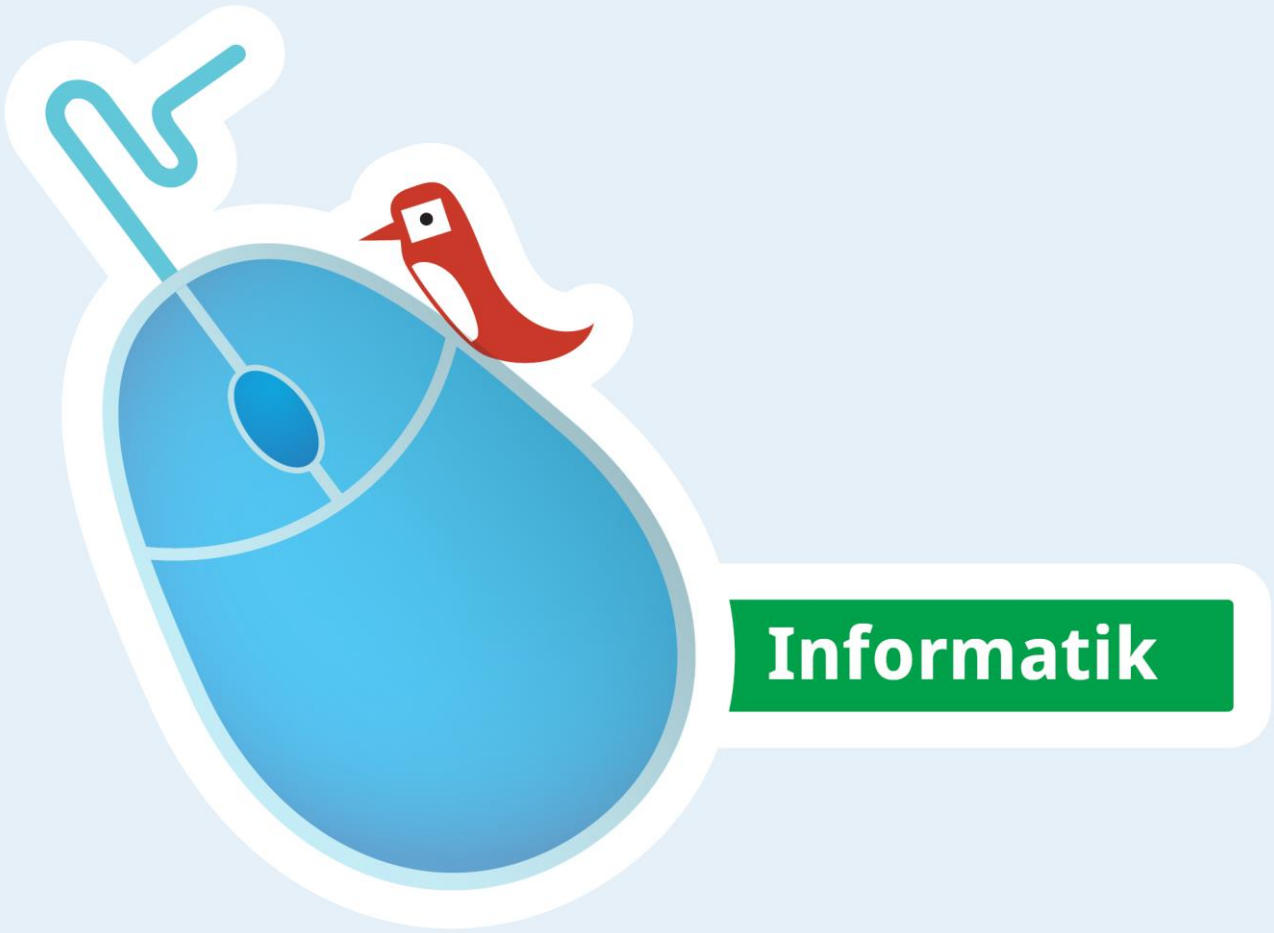


Unterrichtsmaterialien zum Thema

Raspberry Pi - NOAA Satellitenbilder

Klasse 10-13

Material für SchülerInnen

Übersicht – NOAA Satellitenbilder

!!! Wichtig für die gesamte Installation !!!

1. Durch Klicken auf blau hinterlegte Wörter gelangt ihr zu der jeweiligen Homepage oder zu einem Wikipedia-Artikel, um weitere Informationen zu erhalten.

2. Achtet darauf, dass der Raspberry Pi zu jedem Zeitpunkt während der gesamten Installation mit dem Internet verbunden ist!

3. Achtet darauf was der Raspberry Pi ausgibt! Sollten Errors oder Fehlercodes auftreten, dann wiederholt den jeweiligen Schritt.

Materialliste

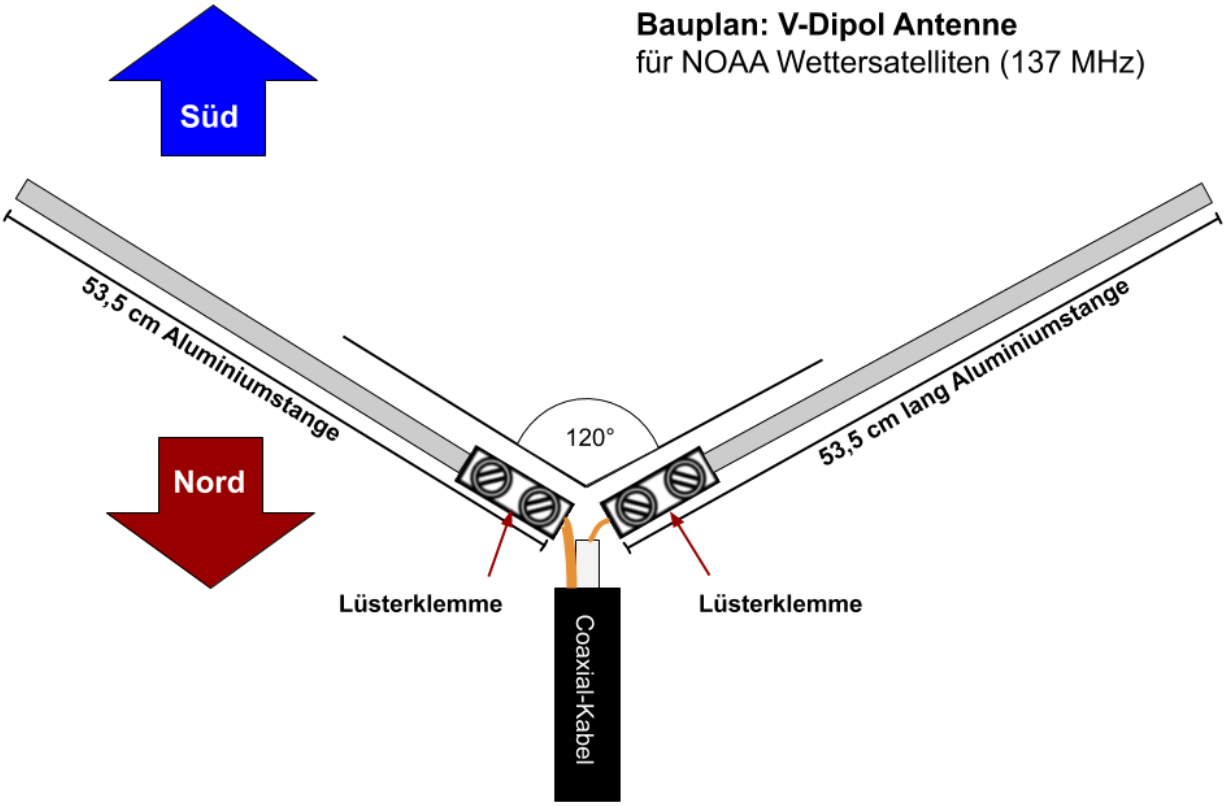
Allgemein:

