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.

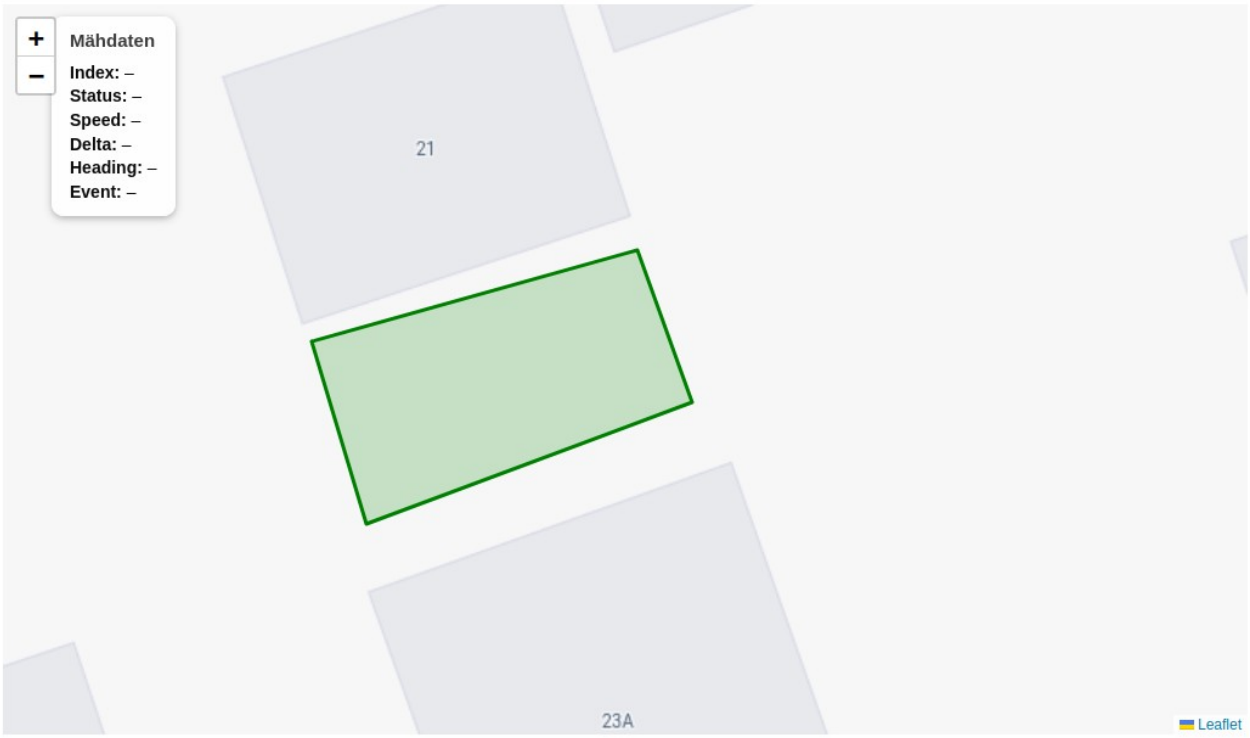
Anpassen der Html Vorlage

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 mi 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.



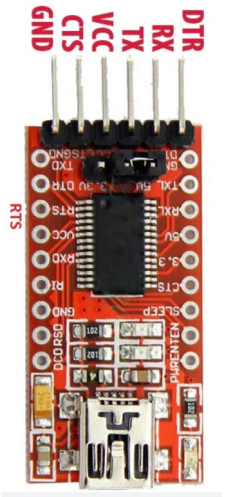
Die Bedienung der Seite ist intuitiv. Später ist gedacht, dass die mower.py ein logfile von dem Mähvorgang erstellt. Dieses logFile kann dann durch den Auswahlbutton selektiert und geladen werden. Jeder Eintrag in in dem logFile liefert einen Datenpunkt auf der Karte. Neben den Geodaten habe ich geplant 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"
```

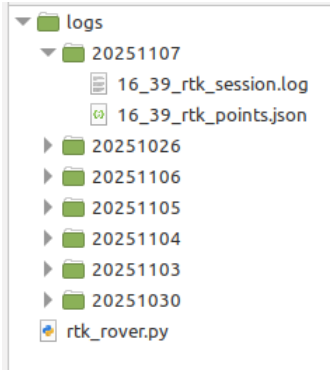
("VRS_3_3G_NW" ist der Mountpoint der GPS Glonass und Galileo Korrekturdaten mit einer Genauigleit von 1 – 2 cm bereitstellt.

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

