Technik

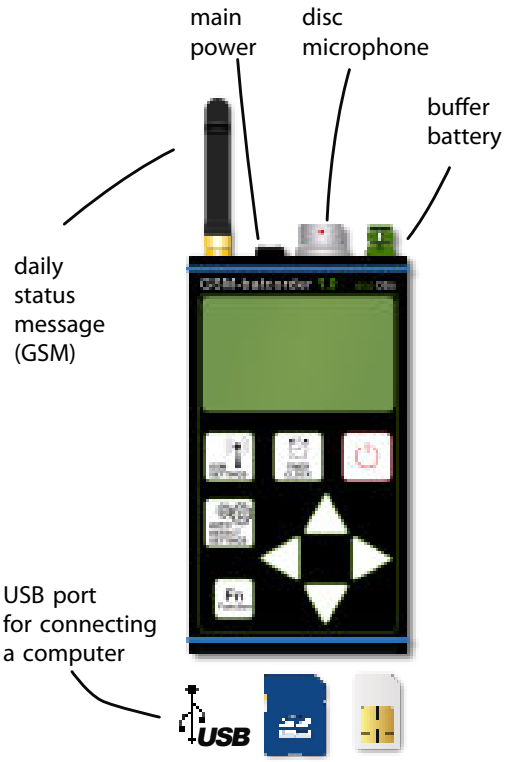
Übersicht GSM-batcorder und RaspberryPi

Im Folgenden eine kurze Beschreibung der RaspberryPi-Lösung. Dazu beschreiben wir zuerst den GSM-batcorder und die typische Anwendung, damit auch Funktionen und das Verhalten während des Betriebs bekannt ist. Stand 25.1.2017

GSM-batcorder

Basis unserer „neuen“ Technik

Seit 2017 bieten wir den GSM-batcorder an. Er wurde speziell für den Einsatz im Gondelmonitoring entwickelt. Der GSM-batcorder ist ein normaler batcorder, der jedoch die Technik des Steuermoduls aus der WKA-Erweiterung beinhaltet. Außerdem erlaubt den Anschluss eines Computers über USB zum Auslesen der Daten. Dies ist immer dann möglich, wenn der GSM-batcorder morgens die Fledermaus-Suche einstellt und die SD-Karte über USB freigibt.



Kurze Beschreibung der Funktionen

- Status-SMS: Jeden Morgen wird nach Beenden der Fledermaus-Suche ein kurzer System-Status per SMS gesendet. Dieser beinhaltet Anzahl Aufnahmen, Speicherplatz der SD-Karte, Mikrofonempfindlichkeit und u.a. den Zustand der Stromversorgung (230V ja/nein)
- Disk-Mikrofon: Mikrofonscheibe mit 50 cm Kabel
- Puffer-Batterie: der GSM-batcorder läuft bei der Fledermaus-Suche über Batterie und koppelt das Stromnetz ab. Tagsüber wird diese Batterie dann wieder geladen. Bei Stromausfällen der Anlage kann das System so mehrere Tage autonom arbeiten
- Anschluss ans 230V Netz mittels Netzteil (EN61000-4 ESDair 15KV)
- Jeden Abend und Morgen wird ein Testsignal fürs Mikrofon vorgespielt und als Aufnahme gespeichert

Datenspeicherung
Der batcorder speichert Aufnahmen je erkannter Fledermaus auf der eingelegten Speicherkarte. Entsprechend der Einstellung Threshold erreichen diese Aufnahmen zwischen minimal 200ms und meist nicht über 5 Sekunden Aufnahmedauer. Je Sekunde wird ein MB Speicherplatz benötigt. Vereinfacht kann man daher je Aufnahme mit einem MB rechnen. Außerdem protokolliert der batcorder in der Datei LOGFILE.TXT alle An/Abschaltvorgänge, SMS-Versand, Fehler etc. mit.

Typische Datenmengen
Wir führen selber keine Gondelmonitorings durch, daher fehlt uns ein direkter Erfahrungswert. Jedoch haben wir in Rücksprache mit Kunden einen Überblick gewonnen. Es gibt zum einen große Unterschiede bedingt durch die Gondeltypen, aber auch Jahreszeitlich ändert sich die Menge an Aufnahmen regelmässig. So ist zum Beispiel ab Juli bis September durch Fledermäuse eine höhere Anzahl Aufnahmen zu erzielen. Gemittelt auf die Erfassungstage lassen sich grob folgende Unterscheidungen machen:

- regelmässig werden **< 10 Aufnahmen** im Mittel je Nacht gespeichert, nur wenige Nächte mit bis zu 100 Aufnahmen
- häufig werden im Mittel **10 bis 250 Aufnahmen** gespeichert, einzelne Nächte mit bis zu 1000 Aufnahmen (Störgeräusche oder Fledermäuse) sind möglich
- selten werden im Mittel **500 Aufnahmen** gespeichert, einzelne Nächte mit bis zu 5000 Aufnahmen (Störgeräusche) sind möglich
- selten werden je Nacht **>1000 Aufnahmen** gespeichert, Ursache sind dann immer massive Störgeräusche

Seit 2016 verfügt der batcorder über einen digitalen Störfilter, d.h. die Häufigkeit der seltenen Ereignisse könnte noch geringer werden, hier fehlt jedoch fundiertes Feedback. Bisher zeichnet es sich jedoch ab, dass der Filter sehr gut funktioniert. Einzig minusartige Geräusche werden dann nach wie vor nicht gefiltert.