- ☐ Raspberry Pi 3 (Model B oder B+)
- ☐ Netzteil und oder eine Powerbank
- ☐ SD-Karte (mind. 16Gb)
- ☐ Einen RTL-SDR-USB-Stick (z.B. den Nooelec NESDR SMArt v4)
- ☐ Maus, Tastatur & Monitor
- ☐ Einen weiteren Computer (zur Installation des Raspberry Pi)
- ☐ Eine konstante Internetverbindung für den Installationsprozess
- ☐ Eine kurzzeitige Internetanbindung (ggf. W-LAN Hotspot) am späteren Antennen-standort

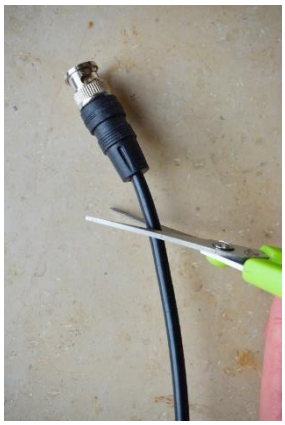
Für den Bau der Antenne:

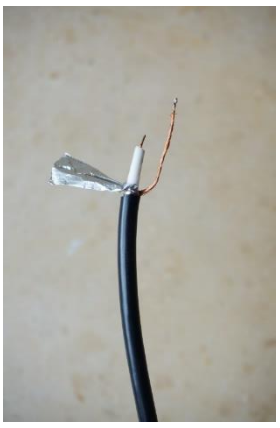
- ☐ 2x Aluminiumstange, mind. 55cm lang & 3,4mm Ø
- ☐ 2x Lüsterklemme
- ☐ Coaxial-Kabel mit 50Ω & SMA-Anschluss
- ☐ Holzbrett o.ä.
- ☐ Klebeband
- ☐ Abisolierzange
- ☐ Metallsäge
- ☐ Stift & Schere

Bauanleitung: V-Dipol-Antenne



1. Schneidet zuerst das eine Ende des Kabels ab. Legt dann die innere Kupferlitze sowie die äußere Kupferummantelung frei.





2. Die NOAA Wettersatelliten senden auf unterschiedlichen, aber nebeneinanderliegenden Frequenzen:

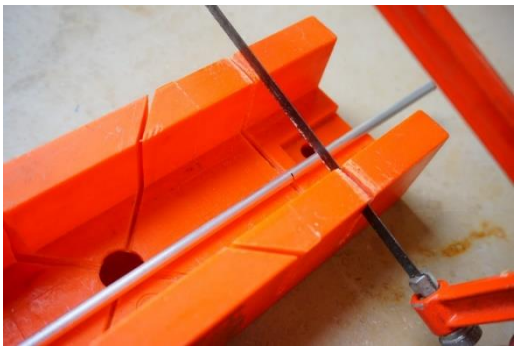
NOAA 15 = 137.620 MHz
NOAA 18 = 137.9125 MHz
NOAA 19 = 137.100 MHz
} ca. **137.55 MHz**

137.55 MHz entspricht dabei ungefähr einer Wellenlänge von λ = 2.18 m. Die ideale Länge des Dipols, also unserer Stangen, sollte insgesamt die Hälfte der zu empfangenden Wellenlange betragen: $\frac{\lambda}{2} = \frac{2{,}18m}{2} = 1{,}09\ \mathrm{m}$

Da sich die Hälfte der Wellenlänge (1,09 m), auf die Länge beider Stangen zusammen bezieht, muss die Länge noch einmal halbiert werden:

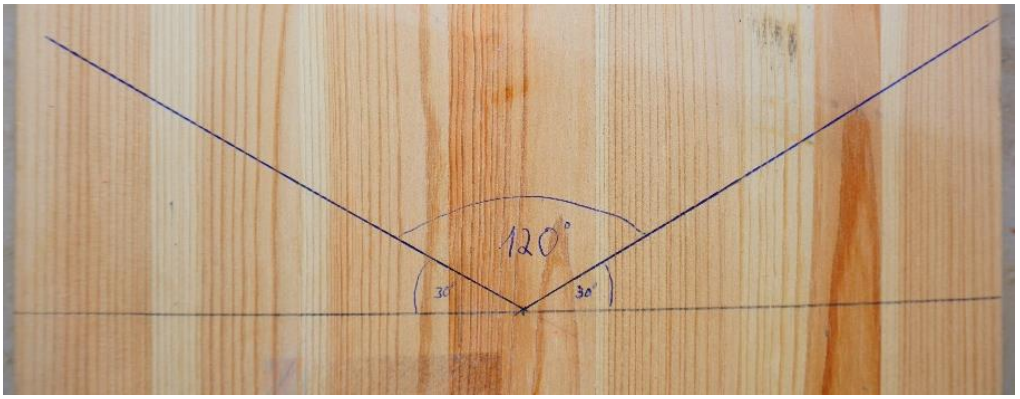
$$\frac{\lambda}{2} : 2 = \frac{\lambda}{4} = \frac{2.18m}{4} = 54{,}5\ cm$$

Da die Lüsterklemmen sowie das offene Ende des Coaxial-Kabels ebenfalls leiten, reduziert sich die Länge der beiden Stäbe noch einmal um ca. 1 cm, sodass die beiden Stäbe jeweils ca. 53,5 cm lang sein müssen.
Sägt mit der Metallsäge die Aluminiumstangen auf **53,5 cm** zu.



3. Befestigt mithilfe der Lüsterklemmen jeweils eine Stange mit der inneren Kupferlitze und eine Stange mit der äußeren Ummantelung.

4. Wie in der Skizze zu sehen ist, müssen die beiden Stangen in einem Winkel von 120° zueinander angebracht sein. Skizziert einen Winkel von 120° auf eurer Unterlage



5. Nun könnt ihr mit Klebeband die Stangen auf die gezeichneten Linien kleben und befestigen. Stabilisiert ggf. ebenfalls eure Lüsterklemmen mit etwas Klebeband, sodass sich an eurer Konstruktion nichts mehr bewegen kann.
Schließt das andere Ende an den Raspberry Pi an und fertig ist eine einfache Antenne zum Empfangen von Bildern aus dem All!

Weitere Möglichkeiten für Antennen zum Empfangen der NOAA-Satelliten findet ihr hier:

- Quadrifilar Helix Antenne (QFH)
- Kreuzdipole Antenne
- Doppelkreuz Antenne

6 Wichtig bei der Platzierung eurer Antenne:

- Beachtet die Ausrichtung! Die Stäbe müssen Richtung Süden zeigen!
- Platziert die Antenne möglichst hoch und freistehend, damit keine Objekte wie Bäume oder Gebäude den Empfang stören können.
- Meidet die Nähe zu Metallgegenständen in unmittelbarer Umgebung!
- Keine Bodennähe! Haltet ausreichend Abstand zum Boden, da bereits die Feuchtigkeit im Boden zu einem schlechten Ergebnis führen kann.
- Für den Fall, dass es währenddessen regnen sollte, wickelt den Raspberry Pi und die Powerbank in eine Plastiktüte, um Feuchtigkeit von der Elektronik fern zu halten!

Die Programmierung

1) Vorbereitung

Als Erstes muss ein passendes Betriebssystem mit Hilfe eines weiteren Computers (Windows, macOS, Ubuntu) auf die SD-Karte aufgespielt werden. Hierzu bieten sich am besten der Raspberry Pi Imager an, welcher hier: https://www.raspberrypi.org/downloads/ heruntergeladen werden kann.

Als Betriebssystem wählt zunächst „Raspberry Pi OS (other)“ und dann „**Raspberry Pi OS Full (32-Bit)**“ aus.