```
########## Hole Korrecturdaten 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] $PQTMVERNO*58
[16:28:02.714] [TX] $PQTMCFGRCVRMODE,R*32
[16:28:02.915] [TX] $PQTMCFGRCVRMODE,W,2*29
[16:28:03.115] [TX] $PQTMCFGRCVRMODE,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 sekündlich die Geo Daten nach /dev/shm/rtk_point.json geschrieben. Diese sollen später sekündlich 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 täglichem 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

Derzeit läuft bei mir auf einem separatem Pi3 meine Basistation. Diese habe ich mit einem LC29H(BS) von Quectel und einer YB0017AA Antenne installiert und an einem festen Standort mit freier Sicht zum Himmel montiert. Mein Pi erkennt die Basistation an der Schnittstelle /dev/ttyUSB0

Zur Inbetriebnahme der Baisistation benötigst Du zwei python scripts. Als erstes muss die Basisstation wissen an welchem Punkt sie steht, damit sie diese als Referenzdaten zu dem Rover schicken kann. Wenn Antenne und Platine über das USB Kabel mit dem Pi verbunden ist, dann rufst Du mit „python LC29HBS.py“ auf.

Klappt alles, so gibt Dir das Modul folgendes aus, was auf dem Screenshot zu sehen ist.

```
pi@raspberrypi:~/dev/RTK-BASE $ python LC29HBS.py
[TX] $PQTMRESTOREPAR*13
[RX] $PQTMRESTOREPAR,OK*3B
[TX] $PQTMVERNO*58
[RX] $PQTMVERNO,LC29HBSNR11A01S,2023/02/13,10:14:06*27
[TX] $PQTMCFGSVIN,R*26
[RX] $PQTMCFGSVIN,OK,2,0,0.0,3879131.7429,591058.6174,5011570.3587*76
pi@raspberrypi:~/dev/RTK-BASE $
```

Der erste TX Befehl setzt das Modul auf Werksauslieferung zurück. Der zweite TX Befehl fragt den Firmwarestand ab und der dritte TX Befehl fragt ab, ob es sich das Modul im Survey-IN prozess befindet oder im normalen Betriebs Modus arbeitet. Hier ist es der normale Modus wie er für die Basisstation notwendig ist. Das erkennst Du an der 2 hinter dem OK in der letzten Zeile. Ich empfehle auch die Lektüre des Datenblattes. Den Link habe ich aus dem Waveshare Wiki zu den LC29H Pi Erweiterungen, auch sehr interessant, wenn man näheres Wissen möchte.

Nun hast Du 2 Möglichkeiten der Basisstation mitzuteilen wo sie sich befindet, damit sie die Korrekturdaten passend zu ihrer Position an den Rover senden kann.

Positionierung der Baisstation über Lat und Lon

Wenn Du die exakte Lat & Lon Werte und die Höhe Deiner Antenne kennst, so kannst Du Lat & Lon mit dem convert.py in ECEF Werte umwandeln. Editiere in der convert.py die Zeilen 22, 23 und 24 mit den Dir bekannten Werten. Du kannst diese u.U. mit Google-Maps oder anderen Karten-Tools bestimmen

```
17
18  ECEF_X = 3784034.4618
19  ECEF_Y = 899874.5653
20  ECEF_Z = 5037987.4823
21
22  LAT =  52.516181
23  LON =  13.376935
24  ALT =  34
25
```

Hast Du LAT LON und auch wichtig die Höhe der Antenne eingetragen, dann rufts Du python convert.py LAT auf und erhältst diese Ausgabe.