RaspberryPi - Anbindung

Der RaspberryPi kann mehrere Aufgaben in Kombination mit dem GSM-batcorder erfüllen. Im einfachsten Fall bietet er die Möglichkeit der automatischen Datensicherung. Die Ideallösung stellt die Datenfernübertragung über dar. Aktuell gibt es eine einfache Skriptlösung für die Datensicherung, die im Folgenden vorgestellt wird.

Datensicherung

Die bisherige Implementierung nutzt die automatische Aktivierung des USB-Anschluss des GSM-batcorder mit Beendigung der Fledermaus-Suche. Die Aktivierung des USB-Ports bewirkt auf Seite des angeschlossenen RaspberryPi das automatische Aktivieren der SD-Karte des batcorder. Dieses Ereignis wird durch USBMOUNT[1] erkannt. Daraufhin wird ein bereitgestelltes Skript ausgeführt (im Anhang).

Das Skript erwartet einen Link zum GSM-batcorder und dem Backup-Medium im Verzeichnis /var/run/usbmount/ mit den festen Namen **GSM-BC** und **BACKUP**. Dazu stellt das Skript sicher, dass die Laufwerke als Link mit dem Labelnamen innerhalb von /var/run/usbmount/ zur Verfügung gestellt werden. Sind beide Links vorhanden, und damit batcorder und Backup verfügbar, läuft das Skript weiter. Der batcorder formatiert die Karte immer mit diesem Namen, das Speichermedium muss der Anwender vorher als BACKUP benennen. Beachten Sie, dass nur FAT32-formatierte Medien sauber und zuverlässig vom RaspberryPi genutzt werden können. Daher muss das Backup-Medium immer als FAT32 formatiert sein. Wenn Sie abweichende formatierte Medien verwenden, müssen Sie gegebenenfalls die RaspberryPi Installation um benötigte Pakete erweitern.

Das Skript holt sich den *device node*, da es diesen benötigt, um die SD-Karte fürs Löschen umzubenennen (siehe später im Skript)

Als nächstes wird die LOGFILE.TXT in einem täglich neuen ZIP im Backup gesichert. Damit entsteht je Backupvorgang eine komprimierte Kopie der aktuellsten Log-Datei.

Dann wird der Speicher auf dem Backup geprüft, wenn dieser über 90% gefüllt ist, wird das Skript beendet.

Im nächsten Schritt wird der Speicher des batcorders geprüft. Ist dieser zu mehr als einem wählbaren Wert (%; Standard 90%) gefüllt, dann wird das Label ein DELETEME geändert. Damit erkennt der GSM-batcorder beim nächsten Start, dass er Löschen darf. Wird die Schwelle zB auf > 100 gesetzt, wird niemals gelöscht.

Im Anschluss werden mittels rsync mit Quelle GSM-batcorder und Ziel-Ordner Backup/GSM_Backups alle neuen Dateien gesichert.

[1] https://usbmount.alioth.debian.org

Noch nicht implementiert ist das Speichern des Backups mittels rsync direkt in eine ZIP-Datei, um Platz zu sparen auf dem backup und etwaige Transfers übers Netz zu beschleunigen.

Mögliche Setups GSM-batcorder / RaspberryPi

Bei der Verwendung des GSM-batcorders mit einem RaspberryPi gibt es diverse Möglichkeiten des Einsatzes. Diese werden im Folgenden kurz skizziert:

- **GSM⟷RPi** : Nur Backup-Dienst des RaspberryPi, beide Geräte am selben Ort (Gondel) installiert
- **GSM⟷USB/LWL/USB⟷RPi**: batcorder in Gondel, RaspberryPi im Mastfuß, über USB auf Lichtwellen-Leiter mit dem GSM verbunden; nur Backup durch den Raspberry und Zugriff durch Kunde auf RaspberryPi im Mastfuß
- **GSM⟷RPi⟷Netzwerk_WEA**: Beide Geräte in Gondel, RaspberryPi ins Netzwerk der WEA integriert; Kunde kann Daten vom Mastfuß übers Netzwerk vom RaspberryPi abrufen
- **GSM⟷RPi⟷Netzwerk_WEA⟷DSL** : wie vorher, Daten über DSL vom RaspberryPi abrufbar respektive RaspberryPi überträgt Daten automatisch auf Server des Kunden
- **GSM⟷USB/LWL/USB⟷RPi⟷DSL**: batcorder in Gondel, RaspberryPi im Mastfuß, über USB auf Lichtwellen-Leiter mit dem GSM verbunden; RaspberryPi über DSL-Anschluss der Anlage von Außen abrufbar oder Upload der Daten auf Server des Kunden
- **GSM⟷RPi⟷LTE-Modem**: wie vorher, Daten über DSL vom RaspberryPi abrufbar respektive RaspberryPi überträgt Daten automatisch auf Server des Kunden