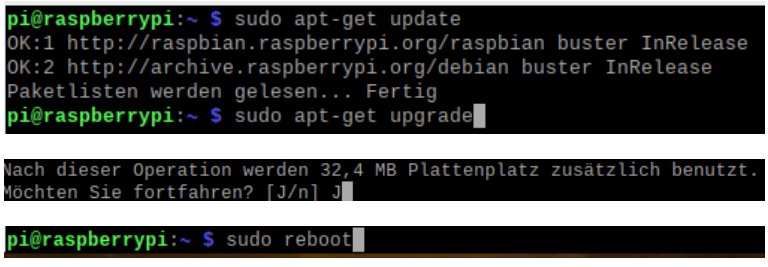
Eine ausführliche Anleitung der Installation von Raspbian (dem Betriebssystem) auf dem Raspberry Pi findet ihr hier: https://www.raspberrypi.org/blog/raspberry-pi-imager-imaging-utility/

Schließt Maus, Tastatur und Monitor an den Raspberry Pi an und schiebt die SD-Karte in den SD-Kartenslot. Sobald ihr das Netzteil oder eine Powerbank an dem Raspberry Pi angeschlossen habt, fährt er direkt hoch.

Nachdem unser Raspberry Pi das erste Mal hochgefahren ist, müsst ihr ihn Einrichten und mit dem Internet, entweder über LAN oder W-LAN, verbinden. Dieser Schritt kann, je nach Internetanbindung, einige Zeit in Anspruch nehmen. Denkt daran, dass ihr während der gesamten Installation eine stabile Internetverbindung benötigt!

Bevor Ihr mit dem Einrichten des Satelliten-Receivers anfangen könnt, muss das System auf den aktuellen Stand gebracht werden. Öffnet dazu das Terminal  , welches sich oben links in der Programmzeile befindet. Gebt nacheinander folgende Befehle ein:

```
sudo apt-get update
sudo apt-get upgrade
sudo reboot
```



Ggf. müsst ihr bei der Installation von Paketen oder Updates diese noch einmal mit „Y“ für Yes oder „J“ für Ja bestätigen!

Kopiert euch auch dieses Dokument auf den Raspberry Pi, damit ihr den Code in den späteren Schritten einfach kopieren könnt.

2) Installation der Software

Für die weitere Installation benötigt ihr auch weiterhin das Terminal. Zuerst muss der USB-Treiber für den RTL-SDR-USB-Stick installiert werden.
Steckt dazu den RTL-SDR-USB-Stick in einen der USB-Ports des Raspberry Pi und gebt den folgenden Befehl in das Terminal ein: → mit „Y“ oder „J“ bestätigen

```
sudo apt-get install libusb-1.0
```

```
pi@raspberrypi:~ $ sudo apt-get install libusb-1.0
```

Um anschließend das Programm erstellen zu können, benötigt ihr das Programmierwerkzeug CMake.
Gebt dazu den folgenden Befehl in das Terminal ein: → mit „Y“ oder „J“ bestätigen

```
sudo apt-get install cmake
```

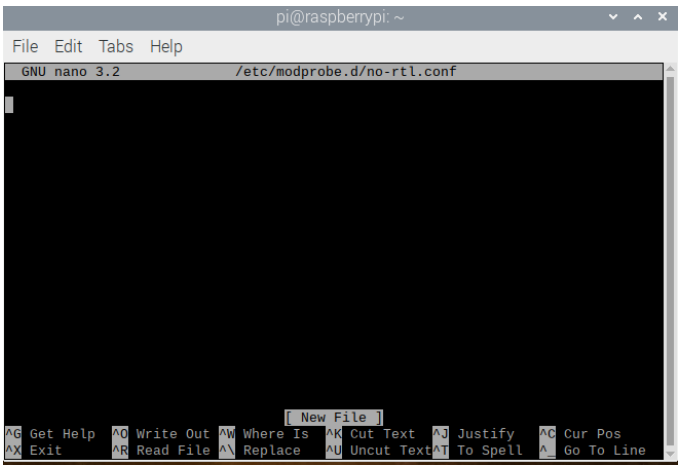
```
pi@raspberrypi:~ $ sudo apt-get install cmake
```

Als nächstes müsst ihr sicherstellen, dass der Raspberry Pi keine Hintergrundprozesse ausführt, welche später mit der Software interferieren könnten. Dazu müsst ihr mit Hilfe des nano-Texteditor eine neue Konfigurationsdatei mit dem Namen „no-rtl.conf“ in dem Dateipfad /etc/modprobe erzeugen.
Gebt dazu den folgenden Befehl in das Terminal ein:

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

```
pi@raspberrypi:~ $ sudo nano /etc/modprobe.d/no-rtl.conf
```

Nun habt ihr in dem genannten Dateipfad eine neue leere Konfigurationsdatei angelegt und der Texteditor hat diese bereits geöffnet. Ein solches Fenster sollte jetzt bei euch im Terminal erscheinen:



Kopiert die folgenden drei Zeilen und fügt sie in den geöffneten Texteditor ein:

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

Speichert die Einträge mit Strg + O, drückt Enter und schließt den Texteditor mit Strg + X.

Jetzt muss die Software für den RTL-SDR-USB-Stick (Software Defined Radio) mit Hilfe von GitHub installiert werden. Gebt dazu nacheinander die folgenden Befehle in das Terminal ein.
Ggf. können einzelne Schritte etwas länger dauern.

```
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/

sudo reboot
```

Als nächstes benötigt ihr das Programm „SoX“, um die empfangenden Audiosignale der Satelliten bearbeiten zu können.
Gebt dazu den folgenden Befehl in das Terminal ein: → mit „Y“ oder „J“ bestätigen

```
sudo apt-get install sox
pi@raspberrypi:~ $ sudo apt-get install sox
```

Des Weiteren benötigt ihr den „AT-Befehlssatz“ zur Positionsbestimmung.
Gebt dazu den folgenden Befehl in das Terminal ein: → mit „Y“ oder „J“ bestätigen

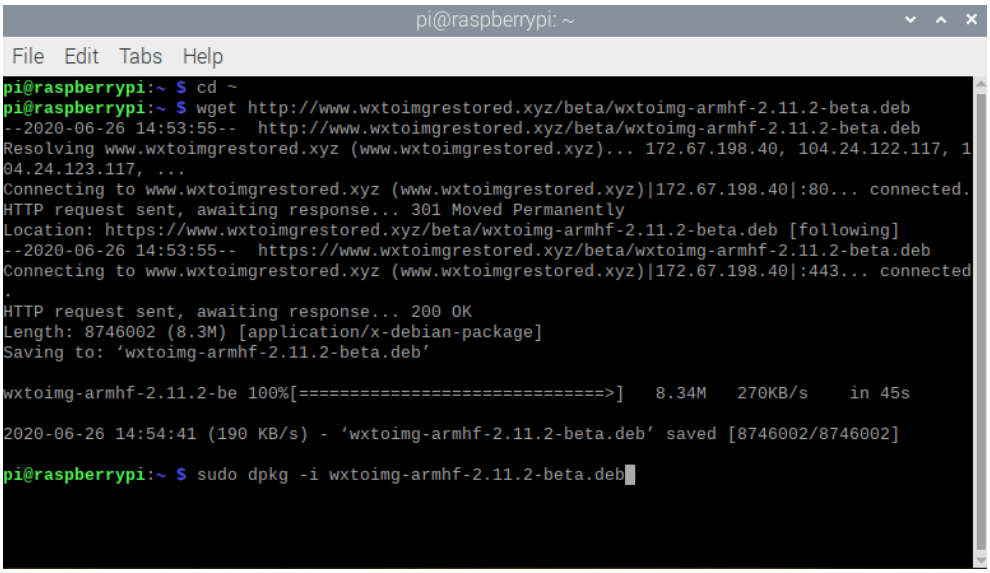
```
sudo apt-get install at
pi@raspberrypi:~ $ sudo apt-get install at
```

