin

get help from FOSHKplugin

Documentation is always a nightmare.
Therefore, with v0.9 an internal help was implemented in FOSHKplugin, which can be called via web browser.
A help page is available at http://ipaddress:portnumber/FOSHKplugin/help that summarises and explains all web commands for FOSHKplugin.



FOSHKplugin v0.10

Nicht sicher | 192.168.15.237:8080/FOSHKplugin/help

FOSHKplugin v0.10 help

Supported URLs and their function:

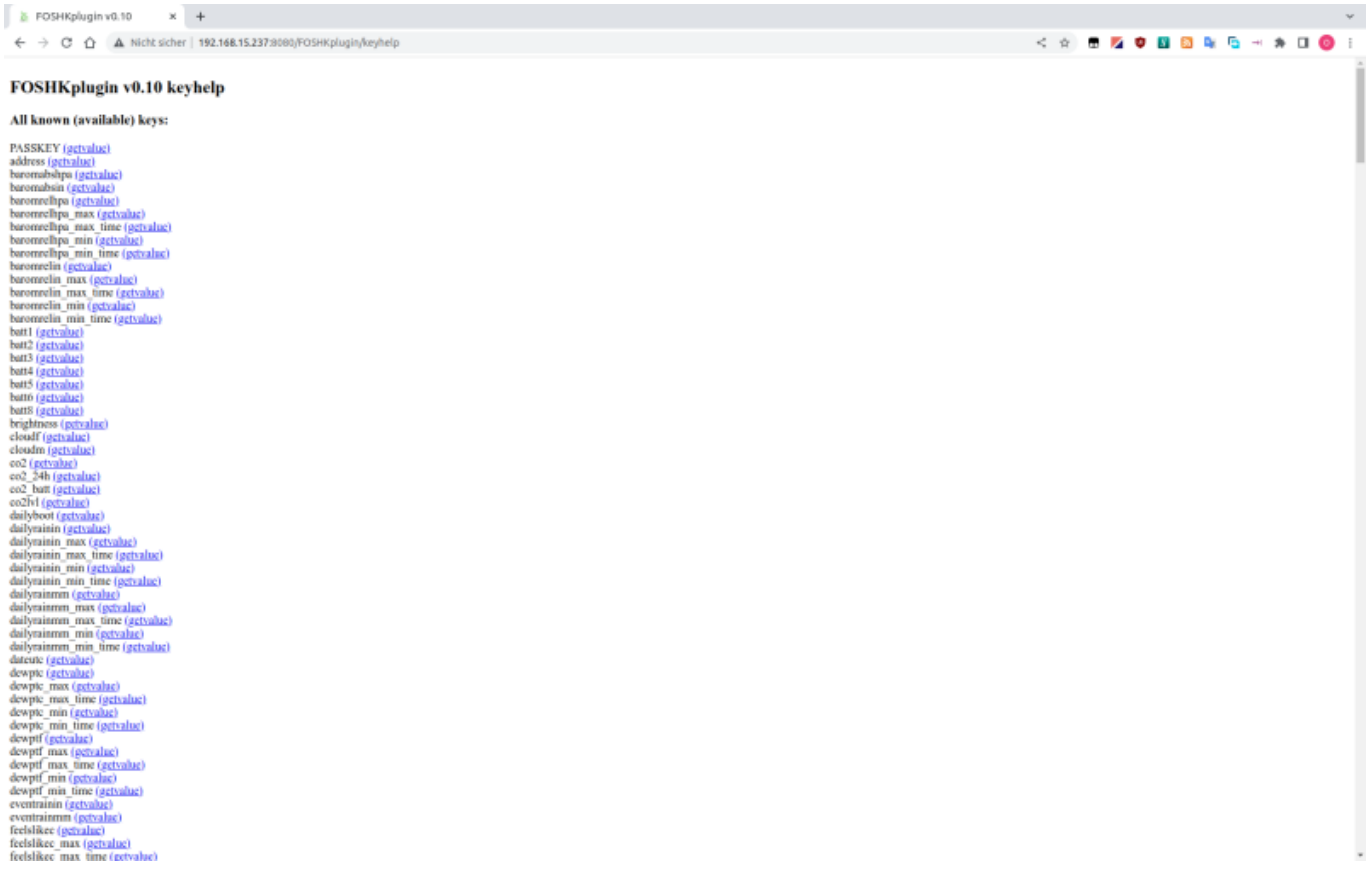
program related:

url	options	comment
http://192.168.15.237:8080/FOSHKplugin/help		this help screen
http://192.168.15.237:8080/FOSHKplugin/fwdstat	refresh	shows statistics for all forwards
http://192.168.15.237:8080/FOSHKplugin/keyhelp	refresh	shows all available keys known to FOSHKplugin
http://192.168.15.237:8080/FOSHKplugin/LBU_PORT		show current UDP command port
http://192.168.15.237:8080/FOSHKplugin/co2warning=disable		disable CO2 warning
http://192.168.15.237:8080/FOSHKplugin/co2warning=enable		enable CO2 warning
http://192.168.15.237:8080/FOSHKplugin/debug=disable		disable debug
http://192.168.15.237:8080/FOSHKplugin/debug=enable		enable debug
http://192.168.15.237:8080/FOSHKplugin/intvlwarning=disable		disable interval warning
http://192.168.15.237:8080/FOSHKplugin/intvlwarning=enable		enable interval warning
http://192.168.15.237:8080/FOSHKplugin/leakwarning=disable		disable leak warning
http://192.168.15.237:8080/FOSHKplugin/leakwarning=enable		enable leak warning
http://192.168.15.237:8080/FOSHKplugin/loglevel=ALL		change logging behaviour, all lines are logged (default)
http://192.168.15.237:8080/FOSHKplugin/loglevel=ERROR		change logging behaviour, only lines with ERROR and OK are output
http://192.168.15.237:8080/FOSHKplugin/loglevel=INFO		change logging behaviour, all lines except ERROR, WARNING, INFO and OK are hidden (recommended)
http://192.168.15.237:8080/FOSHKplugin/loglevel=WARNING		change logging behaviour, all lines except ERROR and WARNING and OK are hidden
http://192.168.15.237:8080/FOSHKplugin/minmax	refresh	show min/max values
http://192.168.15.237:8080/FOSHKplugin/patchW4L		patch W4L (Loxone only)
http://192.168.15.237:8080/FOSHKplugin/printignored=disable		disable console output of ignored log-entries
http://192.168.15.237:8080/FOSHKplugin/printignored=enable		enable console output of ignored log-entries
http://192.168.15.237:8080/FOSHKplugin/pushover=disable		disable pushover notifications (requires PO_ENABLE = True)
http://192.168.15.237:8080/FOSHKplugin/pushover=enable		enable pushover notifications (requires PO_ENABLE = True)
http://192.168.15.237:8080/FOSHKplugin/rebootwarning=disable		disable reboot warning
http://192.168.15.237:8080/FOSHKplugin/rebootwarning=enable		enable reboot warning
http://192.168.15.237:8080/FOSHKplugin/fwdwarning=disable		disable FWD (forward) warning
http://192.168.15.237:8080/FOSHKplugin/fwdwarning=enable		enable FWD (forward) warning
http://192.168.15.237:8080/FOSHKplugin/rebootWS		reboot weatherstation (requires REBOOT_ENABLE = True)
http://192.168.15.237:8080/FOSHKplugin/recoverW4L		unpatch W4L (Loxone only)
http://192.168.15.237:8080/FOSHKplugin/restartPlugin		restart FOSHKplugin (requires RESTART_ENABLE = True)
http://192.168.15.237:8080/FOSHKplugin/state	refresh	get running state of FOSHKplugin
http://192.168.15.237:8080/FOSHKplugin/status	refresh, bool	show all statuses
http://192.168.15.237:8080/FOSHKplugin/updatewarning=disable		disable firmware update warning
http://192.168.15.237:8080/FOSHKplugin/updatewarning=enable		enable firmware update warning

data related:

url	options	comment
http://192.168.15.237:8080/	separator, units, minmax, status, bool	general overview as webpage
http://192.168.15.237:8080/CSVFILE	separator, units, minmax, status, bool	output with key=value as CSV
http://192.168.15.237:8080/DATA	refresh, separator, units, minmax, status, bool	get data as text
http://192.168.15.237:8080/JSON	refresh, units, minmax, status, bool	output as as JSON
http://192.168.15.237:8080/STRING	refresh, separator, units, minmax, status, bool	output as string

An overview of the variable names that can be queried is available at ***http://ipaddress:portnumber/FOSHKplugin/keyhelp***.
This is especially interesting if you want to query individual values via http using getvalue.



FOSHKplugin v0.10

Nicht sicher | 192.168.15.237:8080/FOSHKplugin/keyhelp

FOSHKplugin v0.10 keyhelp

All known (available) keys:

PASSKEY (getvalue)
address (getvalue)
baromabshpa (getvalue)
baromabsin (getvalue)
baromrelhpa (getvalue)
baromrelhpa_max (getvalue)
baromrelhpa_max_time (getvalue)
baromrelhpa_min (getvalue)
baromrelhpa_min_time (getvalue)
baromrelin (getvalue)
baromrelin_max (getvalue)
baromrelin_max_time (getvalue)
baromrelin_min (getvalue)
baromrelin_min_time (getvalue)
batt1 (getvalue)
batt2 (getvalue)
batt3 (getvalue)
batt4 (getvalue)
batt5 (getvalue)
batt6 (getvalue)
batt8 (getvalue)
brightness (getvalue)
cloudf (getvalue)
cloudm (getvalue)
co2 (getvalue)
co2_24h (getvalue)
co2_batt (getvalue)
co2lvl (getvalue)
dailyboot (getvalue)
dailyrainin (getvalue)
dailyrainin_max (getvalue)
dailyrainin_max_time (getvalue)
dailyrainin_min (getvalue)
dailyrainin_min_time (getvalue)
dailyrainmm (getvalue)
dailyrainmm_max (getvalue)
dailyrainmm_max_time (getvalue)
dailyrainmm_min (getvalue)
dailyrainmm_min_time (getvalue)
dateutc (getvalue)
dewpic (getvalue)
dewpic_max (getvalue)
dewpic_max_time (getvalue)
dewpic_min (getvalue)
dewpic_min_time (getvalue)
dewptf (getvalue)
dewptf_max (getvalue)
dewptf_max_time (getvalue)
dewptf_min (getvalue)
dewptf_min_time (getvalue)
eventrainin (getvalue)
eventrainmm (getvalue)
feelslikec (getvalue)
feelslikec_max (getvalue)
feelslikec_max_time (getvalue)

In addition, as of v0.10 there is also a simple web page that shows the status of all active forwards.

