

## Webseite vorbereiten.

Kompletten- GPS mit Unterordnern in ein Verzeichnis auf dem Rechner kopieren, worauf der Webbrowser vom Rechner Zugriff hat. Das kann auf dem Pi sein, muss aber nicht. Zur späteren Analyse empfehle ich das nicht auf dem Pi zu installieren.

## Anpassen der Html Vorlage

Aus Google-Maps das passende Bild für den Hintergrund der Karte herausschneiden und als karte.png im Order GPS/pngs ablegen.

Am besten die gleiche Größe wie im html File 600x1024 verwenden, ist aber nicht zwingend erforderlich. Dann mit Hilfe von Google-Maps die Geodaten von der oberen Bildecke Links bestimmen in diesem Fall ist das Lat 53.32652 Lon 10.36567 danach die untere Bildecke rechts bestimmen. Hier ist es Lat 53.32626 Lon 10.36677

Nun im Quellcode der Analyse.html diese Daten in Zeile 109 und Zeile 110 eintragen.

Beim Laden der Analyse.html sollte nun ein Bild ähnlich diesem im Browser erscheinen.

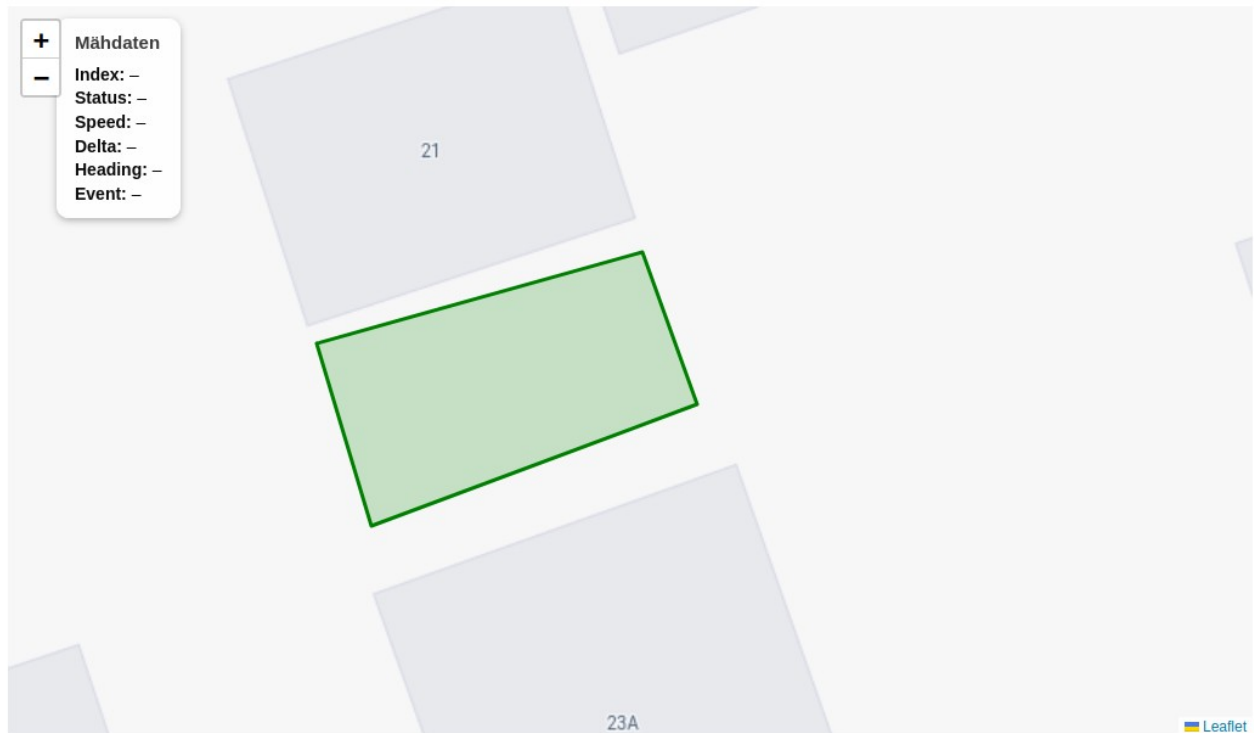


An dieser Stelle erkläre ich noch kurz wie man den Rasen in das Bild einfügen kann, die richtigen Koordinaten dazu kann aber später der Mähroboter, oder eine Vermessung mit dem GPS-RTK Empfänger liefern. Die Eckpunkte werden im Uhrzeigersinn aufgenommen und starten oben Links. Nehmen wir an, die Eckpunkte vom Rasen liegen bei 53.32634, 10.36597 53.32637, 10.36615 53.32632, 10.36618 53.32628, 10.36600 dann geben wir genau diese Daten in die Datei GPS/rasen/rasen.json ein. Achtung es ist eine json Datei so sollte dann so aussehen.

```
[  
  [53.32634, 10.36597],  
  [53.32637, 10.36615],  
  [53.32632, 10.36618],  
  [53.32628, 10.36600]  
]
```

Später kann durch Vermessung mit dem rtk\_rover.py die einzelnen Punkte genau bestimmt werden und das polygon in der Karte aktualisiert werden.