Um herauszufinden, wann die Satelliten exakt über eure Position fliegen, benötigt ihr das Programm „predict“.
Gebt den folgenden Befehl in das Terminal ein:

```
sudo apt-get install predict
pi@raspberrypi:~ $ sudo apt-get install predict
```

Als letztes benötigt ihr das Programm „wxtoimg“. Dieses Programm dekodiert die Audiosignale der Satelliten in das endgültige Satellitenbild um.
Gebt dazu die folgenden Befehle nacheinander in das Terminal ein:

```
cd ~
wget http://www.wxtoimgrestored.xyz/beta/wxtoimg-armhf-2.11.2-beta.deb
sudo dpkg -i wxtoimg-armhf-2.11.2-beta.deb
```

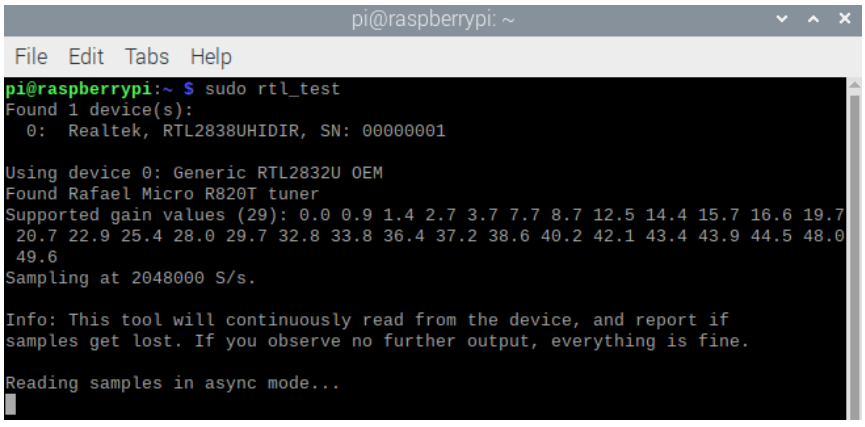


Jetzt habt ihr die notwendigen Softwarepakete installiert und könnt das Set-Up testen.
Gebt den folgenden Befehl in das Terminal ein, um zu sehen, ob die bisherige Installation erfolgreich war:

```
sudo rtl_test
```

```
pi@raspberrypi:~ $ sudo rtl_test
```

Jetzt sollte folgendes angezeigt werden:



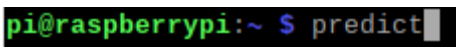
Sollte eure Ausgabe von diesem Bild abweichen, so ist irgendetwas während der vorherigen Schritte schiefgelaufen. Wiederholt dann die vorherigen Schritte erneut, da ihr erst fortfahren könnt, wenn eure Ausgabe mit der auf dem Bild identisch ist.
Sofern die Ausgabe gleich ist, könnt ihr mit Strg + C das Ausgabefester schließen und fortfahren.

3) Einrichtung

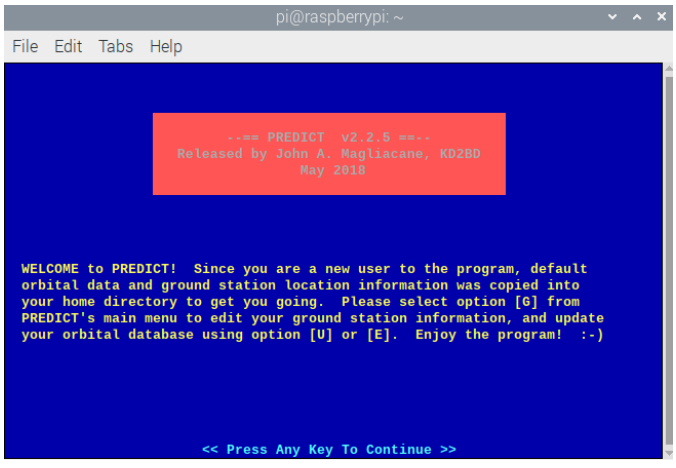
Als nächstes muss das zuvor installierte Programm „predict“ ausgeführt werden. Hier müsst ihr dem Programm eure Position nennen, damit es später möglichst exakt vorhersagen kann, wann und welcher Satellit genau über euch fliegen wird.
Dazu benötigt ihr den Breitengrad und den Längengrad eures Standortes bzw. des späteren Standortes der Antenne. Am besten findet ihr diese mit Hilfe von Google Maps heraus, indem ihr an einen beliebigen Punkt in der Nähe von eurer Position auf die Karte klickt oder die Standortfunktion verwendet. Beispielsweise lauten die Koordinaten der Ruhr-Universität Bochum Google Maps: 51.443790, 7.261788. Die erste Koordinate gibt den Breitengrad und die zweite Koordinate den Längengrad an. In diesem Fall bedeutet das, dass die Ruhr-Universität Bochum bei 51.443790° nördlicher Breite und 7.261788° östlicher Länge liegt. In Google Maps werden die nördlichen Breitengrade positiv und die südlichen als negativ, sowie die östlichen Längengrade als positiv und die westlichen als negativ angezeigt.
Bei „predict“ werden die Längengrade genau andersherum angegeben. Dies hat zur Folge, dass ihr vor dem Längengrad ein „-„ eingeben müsst, wenn ihr euch östlich des Nullmeridians befindet und eure Koordinaten in das Programm eingebt.

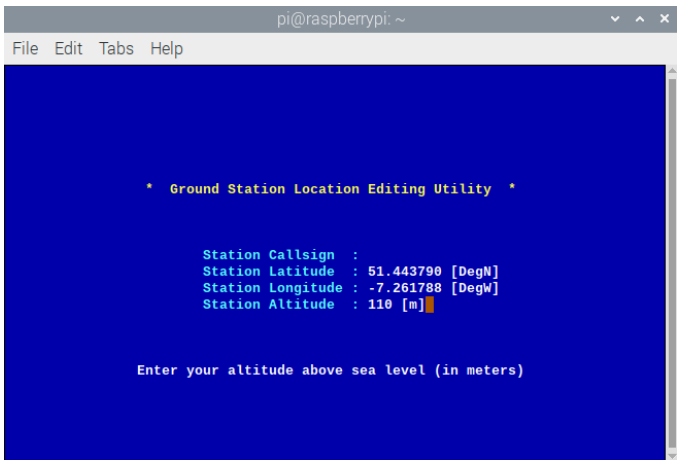
Gebt predict in das Terminal ein:

```
predict
```



Jetzt sollte dieses Fenster Erscheinen:



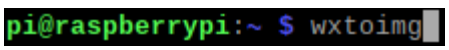


Drückt Enter, damit ihr eure Koordinaten eingeben könnt. Lasst die Eingabe in „Station Callsign“ frei. Gebt die erste Koordinate (Breitengrad) bei „Station Latitude“ ein und die zweite Koordinate (Längengrad) bei „Station Longitude“.
Denkt daran, dass die zweite Koordinate (Längengrad) in Deutschland immer östlich des Bezugsmeridianes (Greenwich) liegt und ihr somit das Minuszeichen vor eure Koordinate schreiben müsst!
Zuletzt gebt ihr die Höhe eures Standortes bei „Station Altitude“ ein. Diese findet ihr mit Google Earth oder anderen Apps heraus. Im Gegensatz zu euren Koordinaten muss die Höhenabgabe aber nicht sehr genau sein.
Für das Beispiel der Ruhr-Universität Bochum würde es wie im Bild links aussehen.
Bestätigt eure eigenen Eingaben mit Enter, schließt das Fenster und öffnet das Terminal erneut.