There you can see at a glance whether and since when a forward has been hanging and, if so, with what error message.
This can be reached via ***http://ipaddress:portnumber/FOSHKplugin/fwdstat***.

FOSHKplugin v0.10 fwdstat

forward statistics at 11/11/2022 13:51:44 (warn threshold globally set to 15 or forward specific attempts - see (int) in errcount row):

forward	type	url	last attempt	last ok	last state	err count (int
FWD-01	WETTERSEKTOR	http://wetter.phantasoft.de/files/sendfile.php	11/11/2022 13:51:32		401	1 (0)
FWD-02	MQTTIMP	192.168.15.236@OliTestIMP	11/11/2022 13:51:32	11/11/2022 13:51:33	OK	0 (15)
FWD-06	WSWIN	http://wetter.phantasoft.de/files/sendfile.php	11/11/2022 13:51:32	11/11/2022 13:51:32	OK	0 (15)
FWD-07	WSWIN	ftps://wetter.phantasoft.de/httpdocs/wetter.phantasoft.de/files/wswin-growing.csv	11/11/2022 13:51:32	11/11/2022 13:51:33	OK	0 (15)
FWD-08	TXTFILE	ftps://wetter.phantasoft.de/httpdocs/wetter.phantasoft.de/files/237-txtfile.txt	11/11/2022 13:51:32	11/11/2022 13:51:33	OK	0 (15)
FWD-09	RAWTEXT	ftps://wetter.phantasoft.de/httpdocs/wetter.phantasoft.de/files/237-rawtext.txt	11/11/2022 13:51:32	11/11/2022 13:51:33	OK	0 (15)
FWD-10	WETTERCOM	http://192.168.15.100/wettercom/	11/11/2022 13:51:32		404	1 (15)
FWD-11	WETTERSEKTOR	http://192.168.15.100/wettersektor/	11/11/2022 13:51:32		404	1 (15)
FWD-12	RAWUDP	192.168.15.255:12360	11/11/2022 13:51:32	11/11/2022 13:51:32	OK	0 (15)
FWD-13	INFLUXMET	http://192.168.15.237@WeatherMET	11/11/2022 13:51:32	11/11/2022 13:51:33	OK	0 (15)
FWD-14	INFLUXIMP	http://192.168.15.237@WeatherIMP	11/11/2022 13:51:32	11/11/2022 13:51:33	OK	0 (15)
FWD-15	WU	http://192.168.15.100/WU-upload.php?	11/11/2022 13:51:32		404	1 (15)
FWD-16	WU	http://192.168.15.100/AMB-upload.php?	11/11/2022 13:51:32		404	1 (15)
FWD-17	MT	http://192.168.15.100/MT?	11/11/2022 13:51:32		404	1 (15)
FWD-18	WC	http://192.168.15.100/MT?	11/11/2022 13:51:32		404	1 (15)
FWD-19	AWEKAS	http://192.168.15.100/AWEKAS?	11/11/2022 13:51:33		404	1 (15)
FWD-20	WEATHER365	http://192.168.15.100/weather365/	11/11/2022 13:51:33		404	1 (15)
FWD-22	EWUDP	192.168.15.255:12361	11/11/2022 13:51:33	11/11/2022 13:51:33	OK	0 (15)
FWD-23	APRS	cwop.aprs.net:14580	11/11/2022 13:51:33	11/11/2022 13:51:33	OK	0 (15)

---

store missed forwards and transfer them when the forward destination is available again (from v0.10)

store missed forwards and transfer them when the forward destination is available again (from v0.10)

Basically, the weather station sends the data to FOSHKplugin, which receives the data, processes it and forwards it to the configured destinations in the appropriate format.

If a destination cannot be reached, FOSHKplugin tries several times to deliver the data after all. If this does not succeed even after 3 attempts (each with a pause of 6 seconds longer), the data is discarded. This results in gaps in the data recording at the corresponding destination, which is annoying to say the least.

Reasons for non-accessibility can be a failure of the connection (Internet, LAN cable, WIFI-AP) or a (temporary) failure or maintenance of the server of the target system.

Very few weather services support the subsequent sending of data - they ignore the time contained in the data set (such as dateutc) and instead use the time of receipt of the data for temporal assignment.
With these systems, subsequent sending is therefore not possible - the current weather data would thus be overwritten with outdated values.

InfluxDB (both v1 and v2) use the time that is transferred with the data set instead. Thus, resends and resulting gapless records are possible here. FOSHKplugin stores failed forwards to these forward targets (INFLUXMET, INFLUXIMP, INFLUX2MET, INFLUX2IMP) temporarily and resends the data automatically when the remote station is reachable. This behaviour can be deactivated individually for each forward in the config file:

FWD_QUEUE = False

For each data set, a file FOSHKplugin-queue\FWD-nn\FOSHKplugin-queued-data-nn.YYMMDDhhmmss

is created in the directory of the Conf-File of FOSHKplugin, which is deleted after sending. The directory for saving these queue files can be set via

FWD_QDIR = /path/to/store/queue-files/

in the corresponding forward block.

Awekas at least offers a manual import option for the basic data - an Excel file in the specified format can be uploaded via a web form. FOSHKplugin can at least make the creation of this Excel file a little easier by saving the data in the correct order as a CSV. The CSV can then be opened with Excel and saved as XLSX to allow the easy import into Awekas.
Queuing for the forward type AWEKAS is activated by default. With one line

FWD_QUEUE = False

in the forward block, the generation of the CSV FOSHKplugin-queued-data-nn.csv can be deactivated. Here, too, the following can be specified

FWD_QDIR = /path/to/store/queue-files/

so that the file can also be accessed via file access (Samba).
Instructions for creating the Excel file can be found on the Awekas website.

Currently, the queue formats CMX and EW are supported in the forward type (FWD_TYPE) EW and RAWEW as output format in case of an impossible dispatch. CMX creates an accumulating file in the structure of a REALTIME.TXT, which unfortunately cannot be imported into CumulusMX yet:

FWD_QUEUE = CMX

The EW format is also experimental only (because it is not yet usable):

FWD_QUEUE = EW
where each record is saved in Ecowitt format in a separate file. Here, too, you can configure an output directory for these file(s) by specifying FWD_QDIR.

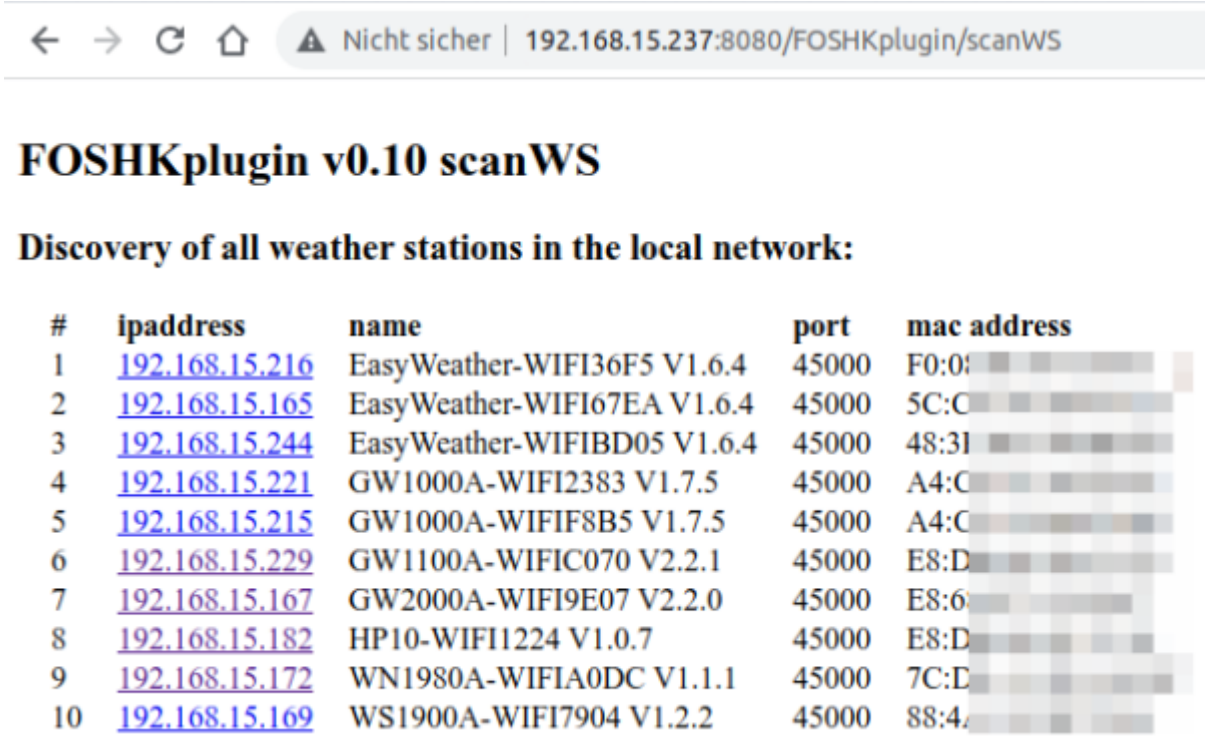
Other queue functions are currently not implemented.
However, as soon as I learn of other queue-enabled services or useful formats for queuing, I can implement them.
If you have any ideas or suggestions, please feel free to share them.

---

scan all weather stations in your network (from v0.10)

scan all weather stations in your network (from v0.10)

For all those who use several weather stations locally, FOSHKplugin from v0.10 on offers a new feature for web-based searches.
With the URL http//ipaddressport/FOSHKplugin/scanWS, all consoles accessible in the network segment are searched for via usual UDP broadcast and displayed as an overview page:



A click on the displayed ipaddress then leads directly to the WebUI of the console in question in a new browser tab - provided this console has a WebUI.

If I am unsure whether all consoles in the network are accessible or whether one has hung up, I use this option. Or if I need the MAC address for any API attempts.

---

custom push notifications (from v0.10)

custom push notifications (from v0.10)

With v0.10 there is the possibility to create own push notifications based on the values received or calculated by FOSHKplugin.

For each data set coming from the weather station, it is checked whether there is a reason for a warning via push notification.
The reason for a notification is the exceeding/falling below of the defined value range as well as the fact that no notification has yet been sent for this rule within a configurable time (hold time).

If this is the case, a message is sent and the time at which this notification was sent is logged (to prevent this message from being sent again and again).

With the next incoming data record, it is checked whether the configurable hold time has been exceeded and whether the value in question is outside the defined limit. However, there is only a (further) warning if the hold time has been exceeded; the hold time starts again from the beginning if there is a setpoint overrun/underrun.

