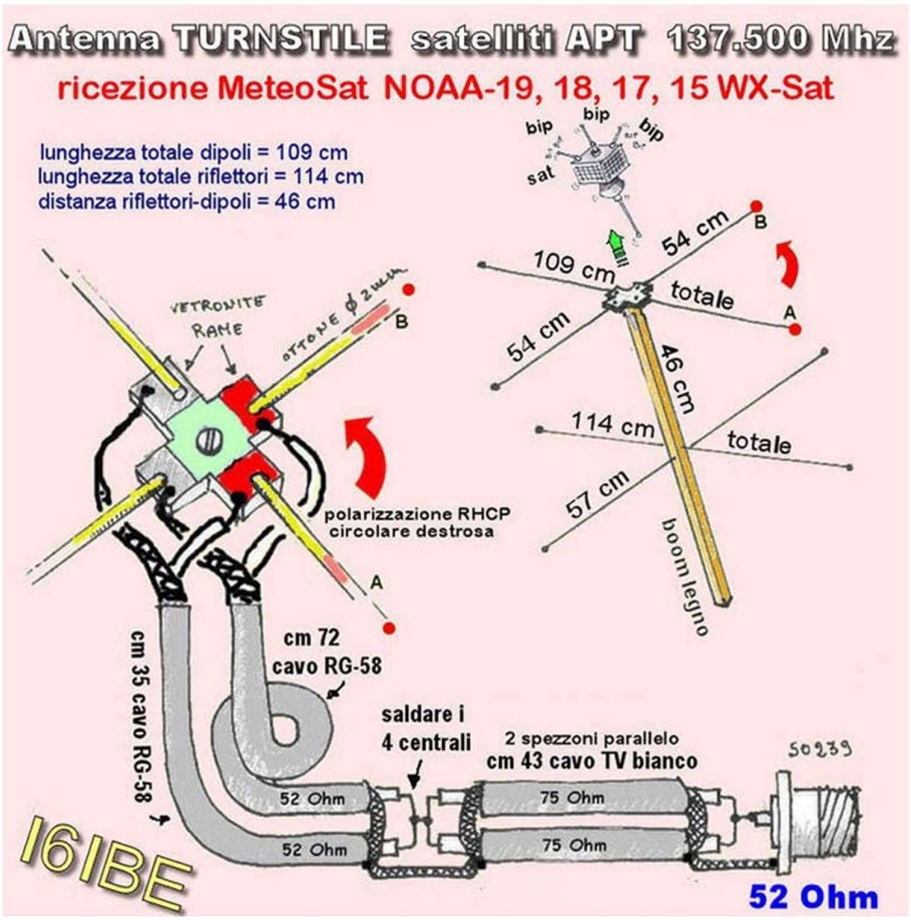
Automated NOAA Satellite Image recording system IQ2XN C.I.S.A.R. Orobie
Administrator: IU2PLZ - Alberto DRAGO – "Linux System Administrator"
Telegram Channel: https://t.me/iq2xnnoaa

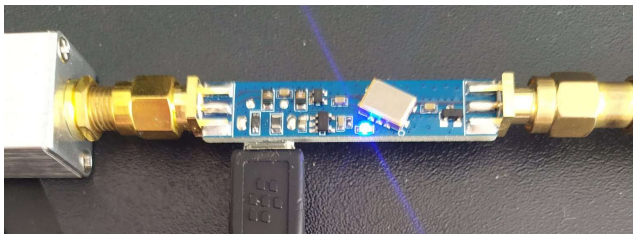
Di seguito la realizzazione di un Sistema Automatizzato finalizzato ad ottenere la decodifica dei satelliti NOAA
Sfrutteremo quindi un Raspberry Pi 3B+ con una chiavetta SDR e una Antenna munita di LNA

Materiale Occorrente:

- Raspberry Pi (consigliata versione 3 o superiore) Alimentato da almeno 2A continui
- Micro USB da 32Gb (capiente per l'archivizione delle immagini ricevute)
- Chiavetta SDR (è stata utilizzata una noeelec professionale)
- Antenna a 137MHz per satelliti (è stata realizzata una "Turnstile" prendendo spunto dall'immagine qui sotto)



- a monte della chiavetta SDR è stato posto un preamplificatore 137Mhz SAW BPF



Per il SO da mettere sul RB scegliamo la versione a 32 bit perché potremmo avere incompatibilità con il SW necessario e le librerie annesse.