Diese Werte trägst Du dann in der rtk_basis.py in den Zeilen 67, 68 und 69 ein.

```
pi@raspberrypi:~/dev/RTK-BASE $ python convert.py LAT
Convert LAT LON 2 ECEF Values
ECEF_X: 3784034.4618
ECEF_Y: 899874.5653
ECEF_Z: 5037987.4823
```

Danach kannst Du mit python rtk_basis.py Deine Basisstation in Betrieb nehmen. In der Konsole werden Dir nun fortlaufend die RTCM Sätze angezeigt, welche die Basisstation zur Verfügung stellt.

```
Datei  Bearbeiten  Ansicht  Suchen  Terminal  Hilfe
[14:52:23.386] [RTCM3] Type 1092 | Size 207 bytes
[14:52:23.386] [RTCM3] Type 1098 | Size 28 bytes
[14:52:24.387] [RTCM3] Type 973  | Size 25 bytes
[14:52:24.387] [RTCM3] Type 1026 | Size 180 bytes
[14:52:24.387] [RTCM3] Type 1036 | Size 103 bytes
[14:52:24.387] [RTCM3] Type 1094 | Size 138 bytes
```

Rufts Du nun in Deinem Mäher die rtk_rover.py auf, und beide Rechner liegen im selben Netzwerk, so wirst Du eine Info in der Konsole der Basisstation sehen, dass sich ein Client verbunden hat. Nach kurzer Zeit, wird sich in der Konsole vom rtk_rover ein RTK-Float einstellen, und je nach Netzwerkverbindung und Sichtverhältnissen nach kurzer Zeit ein RTK-Fix. Sollte es einmal mehr als 5 Minuten dauern, nicht die Geduld verlieren. Er wird sich schon einstellen.

Positionierung der Baisstation über Survey-In Prozess.

Eine andere Möglichkeit die Basisstation auf den exakten GeoPosition einzustellen kannst Du mit dem Survey-In Prozess realisieren. Wichtig ist, dass Du die Antenne für die Basisstation schon fixiert hast und sie vielleicht auch schon an dem finalen Standort montiert hast. Mit dem python script LC29HBS.py 300 s rufts Du einen Survey-In Prozess von 300 Minuten Dauer, also von 5 Std. auf. Je länger der Prozess läuft, umso exakter hat das Modul selbst am Schluss des Prozesses seine Position errechnet.
Ist je nach eingegebener Zeitspanne der Prozess beendet, denn hat das Scipt im Logfile die ECEF_X, ECEF_Y und ECEF_Z Werte ermittel, die Du dann mit Copy& Paste wieder in die rtk_basis.py in die Zeilen 67 – 69 einfügen kannst.

```
168 [2025-11-12 09:42:58.572] [INFO] === Beende Survey-In nach 14 Std. 0 Min. 10 Sek. ===
169 [2025-11-12 09:42:58.572] [TX] $PQTMCFGSVIN,R*26
170 [2025-11-12 09:42:58.577] [PAIR] $PAIR012*39
171 [2025-11-12 09:42:58.628] [RX] $PQTMCFGSVIN,OK,1,50400,15.0,3879131.7429,591058.6174,5011570.3587*40
172 # Survey Dauer: 50400 Sekunden
173 # 3D_AccuracyLimit: 15.0 m
174
175 ECEF_X = 3879131.7429
176 ECEF_Y =  591058.6174
177 ECEF_Z = 5011570.3587
178
179 Latitude  = 52.12681757
180 Longitude = 8.66345450
181 Altitude  = 116.35
182
183 "lat": 52.12681757, "lon": 8.66345450
184
185 [2025-11-12 09:42:58.628] [INFO] Survey Daten sollen im LC29HBS gespeichert werden!
```

Wichtig ist das s in dem script Aufruf, Es bewirkt, dass am Ende die Daten im Flash des LC29HBS gespeichert werden.

Ich habe nach beiden Methoden meine Basistation eingerichtet und kann sagen, dass der Survey-In Prozess genauere Daten ermittelt.

Viel Erfolg bei der Inbetriebnahme Deiner GPS-RTK Lösung.