The rules for the user-defined pushover messages are stored in the Config file in the [Pushover] area

and correspond to this structure:

```
PO_CUSTOMn = @key [<,<=,==,>,>=,<>,!=] value,message text,activation,hold
time.
```

The n for variable PO_CUSTOMn corresponds to a number from 1 to 99. Thus, a maximum of 99 rules can be configured.

The individual columns are separated by a comma. If a comma is required in the message text itself, it must be escaped with the character "\".
The comparison value may also be a key - this is then specified with @keyname.
The fields message text, activation and hold time are optional and are provided with the default values if not specified.
These defaults are:

```
message text: "condition "condition" is given!" (condition corresponds to
the comparison formula - for example @rainin > 0.55)
activation: True (thus this rule can also be deactivated with False if
required)
hold time: 3600 (this corresponds to 1 hour)
```

Only simple constructs are possible as a formula. A concatenation or more complex calculations is not intended.
The formula therefore always has the structure @key OPERATOR value. **The spaces between the individual components of the condition are mandatory!**
An attempt is always made to perform a numerical comparison first. If this is not successful, a string comparison (case-sensitive) is carried out instead.
Supported operators are <, ⇐, ==, =, >, >=, <> and != (whereby "=" and "==" as well as "<>" and "!=" are interpreted identically).

All keys recognised or generated by FOSHKplugin can be used for comparisons - both metric and imperial values and minmax values. Only the status messages cannot be used because they generate their own notifications.
A list of all keys that can be used at runtime can be called up via a web browser at the address http://ipaddress:port/FOSHKplugin/keyhelp - where ipaddress is the IP address of the host on which FOSHKplugin is running and port is the corresponding port.

If desired, the actual value can be sent in the message text with @value - before sending, @value is replaced by the measured value. The comparison value (i.e. the value after the operator) can be output in the message text with @comp. Please note that the "#" character cannot be used in the message text under any circumstances because it defines a comment. If you need the hash just replace it with a "$" - FOSHKplugin will convert it to a hash "#". The text colour can be defined within the message text with the usual html tag text. "color" corresponds to the html colours as a name (red, blue, green etc.) or as hex notation (see https://en.wikipedia.org/wiki/Web_colors). Please note that the hash character "#" cannot be used - use the dollar sign "$" instead.

Examples:

```
PO_CUSTOM1 = @tempc <= 2.5,Current temperature (@value°C) is equal or below
2.5°C!,False,3600
```

```
PO_CUSTOM2 = @tempf < 32,Current temperature (@value°F) is below
32°F!,True,86400
PO_CUSTOM6 = @stationtype != GW1000A_V1.7.5,Update detected\, former version
was v1.7.5 - now it's @value!
PO_CUSTOM99 = @tempc <= @dewptc,Temperature (@value°C) is lower than
Dewpoint (@comp°C)!,True
PO_CUSTOM51 = @temp1c < @temp2c,<font color="$ff0000">Temperature of WH31 $1
(@value°C) is lower than WH31 $2 (@comp°C)!</font>,True
```

A few final notes:
Pushover (global) as well as the custom warnings (only) can be activated and deactivated at runtime via web command and UDP. However, the current status and the hold time are not remanent - when FOSHKplugin is restarted, the configuration options stored in the config file become valid again. This control option therefore only serves to temporarily change the status.

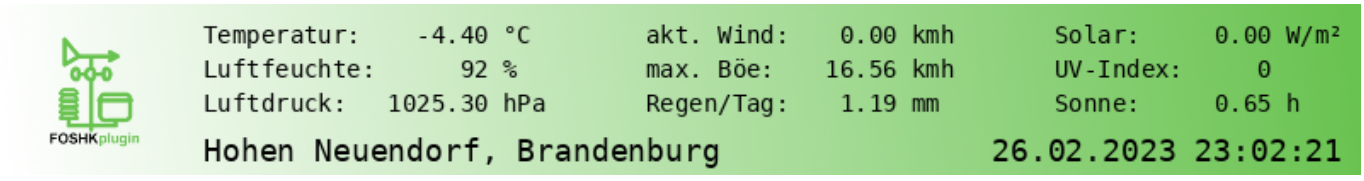
The http command to activate/deactivate the custom warnings is http://ipaddress:port/FOSHKplugin/customwaring=enable or disable (where ipaddress is the IP address of the host on which FOSHKplugin is running and port is the corresponding port).
Each push notification contains a link to the online help of FOSHKplugin - from there, the custom warning can be easily switched off for the rest of the runtime.

A temporary (de)activation is also possible via UDP at runtime using the UDP command Plugin.customwarning=enable or disable.

Creation of banners and stickers (from v0.10)

Creation of banners and stickers (from v0.10)

As a additional forward type (FWD_TYPE) the type BANNER was introduced in v0.10. With this type, images (PNG/JPG/GIF) can be created in which weather data (values) from FOSHKplugin can be embedded:



The generated images can be stored in the file system or sent to a remote server via FTP or http/POST. There, they can be accessed from anywhere, integrated into your own web pages or used as footers in any forum postings (if allowed).
So if you want to display your weather data on your own homepage, you don't have to go the complicated way of sending the data to the web server and having it processed by a server-specific read-in script.
One simply defines an image on the homepage that is automatically exchanged without further programming knowledge - in each case with the current data of the weather station. No further adjustments are necessary for the various CMSs; Joomla, Wordpress, Typo3 etc. simply display an image that updates itself.

These images can have an existing image but also a definable colour as background. This colour can also be transparent, so that only the text of the image is visible and the original background remains. With the option rounded_corners = True you can create rounded corners in the image formats PNG and GIF - regardless of whether the background is an image or a colour. The radius of the rounding can be specified with the corresponding number instead of True. In the standard (True), the value 10 is taken, which corresponds to a small radius and thus a small rounding.

Any image objects (such as logos) can be positioned within the banner.
These logos can also be output specifically depending on certain values. This opens up interesting possibilities for "dynamic" banners.
An example:
There is the FOSHKplugin-internal key wnowlvl, which outputs the current weather situation based on the air pressure. We don't need to discuss the quality of this statement - but here it provides a nice visual example.
According to the definition - depending on the weather situation - the following codes are output:

wnowlvl = 0 –> stormy, rain
wnowlvl = 1 –> rainy
wnowlvl = 2 –> changeable
wnowlvl = 3 –> sunny
wnowlvl = 4 –> dry, thunderstorm

In the logo configuration of the banner defnition, you can now define that a different logo is used for each individual status:

```
logo_1 = 10,100,storm.png,@wnowlvl = 0
logo_2 = 10,100,rain.png,@wnowlvl = 1
logo_3 = 10,100,change.png,@wnowlvl = 2
logo_4 = 10,100,sun.png,@wnowlvl = 3
logo_5 = 10,100,tstorm.png,@wnowlvl = 4
```

Thus, different images (defined in the 3rd column) are displayed depending on the data!
The 4th column of the logo lines is used to determine whether the logo is shown in the banner. If this column is missing or empty, the logo is integrated into the banner.
Otherwise, FOSHKplugin checks the content of the 4th column for validity. Fields to be queried are marked with an "@". You can also compare a field with another field, for example @temp1f ⇐ @tempf.

Permissible operators are: [<,⇐,==,>,>=,<>,!=]
Only if the check is successful, the logo is included in the banner.

These queries can be used to display a specific logo in the special cases of an empty value or a key that does not exist:

```
logo_6 = 10,100,emptyvalue.png,@wnowlvl =       # empty value
logo_7 = 10,100,dontexist.png,@wnowlvl = null   # key does not exist
```

With the options pre_count and dec_count, the length of the (dynamic) output string (pre_count) and the number of digits after the decimal point (incl. rounding) can be defined for each line type. This leads to a better positioning of the output values, should it be necessary.

All FOSHKplugin-internal values can be used for integration into the image - thus values with metric or imperial units can be output (also simultaneously).
Access to internal status values of FOSHKplugin as well as min/max values is also possible. In addition to the keys displayed under http://ipaddress:port/FOSHKplugin/keyhelp, the following keys can also be processed for banners & stickers:

prgname –> program name (FOSHKplugin)
prgver –> program version (v0.10)
winddir_text –> wind direction as string like "N", "NNE", "NE", "ENE", "E", "ESE", "SE", "SSE", "S", "SSW", "SW", "WSW", "W", "WNW", "NW", "NNW" or the German abbreviations for them
pchange1in –> change of air pressure within the last hour in inHg (pchange1 displays this in hPa)
pchange3in –> change of air pressure within the last 3 hours in inHg (pchange3 displays it in hPa)
lightningmi –> distance of last lightning in miles (instead of km)

Special characters such as German umlauts can be used - provided the font used supports them. A special character, however, is the comma ",". Since this is used within the configuration to separate the individual fields, it must be escaped as a string with a backslash: "," thus becomes "\,".
Examples of this can be found on the banner demo page specially set up for this purpose: https://foshkplugin.phantasoft.de/banner.

The embedding of the values is based on a positioning of rows per y-coordinate, which contain individual columns that can be positioned per x-coordinate. The number of rows (per type) and columns is limited to 100.
For each image, 10 (!) different fonts with different font, size and colour can be used: Header, Line, Footer, Special and Custom, Custom1..5. The names are only a distinguishing feature, the contents are shaped by the definition.
Overlapping of fonts is possible without any problems, as each line is positioned in the area of the background by x,y coordinates.

The availability of fonts depends on many factors: Linux distribution, installed software, manually installed fonts.
The command

```
sudo ls -laR /usr/share/fonts/truetype/
```

should display all existing fonts in the system. These file names should then also be usable as font names. If necessary, however, additional fonts (even the Microsoft fonts) can be installed. A few hints on installing the fonts can be found below. If the defined font is not found, the system falls back to DejaVuSansMono.ttf.

The font size to be used depends primarily on the size of the image in which you want to integrate your text. A very large font naturally requires a very large image. When it comes to banners or stickers, the images are rather small and then naturally require a smaller font size. However, it would not be a problem to add pictures of your own webcam with the data of your own weather station.

When specifying the colour definition for fonts and also backgrounds, most colour names such as black, red, yellow, green, ... but also colour specifications in html notation can be configured. When specifying the html notation, however, the "$" must be entered instead of the character "#".