Nun sollte die Karte im html Bild in etwa so aussehen.



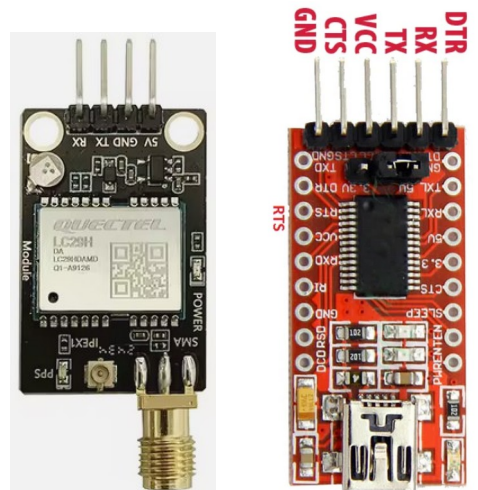
Die Bedienung der Seite ist intuitiv. Später ist gedacht, dass die mower.py ein logfile von dem Mähvorgang erstellt. Dieses logfile kann dann durch den Auswahlbutton selektiert und geladen werden. Jeder Eintrag in dem logfile liefert einen Datenpunkt auf der Karte. Neben den Geodaten habe ich geplant events zu protokollieren. Events wären dann z.B. Stops die ausgelöst werden und den Mäher zum Ausweichen bringen. So könnten events ausgelöst werden von „Perimeter“, „Bumper“, „Kamera“, „GPS“ oder vom „ToF“. Für jedes dieser events ist ein Icon im Ordner GPS/pngs hinterlegt. Zusätzlich gibt es noch für „Start“ und „Ziel“ ein entsprechendes Icon. Daher ist der Datensatz der jetzt schon eingelesen wird entsprechend komplexer, als im Moment erforderlich.

```
{ "time": "11:37:13.347", "lat": 52.12679645, "lon": 8.663306383333333, "fix": "RTK-Fixed",
"sats": "39", "speed": "0.01", "heading": 51.54, "delta": "1.15", "event": "", "image": "" },
```

Im Moment geht es derzeit erst einmal um die Grundfunktion der rtk\_rover.py auf dem Mäher sicherzustellen. Dazu ist die Analyse.html schon ein hilfreiches Tool.

## RTK-Rover

Mein Testaufbau ist derzeit folgender:  
Pi3 mit dem script rtk\_rover.py steht im Gartenhaus.  
Wichtig ist, dass er im WLAN erreichbar ist. Das ist später auch für den Mäher Voraussetzung, da er sonst keine Korrekturdaten erhalten kann.  
Die GPS Antenne YB0017AA liegt im Moment auf dem Dach des Gartenhauses und ist stationär. Über einen Serial => USB Adapter habe ich das Quectel- GPS Modul LC29H(DA) an den Pi 3 angeschlossen. Das Modul wird bei mir auf /dev/ttyUSB0 erkannt.



Im ersten Schritt betreibe ich das Modul über meinen SAPOS Account und dazu reicht auf dem Pi dann der Aufruf des scriptes ohne weiteren Parameter. Sollte später die RTK-Basisstation die Korrekturdaten liefern, so wird das Script mit `rtk_rover.py -b` aufgerufen (b für Basis). Damit der SAPOS Dienst genutzt werden kann sind im python script die entsprechenden Parameter zu setzen. Diese habe ich aus dem Informationsmaterial herausgezogen, das mir die SAPOS NRW zur Verfügung gestellt hat. Das kann bei Dir natürlich anders sein, und muss ggf. angepasst werden.

```
# === Konfiguration ===  
SERIAL_PORT = "/dev/ttyUSB0"  
BAUDRATE = 115200  
NTRIP_SERVER = "80.158.61.104"          # Quelle SAPOS NRW  
NTRIP_PORT = 2101                      # Quelle SAPOS NRW  
NTRIP_MOUNTPOINT = "VRS_3_3G_NW"      # Quelle SAPOS NRW * siehe Anmerkung  
NTRIP_USER = "myuser"  
NTRIP_PASS = "mypwd"
```

("VRS\_3\_3G\_NW" ist der Mountpoint der GPS Glonass und Galileo Korrekturdaten mit einer Genauigkeit von 1 – 2 cm bereitstellt.