Questi i riferimenti Kernel e Versione OS estrapolati dal sistema da me realizzato:

```
pi@weather-satellite:~ $ uname -a
Linux weather-satellite 5.10.103-v7+ #1530 SMP Tue Mar 8 13:02:44 GMT 2022 armv7l GNU/Linux

pi@weather-satellite:~ $ cat /etc/debian_version
11.3
```

Per installare il SO (img scaricata) utilizziamo i soliti SW di preparazione SD (io consiglio “balenaEtcher”)
Colleghiamo il RB ad internet tramite cavo fisico sull’interfaccia di rete (`eth0`)
Installato il SO colleghiamoci via SSH da un client (utente pi – defaul password raspberry)
** ricordatevi di effettuare da subito il cambio password digitando `passwd pi`

Procediamo ad interrogarare i repositori ed eseguire l’upgrade del SO

```
sudo apt-get update && apt-get upgrade
```

Eseguiamo un riavvio completo del sistema

```
sudo reboot
```

Ricolleghiamoci ed installiamo `git` (ci servirà per recuperare tutti i files necessari)
Installiamo anche e `vim`

```
sudo apt-get install -y git vim
```

sono necessarie librerie USBLib e i driver per la RTL2832 e un tool di compilazione che ci verrà utile più avanti

```
sudo apt-get install -y libusb-1.0
sudo apt-get install -y cmake
sudo apt-get install -y libxft2
```

creiamo il file per evitare incompatibilità con il kernel installato modifichiamo un file di configurazione con:

```
sudo vim /etc/modprobe.d/no-rtl.conf
```

ed inseriamo su questo le seguenti righe

```
blacklist dvb_usb_rtl28xxu
blacklist rtl2832
blacklist rtl2830
```

installazione della suite SDR

```
cd ~
git clone https://github.com/keenerd/rtl-sdr.git
cd rtl-sdr/
mkdir build
cd build
cmake ../ -DINSTALL_UDEV_RULES=ON
make
sudo make install
sudo ldconfig
cd ~
```

```
sudo cp ./rtl-sdr/rtl-sdr.rules /etc/udev/rules.d/
```

Nel caso in cui la compilazione dia qualche errore è necessario rilanciare il comando “apt-get update” e, se necessario ri-scaricare le librerie mancanti di “libusb” e “cmake”

eseguimo reboot
Per poter registrare l’audio in uscita e far partire la decodifica ad un dato orario usiamo:

```
sudo apt-get install -y sox
sudo apt-get install -y at
sudo apt-get install -y predict
```

Ora dobbiamo installare il sofware di decodifica (WxtoIMG)

```
cd ~
wget http://www.wxtoimg.com/beta/wxtoimg-armhf-2.11.2-beta.deb
```

procediamo con l'installazione:

```
sudo dpkg -i wxtoimg-armhf-2.11.2-beta.deb
```

Per verificare che la chiavetta sia riconosciuta spegniamo il Raspberry, colleghiamo la chiavetta e riaccendiamo, poi diamo il comando:

```
sudo rtl_test
```

dovreste avere un output sime a questo:

```
Found 1 device(s):
  0:  Realtek, RTL2838UHIDIR, SN: 00000001

Using device 0: Generic RTL2832U OEM
Found Rafael Micro R820T tuner
Supported gain values (29): 0.0 0.9 ...
Sampling at 2048000 S/s.

Info: This tool will continuously read from the device, and report if
samples get lost. If you observe no further output, everything is fine.

Reading samples in async mode...
```

Ora prepariamo il software per prevedere i passaggi:

```
predict
```

Ci apparirà questa schermata:

```
                                --== PREDICT  v2.2.3 ==--
                            Released by John A. Magliacane, KD2BD
                                         May 2006


                                      --==[ Main Menu ]==--

            [P]: Predict Satellite Passes          [I]: Program Information
            [V]: Predict Visible Passes            [G]: Edit Ground Station Information
            [S]: Solar Illumination Predictions    [D]: Display Satellite Orbital Data
            [L]: Lunar Predictions                 [U]: Update Sat Elements From File
            [O]: Solar Predictions                 [E]: Manually Edit Orbital Elements
            [T]: Single Satellite Tracking Mode    [B]: Edit Transponder Database
            [M]: Multi-Satellite Tracking Mode     [Q]: Exit PREDICT
```