There is a predefined variable $datetime that "stamps" the local time of image creation into the

image in a definable format (Python-like: "%d.%m.%Y %H:%M:%S" or "%y/%m/%d %I:%M %p"). The output language of the months and weekdays can be set to German via locale_format = de_DE.UTF-8 or to US English (or any other installed locale) via locale_format = en_US.UTF-8 in the banner configuration file. The language packages installed in the Linux system that can be entered here can be displayed with the command

```
sudo locale -a
```

In the FOSHKplugin configuration file foshkplugin.conf, a usual forward with a unique number of the type "BANNER" (FWD_TYPE = BANNER) is to be created.
The actual banner configuration file is specified as FWD_OPTION with bannerconfig=configname.conf. This separation into FOSHKplugin configuration file and banner definition has the advantage that the banner can also be adapted at runtime by FOSHKplugin. Especially in the development of a banner, several attempts are usually necessary to position the banner components perfectly or to select the "right" font.
This can be done without having to restart FOSHKplugin each time.
Note that FWD_URL contains the destination where the generated image file is to be stored via http/POST or FTP - the actual file name is defined in the banner configuration. The access data required for the upload are defined with FWD_SID and FWD_PWD.
Even if the output file is intended for upload - a local file is always (!) created first, which is specified with image_name in the banner configuration.
Please also consider that without specifying a specific FWD_INTERVAL, a corresponding banner will be generated for EVERY input of data from the weather station. For example, if the weather station transmits at 16-second intervals, a new banner image will be generated every 16 seconds (!) and - if configured - uploaded to the upload destination. This can cause considerable traffic!
You should find a good balance between the timeliness of the banner images and the traffic. 60 seconds (FWD_INTERVAL = 60) should be sufficient. But 5 minutes (i.e. 300 seconds - FWD_INTERVAL = 300) can certainly still be called up-to-date.

A forward would look like this in foshkplugin.conf, for example:

```
[Forward-19]
FWD_ENABLE = True                                                    # you may
disable the forward instead of deleting
FWD_CMT = Banner-Test Simple                                         # comment
field - for your info
FWD_OPTION = bannerconfig=banner-simple.conf                         # file with
banner definition: bannerconfig=/path/to/configfile
FWD_URL = ftps://my.domain.tld/httpdocs/img/                         # you may
specify a local path, a ftp or http address
FWD_INTERVAL = 60                                                    # in seconds
- if not specified a banner image will be created/sent on every weather
station interval
FWD_IGNORE =                                                         # not really
usefull in the BANNER context
FWD_TYPE = BANNER                                                    # use BANNER
for creating banners/stickers
FWD_SID = ftpuser                                                    # user name
for ftp/http access
FWD_PWD = ftppasswd                                                  # password
for uploading via ftp/http
```

```
FWD_EXEC =                                                  # a script to
be started before image creation - e.g. to add custom key=value entries
```

And the banner definition in banner-simple.conf then as follows:

```
[Banner]
# this is just a demo banner definition to see the syntax and possibilities
# there are 10 definable fonts: header, line, special, footer, custom,
custom1, custom2, custom3, custom4 and custom5
# each font definition contains information on font, colour, size, decimal
places (dec_count) and string length (pre_count)
# you may specify up to 100 lines (n) per font definition, where y=Y-
coordinate, X-coordinate for name, name, position for value, value and unit
# each line may contain up to 100 coloumns containing the position of the
text, the text itself as well as the position of the value, the value and
the unit
# fontdefinition_n = y,key_pos,key,val_pos,value,uni,
key_pos,key,val_pos,value,uni, key_pos,key,val_pos,value,uni,
key_pos,key,val_pos,value,uni, key_pos,key,val_pos,value,uni

image_name = banner-simple.png                              # file name
of output image - may be gif, png, jpg, ... and may contain a path
image_width = 300                                           # only valid
if no image is selected as background
image_height = 100                                          # only valid
if no image is selected as background
image_background = white                                    # may be
transparent, any color, http color code (with $ instead of #) or a filename
of an image file
dtime_format = "%y/%m/%d %I:%M %p"                          # use
"%d.%m.%Y %H:%M:%S" for european 24h display - Python like - see
https://www.w3schools.com/python/python_datetime.asp
locale_format = "de_DE.utf8"                                # use
en_US.utf8 for English terms in dates
rounded_corners = True                                      # rounded
corners; available for png/gif only, you may specify the radius by n instead
of True (default: 10)
border_width = 0                                            # width of
the border in pixels
border_color = black                                        # may be any
color name or http colorcode (with $ instead of #)

#logo_1 = y,x,name
logo_1 = 10,250,foshkplugin.png                             # define logo
file name and position - y,y,name - up to 100 logos possible

header_font_name = verdana.ttf                              # if the
specified font is not available the standard font DejaVuSansMono.ttf will be
used
header_font_color = blue                                    # may be any
```

```
color name or the http color code (use $ instead of #)
header_font_size = 8                                            # size in
pixel
#header_1 = y,key_pos,key,val_pos,value,uni, key_pos,key,val_pos,value,uni,
key_pos,key,val_pos,value,uni, key_pos,key,val_pos,value,uni,
key_pos,key,val_pos,value,uni
header_1 = 0,10,$datetime,,,,,,,,                               #
header_0..99 - 100 possible lines with header definition; $datetime uses the
current local time
header_pre_count =                                              # length of
the output value - padded with spaces if necessary
header_dec_count =                                              # default of
decimal places - will be rounded
header_dtime_format =                                           # uses the
format defined here for time/date outputs of this font (see dtime_format in
the global assignment).

line_font_name = verdanab.ttf
line_font_color = black
line_font_size = 14
#line_1 = y,key_pos,key,val_pos,value,uni, key_pos,key,val_pos,value,uni,
key_pos,key,val_pos,value,uni, key_pos,key,val_pos,value,uni,
key_pos,key,val_pos,value,uni
line_1 = 15,10,Temperatur:,140,tempc, °C,,,,,,,,,,
line_2 = 35,10,Luftfeuchte:,140,humidity, %,,,,,,,,,,
line_3 = 55,10,Luftdruck:,140,baromrelhpa, hPa,,,,,,,,,,

footer_font_name = verdana.ttf
footer_font_color = black
footer_font_size = 16
#footer_1 = y, key_pos,key,val_pos,value,uni, key_pos,key,val_pos,value,uni,
key_pos,key,val_pos,value,uni, key_pos,key,val_pos,value,uni,
key_pos,key,val_pos,value,uni
footer_1 = 80,10,Hohen Neuendorf\, Germany,,,,,,,,

special_font_name = verdana.ttf
special_font_color = black
special_font_size = 16
#special_1 = y, key_pos,key,val_pos,value,uni,
key_pos,key,val_pos,value,uni, key_pos,key,val_pos,value,uni,
key_pos,key,val_pos,value,uni, key_pos,key,val_pos,value,uni
#special_1 = 80,10,Hohen Neuendorf\, Germany,,,,,,,,

custom_font_name = verdana.ttf
custom_font_color = black
custom_font_size = 16
#custom_1 = y,key_pos,key,val_pos,value,uni, key_pos,key,val_pos,value,uni,
key_pos,key,val_pos,value,uni, key_pos,key,val_pos,value,uni,
key_pos,key,val_pos,value,uni
```

For testing purposes and as an overview, there is an http overview page that can be reached at

http://ipaddress:port/FOSHKplugin/banner. All images generated by existing conf files (with [Banner] stanza) are displayed there.
Each of the images displayed contains a link to the actual image. There are two different links - image and config. image simply displays the image generated by the given interval (FWD_INTERVAL); config generates a new image every time it is called and displays it - but it is not sent to the destination via ftp/http outside the usual interval.

For creating and optimising banners, it is recommended to start the web page in the background with the banner to be created and a refresh (add &refresh=2 to the URL) and to edit the actual conf file in the foreground - this way you can see the changes made immediately and can position the individual elements very easily.
In the case of transparent banners or also to evaluate the effect of the colouring, the background colour of the web page can be changed with adding &background=grey (or any html colour code - where the "#" character must be replaced by "$") to the URL.

Even if there are problems with incorrect assignments, missing logos or fonts, FOSHKplugin always tries to generate the appropriate image. For the wrong or faulty components, a warning is given in the snd-foshkplugin.log.
Common problems are missing fonts or logos or backgrounds to be embedded. Also the wrong specification of colours for background and font can lead to warnings or not expected results due to the changed syntax ($ instead of #) or typos in the colour definition.
However, these problems should be solved quickly with the tips in the log file and this description.

Problem solving:
If, despite specifying the desired locale (locale_format), the weekdays or months are still not displayed in the expected language, this locale must first be installed. On Debian-based systems, this is done by ticking the corresponding locale after calling

```
sudo dpkg-reconfigure locales
```

.

Additional notes:
Font compatibility:
https://www.linux-magazine.com/Online/Blogs/Off-the-Beat-Bruce-Byfield-s-Blog/Free-equivalents-for-standard-proprietary-fonts

Microsoft fonts:
https://itsfoss.com/install-microsoft-fonts-ubuntu/

free DejaVu fonts:
https://dejavu-fonts.github.io/

free Logos:
Free (public domain) logos for integration into your own banners can be found here. These images can be downloaded in any resolution and colour.
Google also has an extensive collection of fonts and icons that may be used free of charge.
Freely usable background founders for banners can be found at Pexels or Unsplash, among others.
However, you should always make sure that the fonts, icons, logos and backgrounds used can be used legally. I assume no liability!

Definition of user-defined output formats - TAGFILE (from v0.10)

Definition of user-defined output formats - TAGFILE (from v0.10)

For formats that are not (yet) natively supported by FOSHKplugin itself, with v0.10 there is a practicable solution for user-defined output formats with the TAGFILE-Forward.
Here, placeholders within a file - so-called tags - are replaced by FOSHKplugin with the actual values of the weather station.

Example:
The (fictitious) weather service blaWeather would like to have the data transmitted in a custom line-oriented file format - where the first line must contain the outdoor temperature (metric), the second line the minimum temperature as well as the time of the minimum temperature, and the 3rd line the maximum temperature as well as its time:

```
tempc
tempc_min,tempc_min_time
tempc_max,tempc_max_time
```

However, FOSHKplugin does not natively support the output in this format. With a TAGFILE forward, however, this can be solved very easily!
You create a template file with the corresponding structure and the appropriate field assignment:

```
[[@tempc]]
[[@tempc_min]],[[@tempc_min_time]]
[[@tempc_max]],[[@tempc_max_time]]
```

and a TAGFILE forward and thus FOSHKplugin sends the generated file at the selected interval via http(s)/GET or http(s)/POST or via FTP(S).

If you want to integrate your own web pages via iframe or provide complex web pages with current data of the weather station, you can do this as well.
The tag definition is user-defined (per forward) - so you can also use an http comment like <!– @keyname –>. As long as FOSHKplugin has not replaced the tags, they are not visible on the html page.
However, when FOSHKplugin generates the page, the corresponding values are then included instead of these html comments. In the definition, @keyname must be included; however, in the template file, keyname has to be replaced by the desired key. A list of all usable keys can be queried at any time via http://ipaddress:port/FOSHKplugin/keyhelp - where ipaddress and port correspond to the host and port on which FOSHKplugin is running.