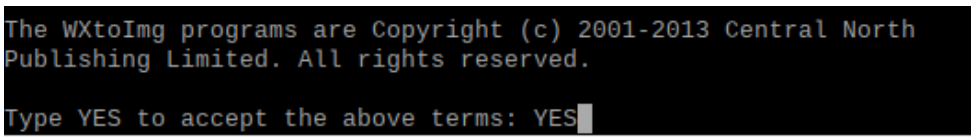
Im zweiten Schritt habt ihr die Software „wxtoimg“ bereits installiert. Jetzt müsst ihr noch den AGBs des Programms zustimmen, um es auf euren Raspberry Pi benutzen zu können.

Gebt wxtoimg in das Terminal ein:

```
wxtoimg
```

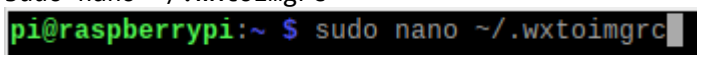


Lest euch die AGBs durch und gebt anschließend „YES“ ein, um die AGBs zu bestätigen.



Wie in dem Schritt zuvor müsst ihr auch „wxtoimg“ eure Position nennen. Das Programm legt dann eine Overlay-Map mit den jeweiligen Ländergrenzen über das Satellitenbild, was euch am Ende bei der Orientierung und Auswertung hilft.
Erstellt mit dem nano Texteditor ein neues Dokument in /.wxtoimgrc:

```
sudo nano ~/.wxtoimgrc
```



Beim Eingeben der Koordinaten ist zu beachten, dass diesmal die Koordinaten identisch mit denen in Google Maps sind, weswegen ihr diesmal kein Minuszeichen vor den Längengrad schreiben dürft! Für den Fall der Ruhr-Universität Bochum würde der Eintrag so aussehen:

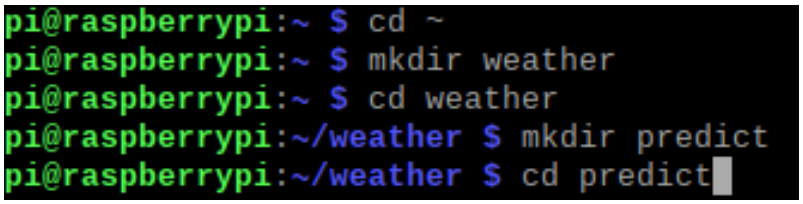
```
Latitude: 51.443790
Longitude: 7.261788
Altitude: 100m
```

Kopiert die drei Zeilen in den geöffneten Texteditor und schreibt statt der Koordinaten der Ruhr-Universität Bochum eure eigenen Koordinaten hinein.
Speichert wieder mit Strg + O, drückt Enter und schließt den Editor mit Strg + X.

4) Die Programme

Nachdem ihr jetzt alle notwendigen Programme installiert und eingerichtet habt, könnt ihr euch an den finalen Schritt wagen – den Code zum Empfangen der Satellitenbilder.
Insgesamt müsst ihr dazu drei Skripte erstellen: „schedule_all.sh“, „schedule_satellite.sh“ und „receive_and_process_satellite.sh“.
Hierzu müsst ihr zunächst mit den folgenden Befehlen im Terminal die Ordnerstruktur anlegen:

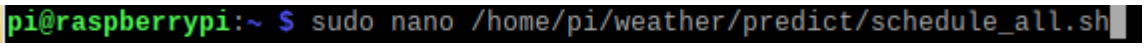
```
cd ~
mkdir weather
cd weather
mkdir predict
cd predict
cd ~
```



Jetzt sollte ein neuer Ordner in /home/pi mit dem Namen „weather“ angelegt sein.
In diesem Ordner sollte sich der Ordner „predict“ befinden.

Das erste Skript „schedule_all.sh“ erstellt ein Textfile, welches die Uhrzeiten der Überflüge der NOAA 15, 18 und 19 Wettersatelliten an eurer Position nennt. Darüber hinaus benötigt das Programm „predict“ diese Informationen, um später zum richtigen Zeitpunkt die Signale der Satelliten empfangen zu können.
Erstellt das Script „schedule_all.sh“ in dem Ordner /home/pi/weather/predict mit:

```
sudo nano /home/pi/weather/predict/schedule_all.sh
```



Kopiert das folgende Skript und fügt es in den geöffneten Editor ein. Speichert wie gewohnt die Eingabe mit Strg + O, drückt Enter und schließt den Editor mit Strg + X.

```
#!/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

atq > /home/pi/weather/passes_`date +"%m-%d-%Y"`.txt
```

Das zweite Skript „schedule_satellite.sh“ ermittelt, ob der Überflug des Satelliten einen kleineren Winkel als 20 Grad von eurer Position hat. Sollte der Winkel kleiner als 20 Grad sein, wird das Programm diesen Satelliten ignorieren und auch nicht versuchen, Daten zu empfangen, da bei einem solchen flachen Winkel in der Regel kein vernünftiges Bild entstehen wird.
Erstellt das zweite Skript „schedule_satellite.sh“ in dem Ordner /home/pi/weather/predict mit:

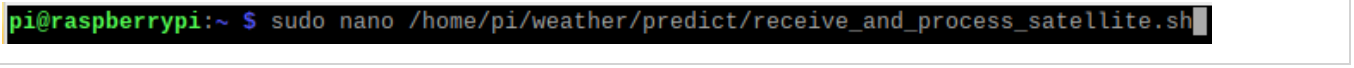
```
sudo nano /home/pi/weather/predict/schedule_satellite.sh
```

Kopiert wieder den folgenden Code und fügt ihn in den Texteditor ein. Speichert mit Strg + O, Enter und schließt den Editor mit Strg + X.

```
#!/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
```

Das letzte Skript „receive_and_process_satellite.sh“, soll die Audiosignale der Satelliten empfangen, speichern und auch am Ende verarbeiten. Sobald dieses Skript ausgeführt wird, nutzt es „*rtl_fm*“, um die Audiosignale zu empfangen. Anschließend verarbeitet das Programm „*SoX*“die Signale weiter und speichert diese ab. Sobald der Überflug der Satelliten vorbei ist und die Daten abgespeichert sind, startet das Programm „*wxtoimg*“, welches die eigentlichen Bilder aus den Audiosignalen erstellt und die Overlay-Map darüberlegt.
Erstellt das dritte Skript „receive_and_process_satellite.sh“ in dem Ordner /home/pi/weather/predict mit:

```
sudo nano /home/pi/weather/predict/receive_and_process_satellite.sh
```

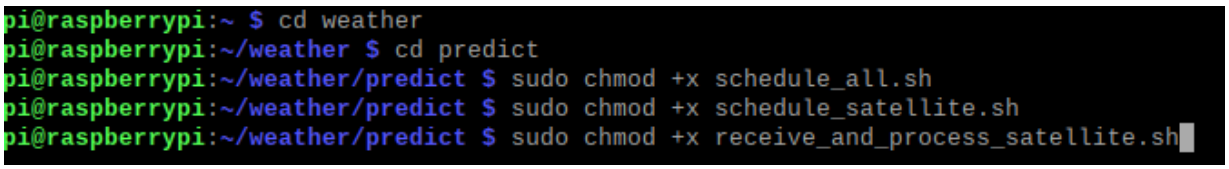


Kopiert wieder den folgenden Code und fügt ihn in den Texteditor ein. Speichert mit Strg + O, drückt Enter und schließt den Editor mit Strg + X.

```
#!/bin/bash

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

# reads and creates folder with current date / time (i.e 05-30-2019_07-48 *windows friendly*)
NOW=$(date +%m-%d-%Y_%H-%M)
mkdir /home/pi/weather/Folder${NOW}

sudo timeout $6 rtl_fm -f ${2}M -s 60k -g 45 -p 55 -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
/usr/local/bin/wxtoimg -m ${3}-map.png -e ZA $3.wav ${3}.png
/usr/local/bin/wxtoimg -m ${3}-map.png -e NO $3.wav ${3}.NO.png
/usr/local/bin/wxtoimg -m ${3}-map.png -e MCIR $3.wav ${3}.MCIR.png
/usr/local/bin/wxtoimg -m ${3}-map.png -e MSA $3.wav ${3}.MSA.png

fi

# copies files to the new folder and deletes the original ones
cp /home/pi/weather/*.png /home/pi/weather/Folder${NOW}/
cp /home/pi/weather/*.wav /home/pi/weather/Folder${NOW}/
rm /home/pi/weather/Folder${NOW}/*-map.png
rm /home/pi/weather/*.png
rm /home/pi/weather/*.wav
```