Ist der RaspberryPi mit dem Internet verbunden (LTE oder DSL), dann besteht die Möglichkeit, aktiv auf die Daten zuzugreifen. Hierzu muss der Router der WEA im Falle von DSL so konfiguriert sein, dass Anfragen über SSH auf den RaspberryPi weitergeleitet werden. Dies muss durch den Betreiber oder ein Service-Team erfolgen. Sollte die WEA nicht über eine feste IP-Adresse verfügen, dann muss zusätzlich ein Dienst aktiviert werden, der dynamisch die wechselnde IP-Adresse des Routers mit einem DNS-Eintrag (Server-Name) verknüpft. Bei LTE ist nicht in jedem Vertrag eine öffentliche IP vergeben, so dass unter Umständen ein aktiver Zugriff von ausserhalb nur mit größerem Aufwand möglich ist.

Alternativ kann bei fehlendem Zugriff von Außen, aber dennoch bestehender Internetverbindung, auch der RaspberryPi die Daten analog zum lokalen Backup mittels rsync auf einen im Internet verfügbaren Server übertragen.

Anhang

Skript für USBMOUNT

```
#!/bin/bash
#######################################
#################################
# the following is probably double,
already inside the standard
# usbmount mount script
########################
# nevertheless we rely on some mount
points and leave it as it is
# version 1.0 - 1. August 2016 um
14:56:21 MESZ
# (c) volker runkel, ecoObs GmbH
#######################################
#################################
# Copyright for parts of this script
# This script creates the volume label
symlink in /var/run/usbmount.
# Copyright (C) 2014 Oliver Sauder
#
# This file is free software; the co-
pyright holder gives unlimited
# permission to copy and/or distribute
it, with or without
# modifications, as long as this noti-
ce is preserved.
#
# This file is distributed in the hope
that it will be useful,
# but WITHOUT ANY WARRANTY, to the
extent permitted by law; without
# even the implied warranty of MER-
CHANTABILITY or FITNESS FOR A
# PARTICULAR PURPOSE.
#

#########
# This script runs on mounting volumes
via usbmount
# we do have a critical problem - run-
time is limited
# so after ca. 5 seconds it gets kil-
led
# currently this is circumvented by
calling the long lasting rsync
# using the at command ... yet, we
shouldn't add much more and move to a
better
# longer lasting runtime system sooner
or later
# especially all the file logging
might slow down
#########

set -e

# **** Delete Threshold / Schwelle zum
Löschen der SD-Karte ****
# set the threshold to any value bet-
ween 0 and 99
# to initiate relabeling of SD card
for deletion as soon as card is filled
# up to that number in percentage
# if automatic deletion is not wanted,
set the value to above 100
# * * *
# Schwelle auf Wert zwischen 0 und 99
# die SD-Karte wird dann zum Löschen
markiert, wenn die Füllung
# in Prozent diesen Wert erreicht hat
# bei Werten über 100 wird niemals ge-
löscht
#
**************************************
************************
DELETE_THRESHOLD=90

# **** Mail to / Mail an ****
# the mail adress reports will get
sent to if mail is configured
# and internet available
# * * *
# Empfänger-Adresse für Status Mails
# wenn Mail konfiguriert und Internet
verfügbar ist
#
**************************************
************************
MAIL_TO="runkel@ecoobs.de"

# **** Raspi ID for Mail / Raspi ID
für Mails ****
# a string that is appended to mail
messages
# to identify this raspi
# * * *
```

```
# Text für Mails um diesen Raspi zu
identifizieren
#
************************************
************************
MAIL_ID="WKA1-Raspi"

date >> /home/pi/GSM-Logging.txt


# Exit if device or mountpoint is em-
pty.
test -z "$UM_DEVICE" && test -z "$UM_-
MOUNTPOINT" && exit 0

# get volume label name of the newly
mounted volume
label=`blkid -s LABEL -o value $UM_DE-
VICE`

# If the symlink does not yet exist,
create it.
# that should already happen with the
default 00_ script, but does no harm
test -z $label || test -e "/var/run/
usbmount/$label" || ln -sf "$UM_MOUNT-
POINT" "/var/run/usbmount/$label"