The website can be as complex as you like. I have put a simple example for demonstration here. The page with the measured values is automatically created by FOSHKplugin and stored on the web server via FTPS.

Storing in the file system (as well as on a remote server) as a growing file (e.g. CSV file) is also possible without any problems thanks to the append function.
The TAGFILE function is not limited to files - weather services can also be supplied with data in the

desired format directly via http/GET or http/POST.

Important!
The tag definition is REPLACED by FOSHKplugin in the output file. In this respect, the tag definition must be unique and must not occur in the normal file content!
In addition, it should be noted that the hash character "#" cannot be used in the tags - it is recognised as a comment prefix and all characters behind it are ignored.

As usual, the basis is a forward definition in foshkplugin.conf, in which the tag definition file is specified within the FWD_OPTION line:

```
[Forward-29]
FWD_ENABLE = True                                                    # you may
disable the forward instead of deleting
FWD_CMT = TAGFILE-Test tag-definition                                # comment
field - for your info
FWD_OPTION = config=tag-definition.conf                              # file with
tag definition: config=/path/to/configfile
FWD_URL = ftps://my.domain.tld/httpdocs/img/                         # you may
specify a local path, a ftp or http address
FWD_INTERVAL = 60                                                    # in seconds
- if not specified a forward will be executed/sent on every weather station
interval
FWD_IGNORE =                                                         # not really
usefull in the TAGFILE context
FWD_TYPE = TAGFILE                                                   # use TAGFILE
for creating tagged files
FWD_SID = ftpuser                                                    # user name
for ftp/http access
FWD_PWD = ftppasswd                                                  # password
for uploading via ftp/http
FWD_EXEC =                                                           # a script to
be started before saving the output - e.g. to add custom key=value entries
```

Here, with config=tag-definition.conf in the FWD_OPTION line, the tag-definition.conf file is stored as the tag definition.
As with the BANNER forward, we also use a second definition file here to allow changes to the definitions during the runtime of FOSHKplugin.

In the tag-definition file there are a few necessary configuration points and several optional setting options for the tag definition:

```
infile =                                                             # mandatory!
- name of the template file with embedded tags (default: "")
outfile =                                                            # name of the
final output file (default: "")
append = False                                                       # determines
whether outfile is overwritten or appended in each interval (default: False)
task = save                                                          # sets final
processing - SAVE for writing a file, GET for sending via http(s)/GET, and
```

```
POST for sending via http/POST (default: save)
tag = [[@keyname]]                                                  # defines the
structure of the tag - this type of string is searched for in the infile and
replaced with the corresponding value (default: <!-- @keyname -->)
postscript =                                                        # possibility
to specify a bash script to be started after saving the output file but
before sending it (default: "")
dtime_format =                                                      # sets output
format for values with a date/time - use "%d.%m.%Y %H:%M:%S" for european
24h display - Python like - see
https://www.w3schools.com/python/python_datetime.asp (default: %d.%m.%Y
%H:%M:%S)
locale_format =                                                     # defines the
language of the names of the weekdays and months - use en_US.utf8 for
English terms in dates (default: "")
pre_count =                                                         # length of
the output value - padded with spaces if necessary (default: "")
pre_fill =                                                          # string to
prepend (default: space) - could be   for html files
dec_count =                                                         # number of
decimal places (default: "" = as is)
dec_separator default =                                             # you may
define a comma as a decimal separator (default: .)
```

If set, the specifications of the tag definition override those of the settings configurable under the same name in FWD_OPTION (separated by commas).

The file defined in the tag definition per infile is the template that defines the output format with the positions of the values of the weather station:

```
time;aqtime;temperature;temp min;temp min time;temp max;temp max
time;humidity;pressure;pressure min;pressure min time;pressure max;pressure
max time
[[@time]];[[@aqtime]];[[@tempc]];[[@tempc_min]];[[@tempc_min_time]];[[@tempc
_max]];[[@tempc_max_time]];[[@humidity]];[[@baromrelhpa]];[[@baromrelhpa_min
]];[[@baromrelhpa_min_time]];[[@baromrelhpa_max]];[[@baromrelhpa_max_time]]
```

So in the context of the TAGFILE forward, 3 files are relevant:

- foshkplugin.conf - definition of the actual forward, the interval, the destination for sending the data and the specification of the file name of the tag definition (FWD_OPTION = config=tagdefinition).
- a file of any name (here in the example: tag-definition.conf) in which the assignment of the tag definitions as well as the specification of the output file name is made.
- a template file with the structure of the target file and the placeholders for the weather station data (infile)

In order not to get confused with this multitude of files, I recommend as a best practice: name the template file *.template and the tag definition file *.tagdef.

Another example to illustrate the TAGFILE function can be found here with the pool web page

automatically generated by FOSHKplugin. It combines the multiple wishes for user-specific and "large" values as a web page and can thus be used as a template for own solutions. The three configuration files required for this forward are:

Entry in foshkplugin.conf:

```
[Forward-38]
FWD_ENABLE = True
FWD_CMT = TAGFILE Pool via FTP
FWD_URL =
ftps://foshkplugin.phantasoft.de/httpdocs/foshkplugin.phantasoft.de/tagfile/
FWD_OPTION = config=tag-pool.conf
FWD_INTERVAL = 60
FWD_IGNORE =
FWD_TYPE = TAGFILE
FWD_SID = ftp-username
FWD_PWD = ftp-password
FWD_EXEC =
```

The TAG configuration of config=tag-pool.conf:

```
[Tagfile]
infile = tag-pool.template
outfile = index-pool.html
append = False
task = POST
tag = <!--@keyname-->
locale_format = de_DE.UTF-8
postscript =
dtime_format = %d.%m.%y %H:%M:%S
pre_count = 4
pre_fill =  
dec_count = 1
dec_separator = .
```

And finally the template file tag-pool.template:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>FOSHKplugin v0.10 TAGFILE demo</title>
    <meta charset="UTF-8">
    <meta name="mobile-web-app-capable" content="yes">
    <meta name="apple-mobile-web-app-capable" content="yes">
    <meta name=apple-mobile-web-app-title content="Personal Weather
Station">
    <meta name="viewport" content="width=device-width, height=device-height,
initial-scale=1, viewport-fit=cover">
    <meta name="theme-color" content="#ffffff">
    <meta http-equiv="refresh" content="30">
```

```
    <link rel="icon" type="image/png"
href="
IAAAAAAAAAAQAAMMOAADDDgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAIAAAAFAAAABgAAAAYAAAADAAAA
A0CbUw9QxGoJUMNpBVDDaRpQw2kOUMNpAlDDaQBQw2kAAAAAAAAAAAUAAAA7AAAAQwAAAFMAAABi
AAAAVAAAAF0wdT9YUcZrYlDDaVlQw2lwUMNpZlDDaUZQw2kHUMNpAAAAAAAAAAADAAAAPAAAADwE
CgVDCRYMTwwdEEQZPiFrK2o5OFLHazlQw2kuUMNpJVDDaTBQw2kkUMNpA1DDaQAAAAAABQUFAE3z
cQBRqGQDUcVqZVHFaoJRxmt3UcVqrlDDaYZQw2mXUMNpm1DDaTtQw2kQUMNpAFDDaQAAAAAAAAAA
AFDDaQBQw2kAUMNpXVDDab5Qw2mZUMNpIVDDaXlQw2lzUMNpiFDDaX1Qw2lzUMNpfVDDaQ5Qw2kA
AAAAAAAAAABQw2kAUMNpAFDDaYBQw2nVUMNpuVDDaRlQw2lxUMNpe1DDaSlQw2kAUMNpAlDDaXJQ
w2kbUMNpAAAAAAAAAAAAUMNpAFDDaQBQw2l/UMNp1FDDabhQw2kZUMNpclDDaZBQw2mEUMNpaFDD
aWxQw2miUMNpH1DDaQAAAAAAAAAAFDDaQBQw2kAUMNpfVDDadNQw2m1UMNpGFDDaXFQw2liUMNp
qFDDaaVQw2moUMNpiVDDaQxQw2kAAAAAAAAAABQw2kAUMNpAFDDaSlQw2lYUMNpRFDDaQpQw2l1
UMNpFlDDaQ5Qw2kPUMNpD1DDaQZQw2kAUMNpAAAAAAAAAAAAAAAAAAAAABQw2kAUMNpF1DDaX5Q
w2lfUMNppVDDaVdQw2mCUMNpKlDDaQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABQw2kAUMNpAFDD
aStQw2mzUMNpolDDabFQw2mUUMNpuFDDaUJQw2kAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAUMNp
AFDDaSlQw2mIUMNpV1DDaR5Qw2mLUMNpJFDDaUJQw2kzUMNpA1DDaQAAAAAAAAAAAAAAAAAAAA
AAAAAFDDaQBQw2k4UMNpgFDDaZhQw2mCUMNpl1DDaXpQw2mWUMNpxlDDaSpQw2kAAAAAAAAAAAA
AAAAAAAAAAAAABQw2kAUMNpN1DDaZ5Qw2lpUMNpElDDaQxQw2kMUMNpIlDDaTRQw2kGUMNpAAAA
AAAAAAAAAAAAAAAAAAAAAUMNpAFDDaTBQw2mIUMNpD1DDaQAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFDDaQBQw2kGUMNpClDDaQBQw2kAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAgAEAAIABAACAAQAA4AcAAOADAADgIwAA4AMAAOADAADg
BwAA8B8AAPAfAADgDwAA4A8AAOAPAADj/wAA5/8AAA==">
    <style>
      html { background: transparent url(img/pool.jpg) no-repeat fixed
center; background-size: cover; background-attachment: fixed; margin: 0;}
      table { width: 100%;}
      td { width: 25%; border-radius: 12px; background-color: rgba(80,80,
80, 0.6);}
      .bigleft     { font-family: sans-serif; font-size: 128px; color:
white;  text-align: left; padding-left:25px; padding-right:25px;}
      .bigmiddle   { font-family: sans-serif; font-size: 128px; color:
white; text-align: center;}
      .bigright    { font-family: sans-serif; font-size: 128px; color:
white; text-align: right; padding-left:25px; padding-right:25px;}
      .smallleft   { font-family: monospace; font-size: 25px; color: white;
text-align: left; padding-left:25px; padding-right:25px;}
      .smallmiddle { font-family: monospace; font-size: 25px; color: white;
text-align: center; width: 100%;}
      .smallright  { font-family: monospace; font-size: 25px; color: white;
text-align: right; padding-left:25px; padding-right:25px;}
    </style>
  </head>
  <body>
    <div class="smallmiddle">FOSHKplugin v0.10 TAGFILE demo<br><!--@aqtime-
-><br><br></div>
    <table class="bg-table">
      <tr>
        <td>
          <div class="smallleft">Außen</div>
          <div class="bigleft"><!--@tempc-->°C</div>
          <div class="smallleft">&#9660; <!--@tempc_min-->°C, <!--
```