Premiamo “G” (se la schermata delle coordinate non appare automaticamente):

```
                *  Ground Station Location Editing Utility  *
```

```
Station Callsign  : IQ2XN
Station Latitude  : X0.0000 [DegN]
Station Longitude : -010.0000 [DegW]
Station Altitude  : 25 [m]
```

Enter the callsign or identifier of your ground station.
Inseriamo i nostri dati, attenzione che la longitudine è espressa in gradi OVEST, per noi quindi dovremmo anteporre il segno "-".

Ora avviamo WxtoIMG per accettare i termini di utilizzo digitando il seguente comando:

```
wxtoimg
```

Chiudiamolo e creiamo il file di configurazione

```
vim ~/.wxtoimgrc
```

Anche qui dovremo inserire le nostre coordinate (a differenza del software precedente la longitudine è in gradi EST, non è quindi necessario il segno "-")

```
Latitude: 40.0000
Longitude: 10.0000
Altitude: 25
```

OK! tutti i programmi sono pronti, ora mancano gli script di decodifica veri e propri, creiamo il primo file con:

per ottenere le immagini con il dettaglio delle precipitazioni dobbiamo necessariamente registrare wxtoimage

lanciare wxtoimage e una volta aperto il software inserire i seguenti dettagli di registrazione per abilitarlo in modalita professional:

```
------------------------------
Kevin Schuchmann
CGHZ-PP9G-EAJZ-AWKK-NDNX
------------------------------
```

adesso eseguimi i seguenti commandi:

```
cd ~
mkdir weather
cd weather
mkdir predict
cd predict
sudo vim schedule_all.sh
```

e al suo interno inseriamo:

```
#!/bin/bash
# Update Satellite Information

wget -qr https://www.celestrak.com/NORAD/elements/weather.txt -O /home/pi/weather/predict/weather.txt
grep "NOAA 15" /home/pi/weather/predict/weather.txt -A 2 > /home/pi/weather/predict/weather.tle
grep "NOAA 18" /home/pi/weather/predict/weather.txt -A 2 >> /home/pi/weather/predict/weather.tle
grep "NOAA 19" /home/pi/weather/predict/weather.txt -A 2 >> /home/pi/weather/predict/weather.tle
#grep "METEOR-M 2" /home/pi/weather/predict/weather.txt -A 2 >> /home/pi/weather/predict/weather.tle

#Remove all AT jobs

for i in `atq | awk '{print $1}'`;do atrm $i;done

#Schedule Satellite Passes:

/home/pi/weather/predict/schedule_satellite.sh "NOAA 19" 137.1000
/home/pi/weather/predict/schedule_satellite.sh "NOAA 18" 137.9125
/home/pi/weather/predict/schedule_satellite.sh "NOAA 15" 137.6200
premiamo CTRL+O per salvare e CTRL+X per chiudere e passiamo al secondo file:
```

```
sudo vim schedule_satellite.sh
in cui incolleremo:

#!/bin/bash
PREDICTION_START=`/usr/bin/predict -t /home/pi/weather/predict/weather.tle -p "${1}" | head -1`
PREDICTION_END=`/usr/bin/predict -t /home/pi/weather/predict/weather.tle -p "${1}" | tail -1`

var2=`echo $PREDICTION_END | cut -d " " -f 1`

MAXELEV=`/usr/bin/predict -t /home/pi/weather/predict/weather.tle -p "${1}" | awk -v max=0
'{if($5>max){max=$5}}END{print max}'`

while [ `date --date="TZ=\"UTC\" @${var2}" +%D` == `date +%D` ]; do

START_TIME=`echo $PREDICTION_START | cut -d " " -f 3-4`

var1=`echo $PREDICTION_START | cut -d " " -f 1`

var3=`echo $START_TIME | cut -d " " -f 2 | cut -d ":" -f 3`

TIMER=`expr $var2 - $var1 + $var3`

OUTDATE=`date --date="TZ=\"UTC\" $START_TIME" +%Y%m%d-%H%M%S`

if [ $MAXELEV -gt 19 ]
  then
    echo ${1//" "}${OUTDATE} $MAXELEV

    echo "/home/pi/weather/predict/receive_and_process_satellite.sh \"${1}\" $2 /home/pi/weather/${1//" "}${OUTDATE}
/home/pi/weather/predict/weather.tle $var1 $TIMER" | at `date --date="TZ=\"UTC\" $START_TIME" +"%H:%M %D"`

fi

nextpredict=`expr $var2 + 60`

PREDICTION_START=`/usr/bin/predict -t /home/pi/weather/predict/weather.tle -p "${1}" $nextpredict | head -1`
PREDICTION_END=`/usr/bin/predict -t /home/pi/weather/predict/weather.tle -p "${1}"  $nextpredict | tail -1`

MAXELEV=`/usr/bin/predict -t /home/pi/weather/predict/weather.tle -p "${1}" $nextpredict | awk -v max=0
'{if($5>max){max=$5}}END{print max}'`

var2=`echo $PREDICTION_END | cut -d " " -f 1`

done
```