Nun habt ihr alle Skripte, die ihr benötigt, erstellt. Jetzt müsst ihr diese in das Betriebssystem implementieren.
Gebt dazu die folgenden Befehle nacheinander in das Terminal ein:

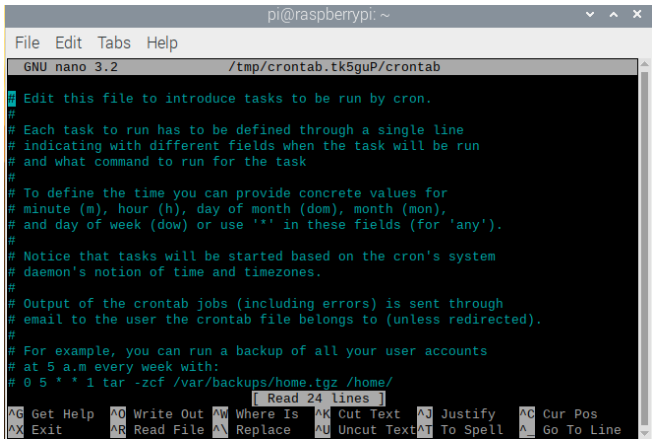
```
cd ~
cd weather
cd predict
sudo chmod +x schedule_all.sh
sudo chmod +x schedule_satellite.sh
sudo chmod +x receive_and_process_satellite.sh
```



Um den Raspberry Pi über einen längeren Zeitraum laufen zu lassen (z.B. mehrere Tage), und um möglichst viele gute Satellitenbilder zu erhalten, sollte das Skript „schedule_all.sh“ automatisiert ablaufen. „schedule_all.sh“ soll dann immer kurz nach Mitternacht starten, um die Überflüge der NOAA Satelliten für den gesamten neuen Tag abzurufen.
Gebt dazu den folgenden Befehl in das Terminal ein und wählt mit „1“ den nano Texteditor aus:

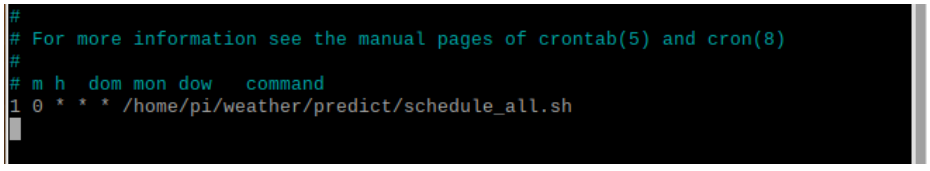
```
crontab -e
```

Jetzt sollte sich dieses Fenster öffnen:



Kopiert den folgenden Befehl und fügt ihn am Ende des Textes ein. Speichert den Eintrag mit Strg + O, drückt Enter und schließt den Texteditor mit Strg + X.

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



Jetzt wird bei eingeschaltetem Raspberry Pi, immer um 00:01 Uhr das Skript „schedule_all.sh“ ausgeführt.

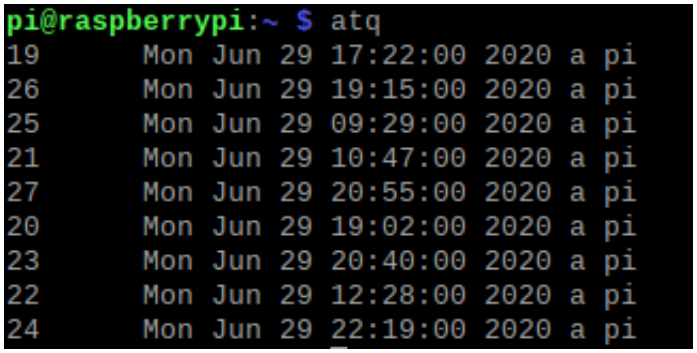
5) Finaler Check

Gebt diesen Befehl in das Terminal ein, um das Programm „schedule_all.sh“ auszuführen.

```
cd ~

/home/pi/weather/predict/schedule_all.sh
```

Damit werden alle kommenden Satelliten angezeigt, und im Ordner /home/pi/weather wird eine Textdatei erstellt, welche die heutigen Satellitenüberflüge beinhaltet.
Gebt „atq“ in das Terminal ein, um die heutigen Satellitenüberflüge in einer Übersicht anzeigen zu lassen. Diese beziehen sich auf die Unix-Zeit bzw. UTC.

```
atq
```



Nun wisst ihr, wann die Satelliten über euren Standort fliegen werden.
Schließt den Raspberry Pi an eure Antenne an. Stellt dabei sicher, dass die Antenne möglichst hoch und frei steht, damit ihr das bestmögliche Ergebnis erzielt. Sollte sich die Antenne nicht in Netzteil-Reichweite einer Steckdose befinden, denkt daran, eine große, vollgeladene Powerbank an den Raspberry Pi anzuschließen und regelmäßig zu überprüfen. Der Austausch der Powerbank sollte nicht während eines Überflugs erfolgen.
Da der Raspberry Pi seine Uhrzeit über das Internet stellt, benötigt er zumindest bei jedem Starten kurzzeitig Internet, damit die Uhrzeiten der Satellitenüberflüge mit der Uhrzeit des Pi übereinstimmen. Wenn ihr also im Gelände euer Set-Up aufbaut, muss der Raspberry Pi z.B. über einen Hotspot durch euer Smartphone zumindest kurzzeitig mit dem Internet verbunden werden.

Wartet einige Satellitenüberflüge (oder Tage) ab und schließt den Raspberry Pi wieder an einen Monitor an. In dem Ordner weather (/home/pi/weather) sollten sich jetzt neue Ordner für jeden einzelnen empfangenden Satellitenüberflug befinden. In diesen Ordnern sind weitere Dateien zu finden:

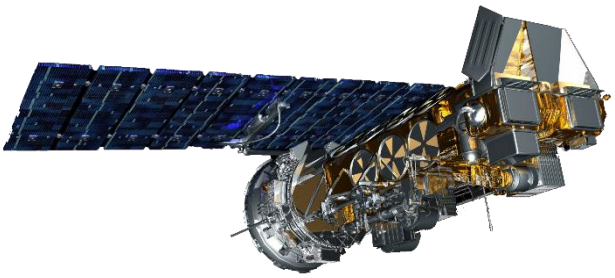
```
NOAA1820200528-132911-map.png     <-- Overlay-Map

NOAA1820200528-132911.png         <-- einfaches Satellitenbild

NOAA1820200528-132911.NO.png      <-- hebt die Wolken hervor

NOAA1820200528-132911.MCIR.png    <-- Map Colored Infrared (Colour Enhancement)

NOAA1820200528-132911.MSA.png     <-- Multi Spectral Analysis (Colour Enhancement)

NOAA1820200528-132911.wav         <-- Audio-Datei
```

Im ersten Teil des Dateinamens (Ziffer 1-6) wird der jeweilige Satellit angezeigt, von dem das Bild stammt. In diesem Fall also der NOAA 18 Satellit.
Anschließend (Ziffer 7-14) folgt das Datum in internationaler Schreibweise (YYYYMMDD). Hier der 2020.05.28, was in deutscher Schreibweise der 28.05.2020 ist.
Im letzten Teil (Ziffer 16-21) steht die genaue Uhrzeit (HHMMSS), an der begonnen wurde, das Bild aufzuzeichnen. Um 13:29 und 11 Sekunden wurde also angefangen, die Daten des Satelliten zu empfangen.