```
@tempc_min_time--><br>&#9650; <!--@tempc_max-->°C, <!--@tempc_max_time-
-></div>
        </td>
        <td style="background-color: unset;"> </td>
        <td style="background-color: unset;"> </td>
        <td>
          <div class="smallright">Pool</div>
          <div class="bigright"><!--@tf_ch2c-->°C</div>
          <div class="smallright">&#9660; <!--@tf_ch2c_min-->°C, <!--
@tf_ch2c_min_time--><br>&#9650; <!--@tf_ch2c_max-->°C, <!--
@tf_ch2c_max_time--></div>
        </td>
      </tr>
    </table>
  </body>
</html>
```

If you have a good knowledge of html and CSS, you can use this function to generate attractive static pages that are permanently updated by the refresh function. And all this without any complex server configuration or PHP scripts - it is simply a web page.

Sometimes it is not enough to just display the values of fields - i.e. @keyname - these may need to be modified beforehand.
FOSHKplugin has implemented a few commands (function blocks) for such special cases, which modify the values before replacing the tags.
Within the tags, a function block is enclosed in square brackets []. Parameters within the functions are separated by commas.

These commands are currently implemented:

substr/copy
only outputs part of the output string; the character count starts at 1
Call: substr(@keyname,startposition,count of chars)

round
rounds the value and reduces the decimal places if necessary; if there are 0 decimal places, the value is rounded and output as an integer
Call: round(@keyname,number of decimal places)

replace
replaces any character string "what" in the value of keyname with the character string "with"
Call: replace(@keyname,what,with)

fillleft
inserts as many characters "with" at the end of the value until the value of keyname corresponds to the length "len"
Call: fillleft(@keyname,with,len)

fillright
inserts as many characters "with" before the value of keyname until it corresponds to the length "len"
Call: fillright(@keyname,with,len)

addleft
inserts a number "count" of the character "with" to the left of the value of keyname
Call: addleft(@keyname,with,count)

addright
inserts a number "count" of the character "with" to the right of the value of the keyname
Call: addright(@keyname,with,count)

strip
removes spaces at the beginning and end of the output string of keyname
Call: strip(@keyname)

concat
joins any number of substrings to form an output string
Call: concat(str1,str2,str3)

dtime
converts a Unix timestamp into a defined human-readable format (format - corresponds to Python notation - see https://www.w3schools.com/python/python_datetime.asp)
Call: dtime(@keyname,format)

tdiff
converts a number of seconds in keyname to days, hh:mm:ss where the format can be specified with format (e.g. %j days %H:%M:%S)
Call: tdiff(@keyname,format)

onempty
replaces an empty return value with the specified string
Call: onempty(@keyname,string)

onvalue
appends the string "string" if the return value is not empty
Call: onvalue(@keyname,string)

dewptF
Calculates the dew point in °F for an imperial temperature value @keyname-temp based on the specified humidity value @keyname-humidity
Call: dewptf(@keyname-temp,@keyname-humidity)

dewptC
Calculates the dew point in °C for a metric temperature value @keyname-temp based on the specified humidity value @keyname-humidity
Call: dewptc(@keyname-temp,@keyname-humidity)

if
string "then" or "else" is output depending on the comparison between two operands (e.g. @keyname) with an operator
Call: if(@keyname,operator,@keyname,then,else)
Operators are "<", "⇐", "==", ">=", ">" and "!="

eval

Simple basic arithmetic function of two operands with one operator ("+","-", "*","/"); the non-rounded result of the calculation is output
Call: eval(@keyname,operator,@keyname)

The functions can also be nested - so you can also use combinations of functions. However, the correct bracketing and the order of the functions must be observed.
Examples:
concat(@lightning_time," oOo ",copy(@lightning_time,1,10)," oOo ",copy(@lightning_time,12,10))
fillleft(substr(@model,1,3)," ",20)
onEmpty(onValue(Round(@soilmoisture8,0),"%"), "n.a.")
<!–@[tdiff(@runtime,%j days %H:%M:%S)]–>
<!–@[dtime(eval(@aqtime,-,@runtime),%x %X)]–>

To implement such a function block into e.g. web page just use the configure tag definition e.g.
<!–@[onEmpty(onValue(Round(@soilmoisture8,0),"%"), "n.a.")]–>
I have published another example of the TAGFILE function including the additional functions here.
The necessary adjustments in the config file and the tagfile definitions can be downloaded as a ZIP package here.
The basic settings pre_count, pre_fill, dec_count and dec_separator defined in the tag configuration are ignored in the functions, as these are defined in the functions.

Attention!
All commands are case-insensitive, so you can use fillLeft or fillleft or FillLeft or any other combination. However, it should be noted that the operands - such as key names - are case-sensitive! For the correct key names, see http://ipaddressoffoshkplugin:port/FOSHKplugin/keyhelp

---

Replacing the console hardware while maintaining an existing database

Replacing the console hardware while maintaining an existing database

Sometimes it is necessary to replace the console hardware - for example because a device is defective or you have upgraded your hardware.
When sending to services where you cannot change the user credentials yourself because it is tied to the hardware (such as the PASSKEY for Ecowitt or the MAC address for Ambient Weather), the previous database is unfortunately not continued, but a new one is created.
A database that may have grown over a long period of time is therefore not continued - you start again from 0. What a pity!

FOSHKplugin offers the option of sending the data under the name of the old console.
The PASSKEY or the MAC address is exchanged when sending to the weather service via FWD_SID.

An example for Ecowitt.net:
A GW1000 with PASSKEY 000102030405060708090A0B0C0D0E0F was previously in use, which is now

being replaced by a GW2000.
As the new device itself would of course send with its own PASSKEY, internal sending to Ecowitt must be deactivated in the device. Instead, the custom server function is used to send the data to FOSHKplugin.
A forward in EW format to Ecowitt.net can then be created in FOSHKplugin, in which the PASSKEY is set to the old PASSKEY via FWD_SID:

```
[Forward-33]
FWD_CMT = sending to Ecowitt.net with adjusted PASSKEY
FWD_SID = 000102030405060708090A0B0C0D0E0F
FWD_IGNORE =
FWD_INTERVAL = 60
FWD_TYPE = EW
FWD_URL = http://cdnrtpdate.ecowitt.net/data/report/
FWD_ENABLE = True
```

Similarly, a console that transmits to Ambient Weather can also be replaced. If the previous MAC address was 00:01:02:03:04:05, this must be entered as FWD_SID in a forward in Ambient Weather format:

```
[Forward-44]
FWD_CMT = Forward to Ambient Weather
FWD_SID = 00:01:02:03:04:05
FWD_IGNORE =
FWD_INTERVAL = 60
FWD_TYPE = AMB
FWD_URL = https://api.ambientweather.net/endpoint?
FWD_ENABLE = True
```

First steps after installing FOSHKplugin

First steps after installing FOSHKplugin

Normally you install FOSHKplugin for a specific reason - for example, because you want to send data to a destination (weather service) that is not supported by the weather station hardware.
The installation is usually done quite quickly and the recipe suitable for the purpose is quickly found and entered as a forward.
However, there are a few more things that are necessary or at least helpful after installing FOSHKplugin and configuring the forward.

I recommend going through the configuration file foshkplugin.conf or foshkplugin.conf.orig (which contains a few notes on the configuration items) line by line.
The configuration options entered in the foshkplugin.conf file are decisive for FOSHKplugin - foshkplugin.conf.orig is used exclusively for documentation purposes.

In the **Config** section, this mainly concerns the items USE_METRIC, UDP_ENABLE, DT_FORMAT and LANGUAGE.
I live in the metric world - therefore the output is provided in metric values by default. The imperial

measurement system can be selected with USE_METRIC = False.
If you want to reduce data traffic in the local network, you should deactivate the UDP transmission of the values with UDP_ENABLE = False, unless this is necessary.
And if you do not want weather status messages in German, the output language can be defined under LANGUAGE. EN, NL, SK, FR and ES are currently supported.
The output format for date and time can be customised using DT_FORMAT.

In the **Warning** section, I recommend taking a look at the SENSOR_MANDATORY configuration item. All keys that must be present for incoming data from the weather station are entered here. If one of these keys is missing, a warning is issued.
This can be used to detect and alert the failure of a sensor.
The default setting is wh65batt. However, if you do not have a WH65/WS69 but a WS90, wh90batt (the battery value of a WS90) should be entered here instead. If you want to monitor the presence of other sensors, you can also enter the corresponding key names here, separated by commas.

For an improved sunshine hours calculation and for forwarding to some weather services, the geographical altitude (in meters) and the longitude and latitude (decimal degrees) of the solar sensor must be entered in the **Coordinates** section.
If you want to use the improved sunshine hours calculation, the configuration item SUN_CALC should also be set to True under section **Sunduration**.

The **CSV** section may also be of interest if the data is to be further processed in Excel, for example.

Otherwise, I recommend using Pushover to receive push notifications about important information from FOSHKplugin. Especially because user-defined push notifications are possible - for example, if the temperature exceeds or falls below a specified level, if it starts to rain or if the temperature falls below the dew point and there is a risk of mould.
You are informed within a few seconds!
The built-in http://foshkpluginhost:port/FOSHKplugin/help help page is also helpful. It lists all the URLs supported by FOSHKplugin's built-in web server.

As the configuration file is only read and activated when FOSHKplugin is started, FOSHKplugin must be restarted after changes to the configuration.

Finally, I strongly recommend checking the firmware versions of the console and any sensors and updating them if necessary. EAR offers assistance here.
The air pressure should also be calibrated correctly.

What else can you do with FOSHKplugin?
Useful functions include banners and tag files. The query options for individual values or data records via http also offer great potential for subsequent processing scenarios.
And if FOSHKplugin is already installed, you can also think about sending it to other weather services or programmes. See this document.
Perhaps you would like to visualise your weather data locally on an Android device? Personal Weather Tablet (PWT) would be a good choice. There is also a suitable recipe for this.
The possibilities are manifold …

Queries through a Prometheus server

Queries through a Prometheus server

Since v0.10, FOSHKplugin supports http queries by a Prometheus server.
This means that the data can be easily stored in a Prometheus time series database and analysed and graphically presented using Prometheus' own visualisations or Grafana.
FOSHKplugin provides either all, only the metric or only the imperial numerical values via the http interface.
This depends on the configured path in the prometheus.yml:

```
# provides all numerical data to a Prometheus server via http
- job_name: "FOSHKplugin"
  static_configs:
    - targets: [192.168.15.100:8096]

# only provides the metric values
- job_name: "FOSHKplugin-Metric"
  metrics_path: /metmetrics
  static_configs:
    - targets: [192.168.15.100:8096]

# use this to retrieve the imperial value only
- job_name: "FOSHKplugin-Imperial"
  metrics_path: /impmetrics
  static_configs:
    - targets: [192.168.15.100:8096]
```

The IP address specified under targets (here: 192.168.15.100) corresponds to the IP address of the host on which FOSHKplugin is running; the port number (8096 in this example) is the local http port (LBH_PORT) configured in FOSHKplugin (8080 by default).
More detailed explanations on Prometheus can be found in the online documentation on Prometheus.

All values are declared as type "gauge" - only keys with "time" in the name are declared as "counter".
Boolean keys such as the status states are transmitted in binary form with 0 (false) and 1 (true) as "gauge" type.
Strings are not provided as these are (apparently) not accepted in Prometheus.

If you have the choice between Prometheus and InfluxDB, you should rather orientate yourself towards InfluxDB. In the current implementation, data retrieval is initiated by Prometheus (pull) - if the connection between FOSHKplugin and Prometheus is interrupted, all data is lost for this period. With the InfluxDB implementation in FOSHKplugin, FOSHKplugin sends the data to InfluxDB (push) and stores all data temporarily if the connection to the remote InfluxDB server is lost. This data is automatically resent once the connection is restored. This means that there are no data gaps.

Processing signal quality data

Processing signal quality data

With Export\ADD_SIGNAL = True in the config file the additional collection of signal quality data of all

active sensors will be enabled.
This requires a console that supports http/JSON-API (currently GW1100, GW1200, GW2000, WN1980C, WS3xx0C).
The input string coming from the console defined via Weatherstation\WS_IP is supplemented by the corresponding keys with the value of the signal quality.
The following key assignment is available:

```
WS90: wh90sig
WS80: wh80sig
WS68: wh68sig
WH65/WS69: wh69sig
WH40: wh40sig
WH32B: wh25sig
WH32: wh26sig
WH57: wh57sig
WH45: wh45sig
WH41/WH43: wh41sigN (where N = 1..4)
WH55: wh55sigN (where N = 1..4)
WH31: wh31sigN (where N = 1..8)
WH51: wh51sigN (where N = 1..8)
WN34: wh34sigN (where N = 1..8)
WN35: wh35sigN (where N = 1..8)
```

The value of these keys is in the range from 0 to 4 and corresponds to the number of bars in the signal display - i.e. 0 = no reception and 4 = best reception.

These values are only available via UDP and getvalue as well as in forwards of type UDP, BANNER, TAGFILE, MQTT and InfluxDB. If these values are also to be sent with a forward in Ecowitt format, an additional

```
FWD_OPTION = blacklist=False
```

must be entered in the forward configuration in the config file for the corresponding forward.

Provide additional data from third-party devices globally for weather services (script)

Provide additional data from third-party devices globally for weather services (script)

With the FWD_EXEC function, it has long been possible to modify the output line for a forward shortly before it is actually sent to the respective weather station service using a script. This script only works on a forward-specific basis and must therefore be set up for each individual forward.
However, if, for example, you want to use the data from an Luftdaten.info fine dust sensor as data from a virtual WH45/WH46 for sending to several services, you would have to set up a separate script for each individual service.
And depending on the target format or FWD_TYPE (Ecowitt format, WU, Awekas, Ambient Weather etc.), this script must also have different content.

As of v0.10, ADD_SCRIPT can be used to define a global script that receives the data originally coming

from the weather station and makes it modifiable. After the change, FOSHKplugin receives the modified or extended data back and only then starts to process this data for each individual forward. This means that there is a centralised data manipulation option that is valid for all output formats.

Example:
The particulate matter sensor from Luftdaten.info can be queried locally via a web interface. The link http://ip-address/data.json (where ip-address corresponds to the IP address of the sensor) returns a JSON file with the values, which can be parsed and processed very easily using standard on-board tools.

JSON - depending on the software version of the device, however, the structure of the JSON may be different:


```
{
   "software_version":"NRZ-2020-133",
   "age":"28",
   "sensordatavalues":[
      {
         "value_type":"SDS_P1",
         "value":"8.25"
      },
      {
         "value_type":"SDS_P2",
         "value":"3.03"
      },
      {
         "value_type":"samples",
         "value":"5051440"
      },
      {
         "value_type":"min_micro",
         "value":"28"
      },
      {
         "value_type":"max_micro",
         "value":"20098"
      },
      {
         "value_type":"interval",
         "value":"145000"
      },
      {
         "value_type":"signal",
         "value":"-73"
      }
   ]
}
```


SDS_P1 corresponds here to the PM10 value and SDS_P2 to the PM2.5 value.
The WH45 recognises the keys tf_co2 (temperature in °F), humi_co2 (humidity), pm25_co2 (the PM2.5 value), pm25_24h_co2 (the 24h mean value of PM2.5), pm10_co2 (PM10 value), pm10_24h_co2 (24h

mean value PM10) in Ecowitt format, co2 (CO2) and co2_24h (24h average CO2). If we do not have a real WH45/WH46 ourselves, we can use the PM10 and PM2.5 values of the luftdaten.info sensor and emulate a WH45 - at least partially - by using the WH45/WH46 keys.
We therefore assign the value of SDS_P2 to the pm25_co2 key and the value of SDS_P1 to the pm10_co2 key using a script.

For a better understanding, here is the sequence of the individual steps:
1. FOSHKplugin receives the general weather data from the console
2. When the data is received, the configured script is started, which executes the following steps:
2.1 Query foreign data - the values must come from somewhere
2.2 merge with the real data from the weather station
2.3 Return to FOSHKplugin
3. FOSHKplugin sends the merged data to all configured weather services

Here is the example script that executes the complete point 2:

```
#!/bin/bash
# complements the output line of FOSHKplugin with values of the particulate
matter sensor from Luftdaten.info and emulates a WH41
# the channel number of the virtual WH41 is defined via the variable WH41
# depending on the output format, the output strings must be modified - in
Ecowitt format, the individual components are combined
# using an "&" - used here
# needs curl and jq - install with sudo apt-get install curl jq
# general help: https://foshkplugin.phantasoft.de/generic#exec

# define the local ip address of the Sensor
ADR=192.168.15.229

# define the channel number of the virtual WH41; 9 = emulate WH45 instead
WH41=9

#JSON=`cat luftdaten2.json`
JSON=`curl -s http://${ADR}/data.json`

# the original output data from FOSHKplugin
instr="$@"

# PM2.5
pm2=`echo ${JSON}|jq ".sensordatavalues[1].value"|sed 's/\"//g'`
if [ ! -z "${pm2}" ]; then
  if [ "${WH41}" = "9" ]; then pm2string="&pm25_co2=${pm2}"; else
pm2string="&pm25_ch${WH41}=${pm2}"; fi
fi

# PM10
pm1=`echo ${JSON}|jq ".sensordatavalues[0].value"|sed 's/\"//g'`
if [ ! -z "${pm1}" ]; then
  if [ "${WH41}" = "9" ]; then pm1string="&pm10_co2=${pm1}"; else
pm1string="&pm10_ch${WH41}=${pm1}"; fi
```

```
fi

# temp in °C - must be converted to °F - F = (C*9/5)+32 -
temp=`echo ${JSON}|jq ".sensordatavalues[2].value"|sed 's/\"//g'`
temp=`echo "scale=2; (${temp}*9/5)+32"|bc`
if [ ! -z "${temp}" ]; then
  if [ "${WH41}" = "9" ]; then tempstring="&tf_co2=${temp}"; else
tempstring="&tf_ch${WH41}=${temp}"; fi
fi

# pressure - not supported by WH41, WH45, WH46 - you could override the
original pressure from wether station
# not used
pressure=`echo ${JSON}|jq ".sensordatavalues[3].value"|sed 's/\"//g'`
if [ ! -z "${pressure}" ]; then
  if [ "${WH41}" = "9" ]; then pressurestring="&tf_pressure=${pressure}";
else pressurestring="&pressure_ch${WH41}=${pressure}"; fi
fi

#humidity
humidity=`echo ${JSON}|jq ".sensordatavalues[4].value"|sed 's/\"//g'`
if [ ! -z "${humidity}" ]; then
  if [ "${WH41}" = "9" ]; then humiditystring="&humi_co2=${humidity}"; else
humiditystring="&humidity_ch${WH41}=${humidity}"; fi
fi

# merge the original string with the new components
echo "${instr}${pm2string}${pm1string}${tempstring}${humiditystring}"
```

The script here is somewhat more extensive because it can provide data for either a WH41 or WH45.

For installation/configuration:
1. save the enclosed file as addLDdata.sh in the installation directory of FOSHKplugin (/opt/FOSHKplugin/)
2. customise addLDdata.sh - adjust the IP address of your particulate matter sensor at ADR
3. make the script executable: chmod ug+x addLDdata.sh
4. make sure that the required tools are available: sudo apt-get install curl jq
5. manual test of the script: ./addLDdata.sh - a line like &pm25_co2=8.15&pm10_co2=17.17&tf_co2=31.55&humi_co2=87.72 should then be displayed
6. customise the forward to PWSDashboard - the line FWD_EXEC = ./addLDdata.sh must be added in the relevant forward
7. restart FOSHKplugin - sudo service foshkplugin restart

Attention!
This is a very powerful feature that should be used with caution. Errors in the script can lead to trouble with the target weather service or even to the failure of FOSHKplugin.
As an immediate measure, always insert an exit line in the script below the

shebang "#!/bin/bash" in the event of any unclear errors - then the script will continue to be started by FOSHKplugin, but will no longer cause any mischief. Scripts should be well tested - even under conditions that are not intended - ideally locally. A text line can be specified as a parameter to which the additional keys and values are appended.
The SND log file logs the script call and any errors detected.

Further processing of data from FOSHKplugin in Home Assistant

Further processing of data from FOSHKplugin in Home Assistant

There are various ways of integrating data from an Ecowitt weather station into Home Assistant.

The easiest way to do this is via the Ecowitt integration contained in the core of HA. This provides most of the keys of the current Ecowitt weather stations.
However, the additional keys generated by FOSHKplugin are of course not processed and therefore do not end up in HA.

Another option - if an MQTT broker is available - is to transmit the data via MQTT.
However, this entails longer configuration work because you have to make an entry in a YAML for each desired data point the Home Assistant:

```
mqtt:
  sensor:
  - state_topic: "GW2000/tc_co2"
    name: "WH45 Temperature"
    unit_of_measurement: "°C"
    device_class: "temperature"
    icon: "mdi:thermometer"
```

This is very time-consuming (with 200 or even 450 topics).

With v0.10, FOSHKplugin now also supports the MQTT discovery of Home Assistant.
HA automatically finds FOSHKplugin as a device with all entities without the need for any configuration measures. Icon, unit and device class are already pre-assigned.
A forward block in the configuration file foshkplugin.conf should look something like this:

```
[Forward-60]
FWD_TYPE = MQTTMET
FWD_CMT = MQTT-Forward of metric values to LoxBerry
FWD_URL = 192.168.15.236:1883@homeassistant/FOSHKplugin
FWD_ENABLE = True
FWD_OPTION = MQTTCYCLE=5,hass=True,devname=GW1000-
Test,minmax=False,status=False
FWD_SID = MQTT-username
FWD_PWD = MQTT-password
```

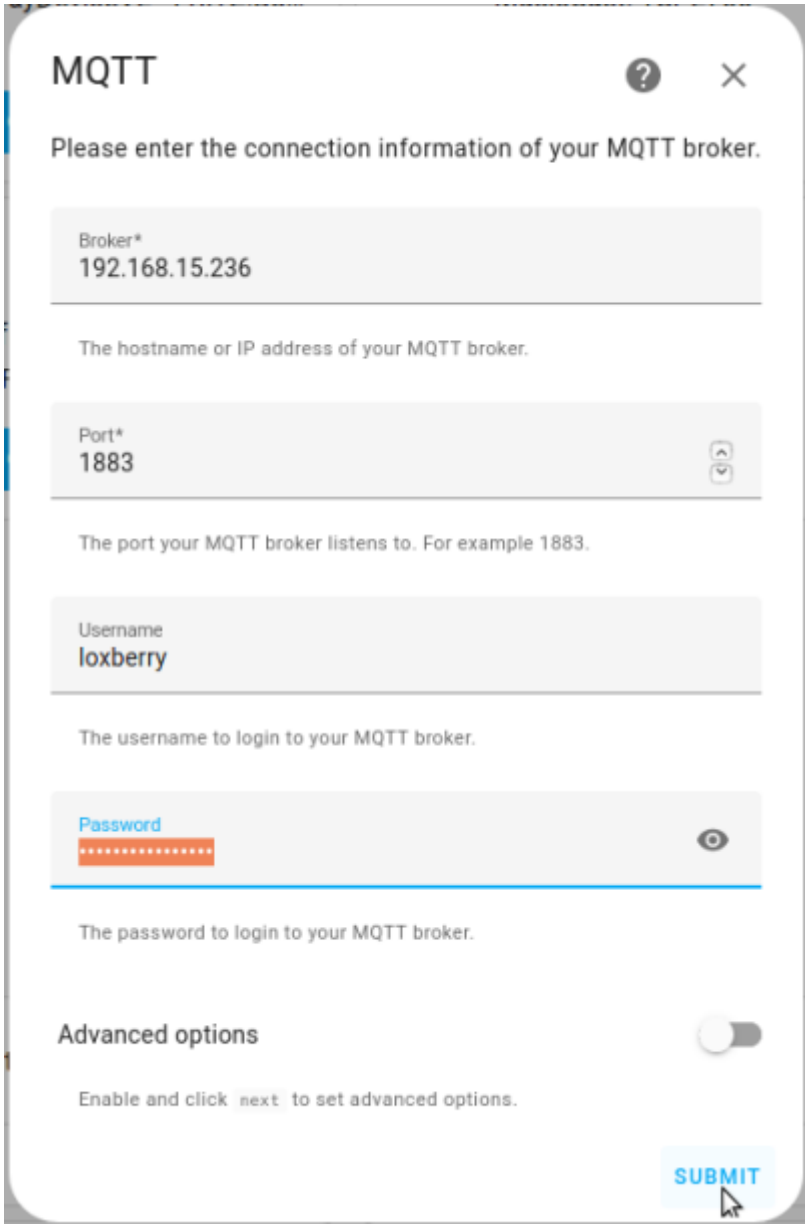
```
FWD_STATUS = True
```

MQTTMET should be selected as the forward type FWD_TYPE for metric values - MQTTIMP should be selected accordingly for imperial measurement systems.
The destination - i.e. the MQTT broker together with the desired hierarchy - must be specified in the FWD_URL. The structure of the address is MQTTipaddress:port@topic-hierarchy%prefix. A prefix "%prefix" is optional and therefore does not have to be specified.
In the FWD_OPTION line, at least hass=True must be entered to activate auto-discovery. The name of the device is defined with devame=name (default: FOSHKplugin).
By default, FOSHKplugin only sends the data to the broker when the value changes. In addition, an interval in minutes can be entered via MQTTCYCLE in which ALL topics - regardless of value changes - are sent to the broker.
With an interval of 0, ALL topics are sent at every send interval.
To minimise data transfer, I recommend entering an interval of 10 or 5 minutes for regular operation.
If the MQTT broker requires a login, the credentials can be configured with FWD_SID (user name) and FWD_PWD (password).

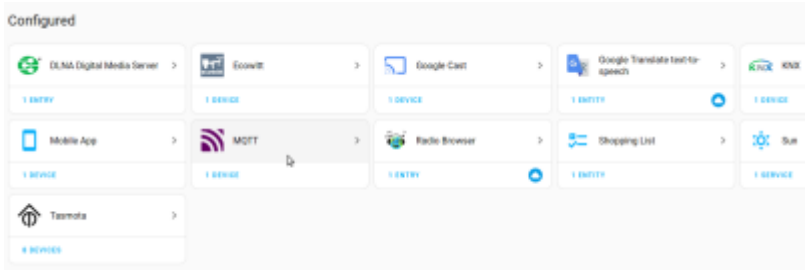
By default, all values including the min/max values (if configured) are transmitted via MQTT. To prevent the transmission of the min/max values, minmax=False can be set as an additional option in FWD_OPTION. If the status messages from FOSHKplugin are also to be made available via MQTT, FWD_STATUS should be set to True. Alternatively, this option can also be specified as status=True in FWD_OPTION.

Attention!
The spelling of the options is case-sensitive - only "MQTTCYCLE", "hass", "devname" "minmax" and "status" apply.
The hierarchy and the device name must not contain any special characters, umlauts or spaces (these are automatically removed).

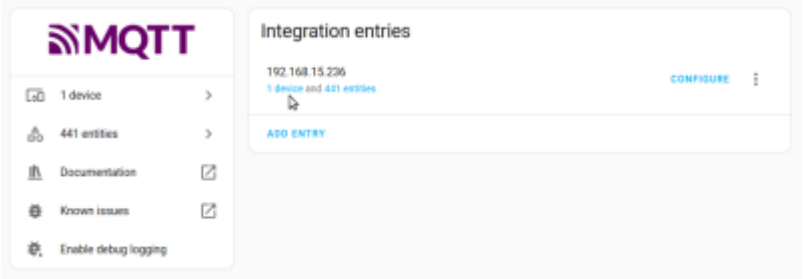
In Home Assistant, select and configure the MQTT integration (address, port and credentials if necessary) under Settings/Devices&Services using the "ADD INTEGRATION" button at the bottom right:



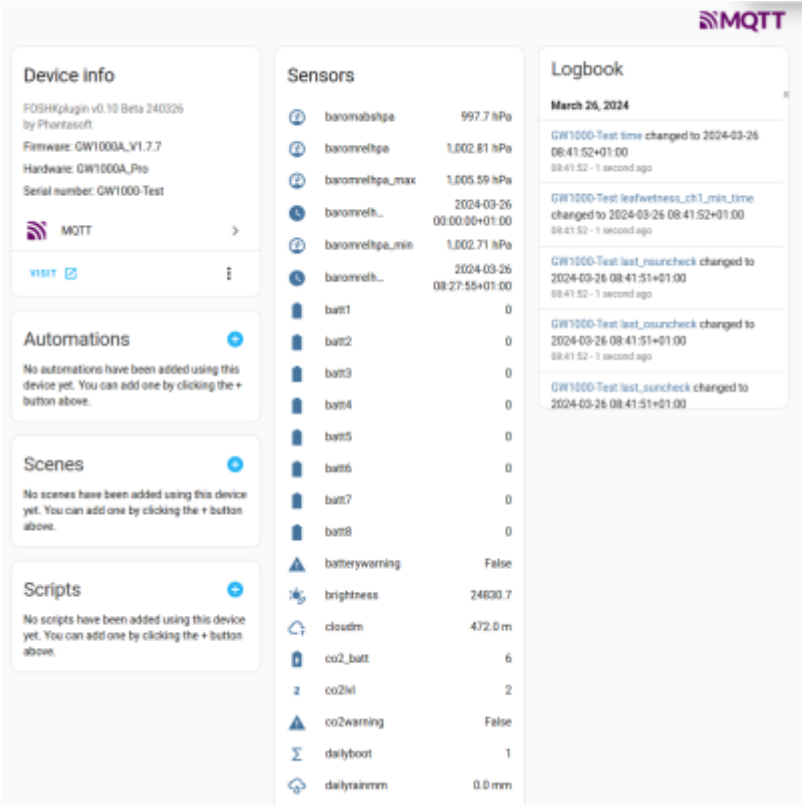
You will then find the MQTT button under the configured integrations, which already shows 1 recognised device:



After clicking on the MQTT button, information about this integration appears, such as the number of devices and the number of entities in this integration:



In the device info you can finally see all information about the FOSHKplugin device and change it if necessary:



All data points generated by FOSHKplugin should now be available within Home Assistant for further use and visualisation.

Advantages of the connection via FOSHKplugin and MQTT

- Signal quality values of the sensors are transmitted (if the console supports the http/JSON API)
- Additional values available such as wind chill, dew point (also for all indoor sensors), spread, feelslike, heat index, AQI for WH41/43/45/46, 10min average for wind speed and direction, max. gust of the last 10 minutes, brightness (lux), cloud height, sunshine duration, air pressure trend and change 1h/3h, if EVAL_VALUES is active
- Any values from other sensors can be integrated
- Availability of min/max values with respective times
- Warnings (thunderstorm, storm, leakage, CO2, station or sensor failure, battery warning, interval warning, update) are available directly in Home Assistant

Note:
The MQTT Explorer, which is available for various platforms, is an extremely helpful tool for analysing any problems or for a general basic understanding of MQTT.

2bc ...