Also Script Konfiguration ggf. anpassen und dann mit `python rtk_rover.py` starten.  
Die Ausgabe sieht derzeit so aus.

```
##### Hole Korrekturdaten von sapos #####  
[16:28:02.504] === RTK Rover Full Startup ===  
[16:28:02.505] === RTK bridge with watchdog starting ===  
[16:28:02.513] === LC29HDA Rover Init Start ===  
[16:28:02.513] [TX] $POTMVERN0*58  
[16:28:02.714] [TX] $POTMCFGRCVRMODE,R*32  
[16:28:02.915] [TX] $POTMCFGRCVRMODE,W,2*29  
[16:28:03.115] [TX] $POTMCFGRCVRMODE,R*32  
[16:28:03.316] [TX] $PAIR434,1*24  
[16:28:03.517] [TX] $PAIR062,0,01*0F  
[16:28:03.717] [DONE] LC29HDA ready for NTRIP startup ✓  
[16:28:03.718] [NTRIP] Connecting to 80.158.61.104:2101  
  
[16:28:03.729] Lat:52.1267968 Lon:8.6633066 [RTK-Fixed] [Δ=0.00 m] [speed:0.00][heading: 0][Sat: 35]  
[16:28:03.791] [NTRIP] Now Connected to SAPOS ✓  
  
[16:28:04.264] Lat:52.1267968 Lon:8.6633066 [RTK-Fixed] [Δ=0.00 m] [speed:0.02][heading: 51.54][Sat: 35]  
  
[16:28:05.230] Lat:52.1267968 Lon:8.6633066 [RTK-Fixed] [Δ=0.00 m] [speed:0.02][heading: 51.54][Sat: 35]  
[16:28:05.793] [NTRIP TX] GGA -> caster: $GNGGA,152805.000,5207.607808,N,00839.798395,E,4,35,0.48,64.443,M,47.176,M,4.0,1467*60  
[16:28:06.080] [NTRIP RX] 578 bytes -> LC29H  
  
[16:28:06.258] Lat:52.1267968 Lon:8.6633066 [RTK-Fixed] [Δ=0.00 m] [speed:0.02][heading: 51.54][Sat: 35]  
[16:28:07.081] [NTRIP RX] 578 bytes -> LC29H  
[16:28:07.082] [NTRIP TX] GGA -> caster: $GNGGA,152806.000,5207.607808,N,00839.798395,E,4,35,0.48,64.442,M,47.176,M,5.0,1467*63  
[16:28:07.089] [NTRIP RX] 96 bytes -> LC29H
```

Ist die Entwicklung des Scriptes abgeschlossen, dann wird die Ausgabe nur in einem debug Aufruf zu sehen sein. Die für die SW erforderlichen Daten werden in zwei logFiles gespeichert. Für die mower.py werden jetzt schon sekundlich die Geo Daten nach `/dev/shm/rtk_point.json` geschrieben. Diese sollen später sekundlich ausgewertet und wirken sich auf die Automatisierung aus.

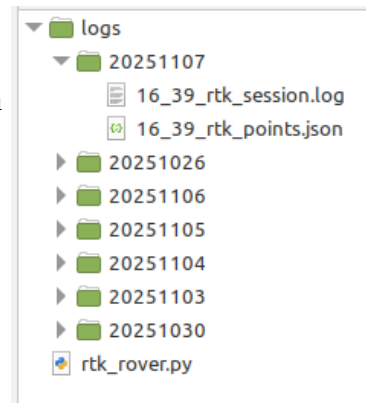
Der Inhalt der `/dev/shm/rtk_point.json` ist:

```
{"time": "16:37:22.259", "lat": 52.1267968, "lon": 8.663306716666666, "fix": "RTK-Fixed", "sats": "36", "speed": 0.0, "heading": 0.0}
```

Durch lat und lon sollen später die aktuelle Position während der Fahrt zum programmierten Ziel verglichen werden und ggf. Korrekturen eingeleitet werden heading und speed sind Kontrollwerte während der Fahrt zu den bestehenden Sensoren. Diese Funktion ist noch nicht implementiert / getestet.

Das zweite logFile steht im täglichen log Verzeichnis. Der Pfad muss später dann noch angepasst werden. In dem dem täglichen log Verzeichnis findet man dann für den rtk\_rover 2 logFiles. Für uns ist im Moment das <uhrzeit>\_rtk\_points.json wichtig. In diesem logFile stehen die aktuellen GPS Daten seit dem Start von rtk\_rover.py Und genau dieses logFile kann man mit dem Durchsuchen Button der Analyse.html Seite selektieren und laden.

Mit dem Slider unterhalb der Map kann nun durch die Datei navigiert werden. Mit dem Play Button rechts neben dem Slider kann läuft eine Art Wiedergabe des logFiles.



Wie schon weiter oben beschrieben, wird dieses Logfile später durch weitere Informationen ergänzt und auf der Map in der Html Seite dann angezeigt. Dieses ist für eine nachträgliche Analyse des Mähvorganges nützlich.

## RTK-Basis Station

mMuss noch erstellt werden

In dem nächsten Kapitel geht es dann um die RTK-Basis Station und das rtk-basis.py Script.