########################################
##########
### work horse operation starts here
####
########################
## most commands leave a trail in /
home/pi/Documents/GSM-Logging.txt
# so you can later on check the script
runtime behaviour
#######

# check mountpoint for backups : needs
to be BACKUP
if ! mountpoint -q /var/run/usbmount/
BACKUP/
  then
    echo "backup filesystem not moun-
ted" >> /home/pi/GSM-Logging.txt
    exit 0
      else
      echo "backup filesystem mounted"
>> /home/pi/GSM-Logging.txt
fi

########################
# check mountpoint for batcorder : vo-
lume name/label needs to be GSM_BC
# set automatically by the gsm-batcor-
der when formatting the card
if ! mountpoint -q /var/run/usbmount/
GSM_BC/
  then
    echo "batcorder filesystem not
mounted"  >> /home/pi/GSM-Logging.txt
    exit 0
      else
      echo "batcorder filesystem moun-
ted" >> /home/pi/GSM-Logging.txt

fi

# now we need to get the device node
for possible relabeling later on
GSM_DEVICE=""
if [[ $label = "GSM_BC" ]]; then
      GSM_DEVICE=$UM_DEVICE
      echo "batcorder filesystem on
$GSM_DEVICE"  >> /home/pi/GSM-Loggin-
g.txt
fi

########################
# create filename for logfile copy ba-
sed on current timestamp
# and create a zip with logfile
# NEEDS TESTING -> what happens if no
LOGFILE.TXT exists!?
# what happens if no correct clock is
running ?
# running number just as a revision
might be better
echo "Backing up logfile" >> /home/pi/
GSM-Logging.txt
current_timestamp=$(date +%Y%m%d-%H%M
%S)
echo "zip /var/run/usbmount/BACKUP/
GSM_Backups/LOGFILE-$current_timestamp
/var/run/usbmount/BACKUP/GSM_Backups/
LOGFILE.TXT" | at now

########################
# Test if Backup volume is at least
90% full
# if it is, no further backups!
#
```

```
backupUsedSpace=$(df -k /var/run/us-
bmount/BACKUP/ | tail -1 | awk '{sub
(/%/, "", $5); print $5;}')
if [[ "$backupUsedSpace" -gt 90 ]];
then
      exit 0;
fi

########################
# Test if Batcorder volume is at least
90% full
# if it is, the mlabel command initia-
tes delete of card
# else we do a normal rsync
#

myUsedSpace=$(df -k /var/run/usbmount/
GSM_BC/ | tail -1 | awk '{sub (/%/,
"", $5); print $5;}')
if [[ "$myUsedSpace" -gt "$DELETE_TH-
RESHOLD" ]]; then
      if [[ -n "$GSM_DEVICE" ]]; then
            echo "Rsync and DELETEME
relabeling, used on BACKUP $backupU-
sedSpace" >> /home/pi/GSM-Logging.txt
          echo "rsync -a /var/run/us-
bmount/GSM_BC/* /var/run/usbmount/
BACKUP/GSM_Backups && umount $GSM_DE-
VICE && mlabel -i $GSM_DEVICE ::DELE-
TEME" | at now
            echo "Normal rsync star-
ted, sd card delete initiated, used on
BACKUP $backupUsedSpace" | mail -s
"$MAIL_ID: Raspi update" $MAIL_TO

      else
            echo "Normal Rsync, GSM
device not available, no DELETEME re-
labeling! Used on BACKUP $backupUsedS-
pace" > /home/pi/GSM-Logging.txt
            echo "rsync -a /var/run/
usbmount/GSM_BC/* /var/run/usbmount/
BACKUP/GSM_Backups" | at now
      echo "Normal Rsync, GSM device
not available, no DELETEME relabeling!
Used on BACKUP $backupUsedSpace" |
mail -s "$MAIL_ID: Raspi update"
$MAIL_TO
      fi
else
      echo "Normal Rsync, used on
BACKUP $backupUsedSpace" >> /home/pi/
GSM-Logging.txt
      echo "rsync -a /var/run/us-
bmount/GSM_BC/* /var/run/usbmount/
BACKUP/GSM_Backups" | at now
      echo "Normal rsync started, used
on BACKUP $backupUsedSpace" | mail -s
"$MAIL_ID: Raspi update" $MAIL_TO
fi

######## MISSING #####
####
# we could also unmount Batcorder af-
ter rsync
####
# error management: rsync fails ?!,
currently an email is sent to root on
raspi
# once sd card BACKUP showed i/o er-
ror, and was then mounted ro
####
# other errors, can they be detected
and acted upon?
# sending mails if email is available
# writing a logfile in pi
####
# A script that tests if names/labels
are correct
# user inserts new backup card, and
then ssh's and starts a script
######## MISSING #####

exit 0
```