Der Raspberry Pi kann jedoch maximal einen Satelliten auf einmal verarbeiten. Sollten zwei Satelliten zum gleichen bzw. ähnlichen Zeitpunkt über eure Position fliegen, kann er im besten Fall nur den ersten Satelliten aufzeichnen.

Die Aufzeichnung und Verarbeitung solcher Satellitenbilder ist sehr komplex und auch stark abhängig von der Antenne bzw. ihrer Umgebung, sodass nicht jeder Überflug auch zwangsläufig ein schönes Satellitenbild liefert. Am besten lasst ihr die Programme für mehrere Satellitenüberflüge bzw. Tage laufen, damit die Wahrscheinlichkeit höher ist, auch schöne und fehlerfreie Satellitenbilder zu erhalten.

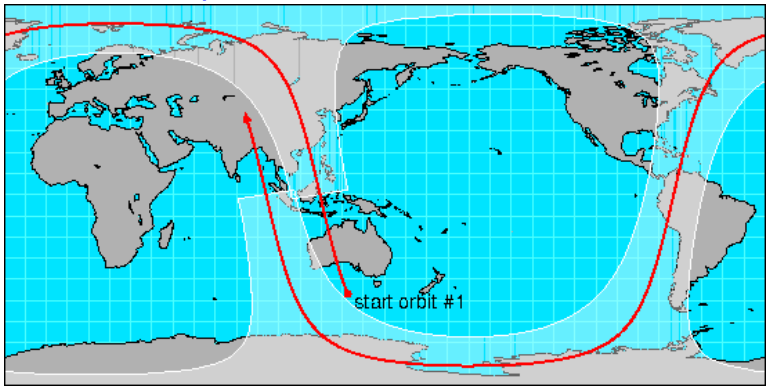
Auswertung Teil I



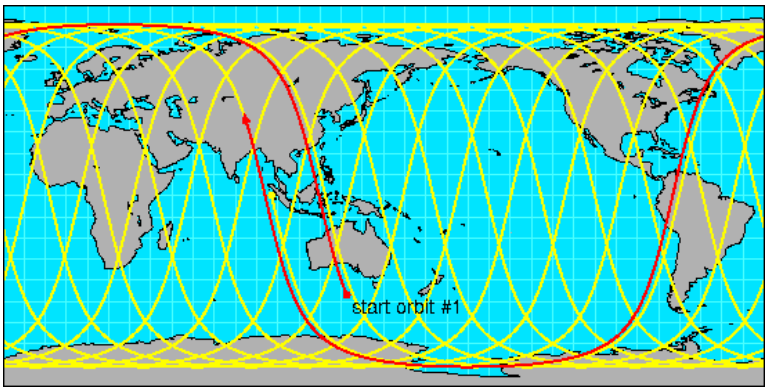
NOAA 18/19 (NOAA NESDIS Environmental Visualization) Laboratory)

Öffnet die .wav Datei von einem der Satelliten-überflüge und hört sie euch an.
Woran erinnert euch das Audio Signal, welches die NOAA-Satelliten aussenden?

Die Wettersatelliten der National Oceanic and Atmospheric Administration der USA (kurz NOAA), umkreisen sonnensynchron in einer Höhe von etwa 850 Kilometern die Erde. Wie auf den Abbildungen zu erkennen, umkreisen die Wettersatelliten die Erde immer von Pol zu Pol, also haben sie einen polaren Orbit.
Aktuell sind nur noch die Satelliten NOAA 15,18 und 19 aktiv. Diese senden kontinuierlich über UKW-Funk (137-138 MHz) Radiosignale aus, welche wir mit einer Antenne (z.B. der V-Dipol Antenne) empfangen können, wenn der Satellit genau über unserer Position ist. Die Signale können wir dann in Form von Audiosignalen wahrnehmen. Dabei geben die Töne, die ähnlich wie Morsecodes klingen, das aufgenommene Bild wieder. Jeder einzelne Ton, den ihr in der .wav Datei hören könnt, entspricht dem Farbwert in Graustufen eines jeweiligen Pixels des eigentlichen Satellitenbildes.
Diese Übertragungsart wird „Automatic Picture Transmission“ (kurz ATP) genannt und existiert bereits seit 1963.

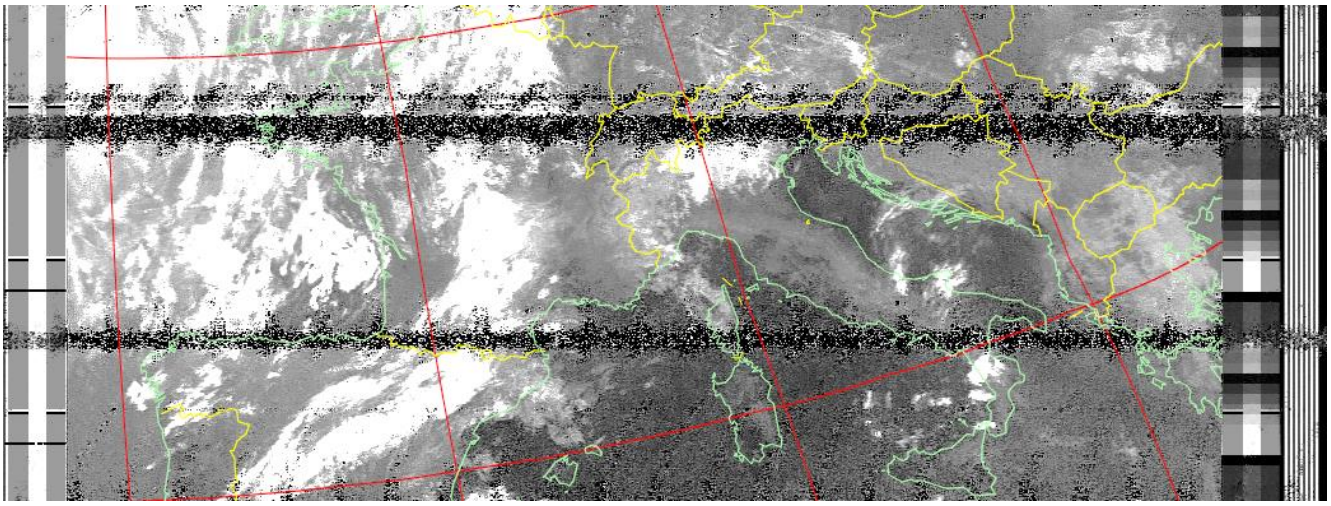


Ablauf eines typischen erdnahen polaren Orbits (tornado.sfsu)