salvare e uscire

Ci siamo quasi, creiamo questo ultimo file:

```
sudo vim receive_and_process_satellite.sh
```

le istruzioni di seguito serviranno per ottenere la ricezione e decodifica dei satelliti NOAA
Incolliamo il seguente contenuto:

```
#!/bin/bash

# $1 = Satellite Name
# $2 = Frequency
# $3 = FileName base
# $4 = TLE File
# $5 = EPOC start time
# $6 = Time to capture

sudo timeout $6 rtl_fm -f ${2}M -s 60k -g 28.0 -p 0 -E wav -E deemp -F 9 - | sox -t wav - $3.wav rate 11025

PassStart=`expr $5 + 90`

if [ -e $3.wav ]
  then
    /usr/local/bin/wxmap -T "${1}" -H $4 -p 0 -l 0 -o $PassStart ${3}-map.png
    sudo /usr/local/bin/wxtoimg -m ${3}-map.png -e ZA $3.wav $3.png
    sudo /usr/local/bin/wxtoimg -m ${3}-map.png -e MCIR-precip -c $3.wav ${3}.MCIR-precip.png
    sudo /usr/local/bin/wxtoimg -m ${3}-map.png -e MSA-precip -c $3.wav ${3}.MSA-precip.png
    sudo /usr/local/bin/wxtoimg -m ${3}-map.png -e HVC -c $3.wav ${3}.HVC.png
fi
```

In questo file facciamo attenzione ad un parametro, ovvero il numero dopo “-g” che dovrà variare da chiavetta a chiavetta a seconda dei parametri supportati del “Gain” che abbiamo letto prima, cerchiamo di trovare il valore che corrisponda al 70-75% del massimo supportato, ci garantirà la migliore qualità. Probabilmente

dovremo cambiarlo in futuro a seconda della qualità immagini ricevute. Un altro parametro di cui tener nota è la correzione in ppm (il numero dopo il parametro “-p”) del quarzo della nostra chiavetta, di default è a 0 ma andrà aggiustato a seconda del nostro hardware

per rendere il tutto eseguibile usiamo:

```
chmod +x schedule_all.sh
chmod +x schedule_satellite.sh
chmod +x receive_and_process_satellite.sh
```

e poi rendiamo il tutto autonomo:

** i file devono avere uwner e gruppo l'utente root

```
export EDITOR=vim
crontab -e
```

inseriamo qui queste due righe:

```
1 0 * * * /home/pi/weather/predict/schedule_all.sh
@reboot /home/pi/weather/predict/schedule_all.sh
```

Un ultimo riavvio e siamo pronti !

Ad ogni passaggio, nella cartella “weather”, verranno salvate tre immagini e il file audio del passaggio (nel caso in cui vorremo rielaborarlo nuovamente).

Possiamo in ogni momento controllare i prossimi passaggi con il comando:

atq
ottendendo questo output:

```
pi@weather-satellite:~ $ atq
1007    Sat Jul 16 19:57:00 2022 a pi
1008    Sat Jul 16 21:38:00 2022 a pi
1015    Sat Jul 16 20:00:00 2022 a pi
1012    Sat Jul 16 23:13:00 2022 a pi
1011    Sat Jul 16 21:34:00 2022 a pi
```