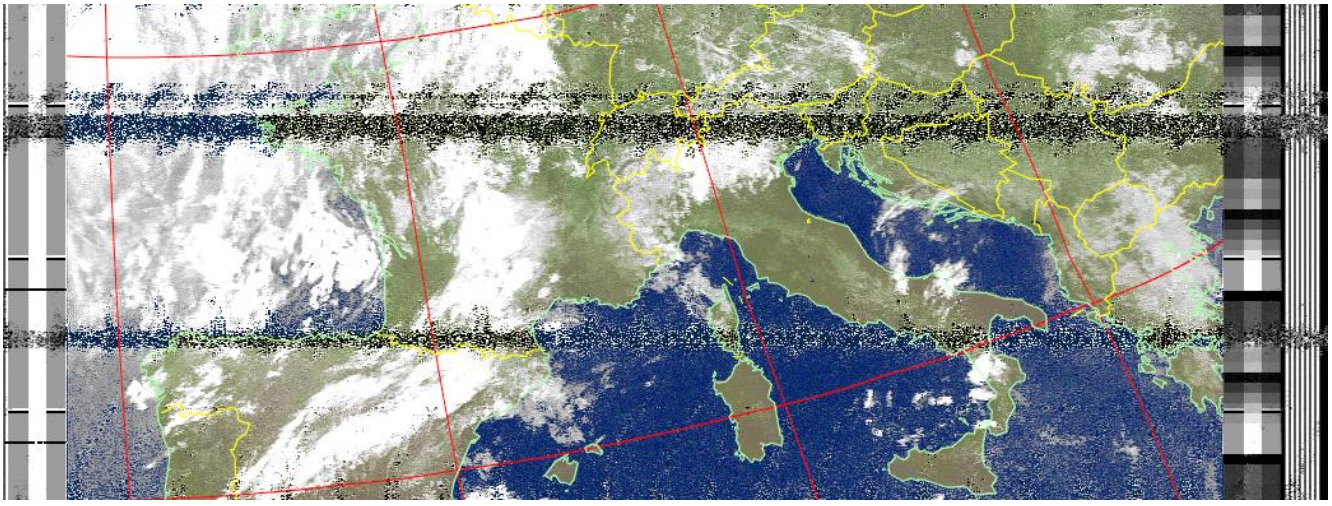


Ablauf nach mehrere polarer Umläufen eines Satelliten (tornado.sfsu)

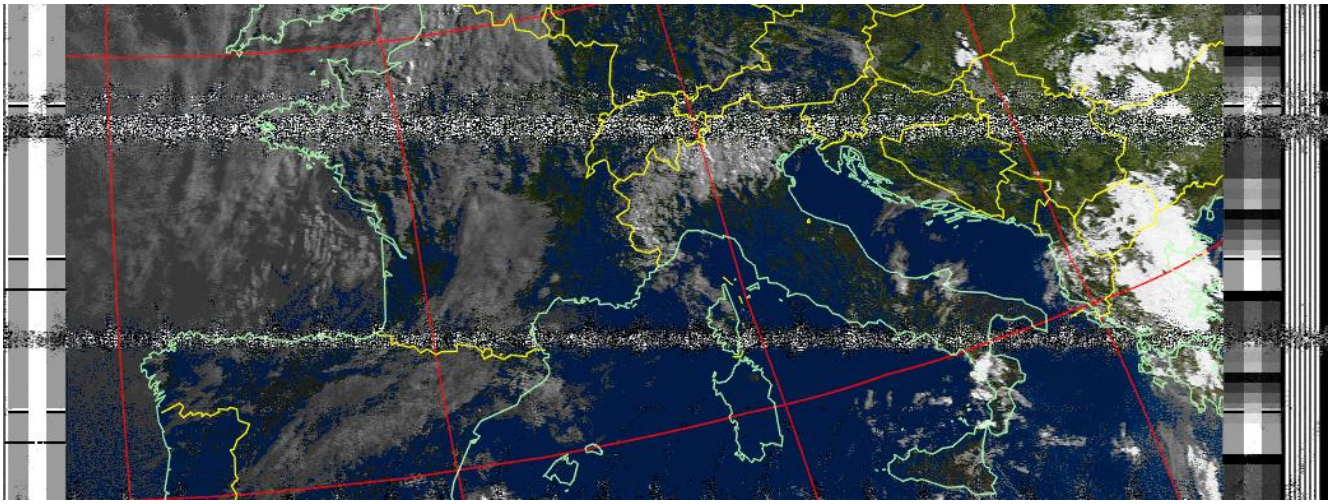
Auswertung Teil II



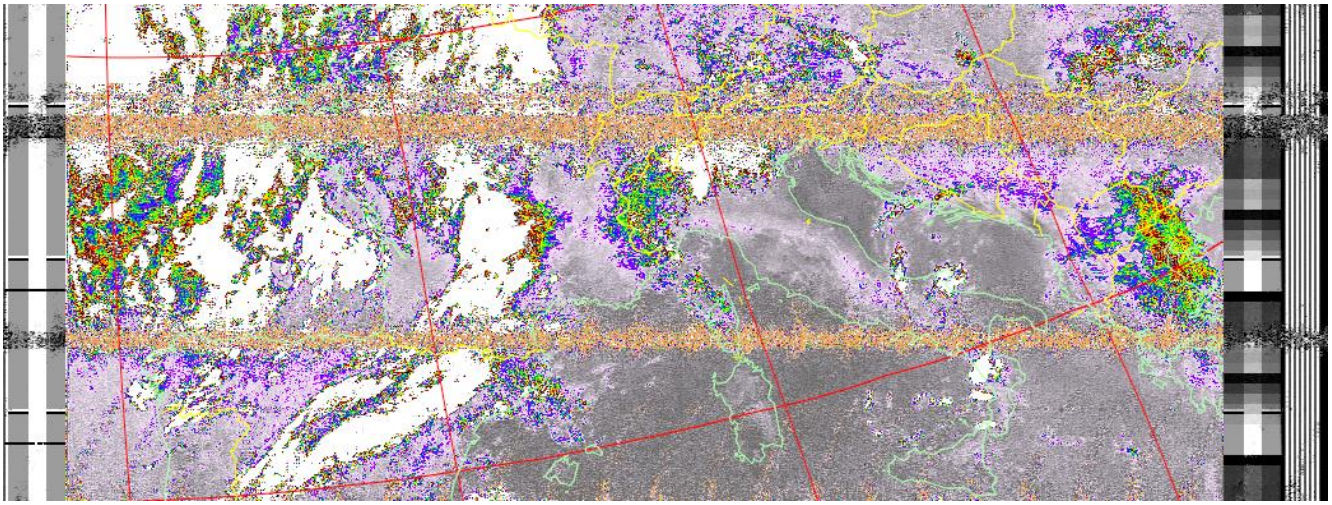
Dieses Graustufenbild (.png) ist das eigentliche Satellitenbild, welches aus der Audiodatei entsteht. Man kann zwar bereits zwischen Wolken, Land und Meer differenzieren, aber ansprechend ist das Bild nicht. Außerdem könnt ihr an den schwarzen, körnigen Streifen im Bild erkennen, dass der Empfang zwischenzeitlich gestört wurde.



In diesem Bild (MCIR) hat die Software „wxtoimg“ das ursprüngliche Graustufenbild bearbeitet. Zur besseren Orientierung wurden hier die Ozeane (Mittelmeer & Atlantik) blau und die Landmassen ohne Wolkenbedeckung grün eingefärbt.



In einem weiteren Schritt wurde die Uhrzeit bei der Bearbeitung dieses Bildes (MSA) durch die Bildbearbeitungssoftware berücksichtigt. Dieses Bild wurde am 14.07.2020 um 7:56 Uhr deutscher Zeit aufgenommen. Man kann sehen, dass die Sonne bereits im Osten aufgegangen ist, jedoch den Westen, Spanien und Portugal, noch nicht anstrahlt. Zu erkennen ist dies an der, verglichen mit dem linken Bereich, größeren Helligkeit im rechten Teil des Bildes.



Das letzte Bild (NO) versucht potentielle Regenwolken farblich hervorzuheben. Jedoch ist diese Form der Regenvorhersage sehr ungenau, da die Genauigkeit durch Bildfehler und damit auch durch unsere Antenne stark variiert.

Abschlussaufgabe:

Öffnet den jeweiligen Wetterbericht zum Aufnahmezeitpunkt des Satellitenbildes und vergleicht diese miteinander.
Stimmt der Wetterbericht mit der empfangenden Wetterkarte überein?