# Kontaktdaten

Samuel Putz: [samuel.putz@gmail.com](mailto:samuel.putz@gmail.com) / 0680 2254788

Simon König: [koenigsimon05@gmail.com](mailto:koenigsimon05@gmail.com) / 0664 5567837

# Github

Alle Projektrelevanten Dateien wurden in einem GitHub Repository zur Verfügung gestellt:

<https://github.com/Decretal/C-Track_2.0>

# Server

Für den Server wurde die Programmiersprach PHP mit dem Laravel Framework verwendet. Um den Server weiterzuentwickeln wird PHP und der Paketmanager Composer benötigt. Nach der Installation von PHP muss in der Konfigurationsdatei php.ini bei den extensions das Semikolon bei den Zeilen mit den Werten curl, fileinfo mbstring, pdo\_sqlite, pdo\_mysql und openssl entfernt werden. Danach kann Composer mit der Setup-Datei installiert werden. Um alle Bibliotheken zu installieren, muss im Ordner mit dem Servercode der Befehlt *composer install* ausgeführt werden.

Einstellungen werden in einer .env Datei im Serverordner festgelegt. Diese Datei muss erstellt werden. Damit das Projekt funktioniert muss dort eine Datenbank konfiguriert werden. Für den Entwicklungsprozess kann mit der Zeile DB\_CONNECTION=sqlite sqlite als Datenbank eingestellt werden. Dadurch wird kein SQL-Server benötigt. Vor dem Start des Servers müssen die benötigten Tabellen mit dem Befehl *php artisan migrate* erstellt werden.

Mit dem Befehl *php artisan serve --host 0.0.0.0 --port 8000* kann der Server gestartet werden. Der Server ist dadurch im lokalen Netzwerk zugänglich.

Um die App nutzen zu können muss ein Nutzer und ein Tracker erstellt werden. In der App ist das derzeit noch nicht möglich. Um den Nutzer zu erstellen kann folgender Befehl in der Windows Powershell genutzt werden:

iwr -Method Post -Body @{email="test@mail.com"; password="asdfsdg)9ouwgn+"; name="Simon König"} http://localhost:8000/api/account/register

Zurückgegeben wird ein Token, mit welchem sich der Nutzer ausweisen kann.

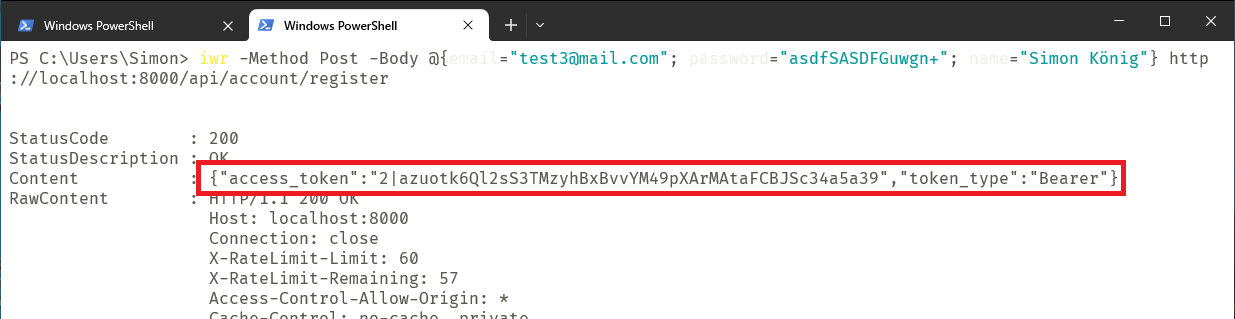


Abbildung 1: Erstellen eines Nutzers

Da die App die Features des Servers noch nicht vollständig in der App eingebaut wurden muss der Token im Code der App hinterlegt werden.

Nachdem der Nutzer erstellt wurde, muss der Tracker noch angelegt werden. Das kann mit folgendem Befehl gemacht werden:

iwr -Method Post -Headers @{Authorization="Bearer 1|wwGyHyxhlLxlRHhAuIIGDmGqjZ6saHcQdzjYF9cif7715c39"} http://localhost:8000/api/trackers -Body @{name="Kuh1"}

Der Token im Header muss durch den des gerade erstellten Nutzers ersetzt werden. Ist das Erstellen erfolgreich, bekommt man eine ID zurück. Im code der Sendeeinheit ist eine ID mit #define T\_ID 1 definiert worden. Diese muss durch die zurückgegebene ID ersetzt werden.

Sollte der Server auch außerhalb des eigenen Netzwerks zugänglich sein muss dieser selbst auf einem PC gehostet und der Port für alle geöffnet werden, oder man mietet einen Server. Für die Vorführung wurde der Server bei dem Anbieter „Heroku“ gehostet. Verwendet wurde folgende Anleitung: https://dev.to/mr\_steelze/deploy-a-laravel-project-on-heroku-with-sql-database-3n08

# Sender

Die Platine muss mit einer Spannung von 1,5-5,5V versorgt werden. Dazu kann der USB-Port auf der Platine oder eine C-Batterie genutzt werden. Für den Betrieb muss die SD-Karte eingesteckt sein, das LoRa Modul ist optional. Wird LoRa verwendet muss der ISM-Dipol an das Adafruit Breakoutboard angeschlossen werden.

Ein Bild, das Lampe, Design, Kunst enthält.

Automatisch generierte Beschreibung mit mittlerer Zuverlässigkeit

Abbildung 2: ISM-Dipol des Hersteller Molex

Nach einschalten der Platine beginnt automatisch die Standortermittlung und die Übertragung mit LoRa. Für die Standortübertragung muss die ID mit #define T\_ID definiert werden.

Für die Programmierung wurde Microchip Studio verwendet. Um den Code auf der Platine hochzuladen, wird ein ISP-Programmer benötigt. Für die Entwicklung wurde ein STK-500 kompatibler Programmer der Firma myAvr verwendet.

Um den Programmer einzurichten, muss dieser in Microchip studio unter tools -> device programming ausgewählt werden. Um den Programmer zu nutzen müssen die Einstellungen mit Apply bestätig werden. Microchip studio sollte nun damit beginnen, die Clock-Frequenz und die Target-Spannung auszulesen. Danach kann der Code mit F5 hochgeladen werden. Wichtig ist, dass die SD-Karte während des Upload-Prozesses nicht angeschlossen ist. Der Upload mittels ISP nutzt dieselbe SPI-Schnittstelle wie die SD-Karte. Dadurch wird diese während des Uploads fälschlicherweise angesprochen und sendet unbrauchbare Daten in den Speicher des Microcontrollers.

Ein Bild, das Text, Screenshot, Schrift, Software enthält.

Automatisch generierte BeschreibungEin Bild, das Text, Screenshot, Display, Software enthält.

Automatisch generierte BeschreibungAbbildung 4: Microchip Studio Device Programming

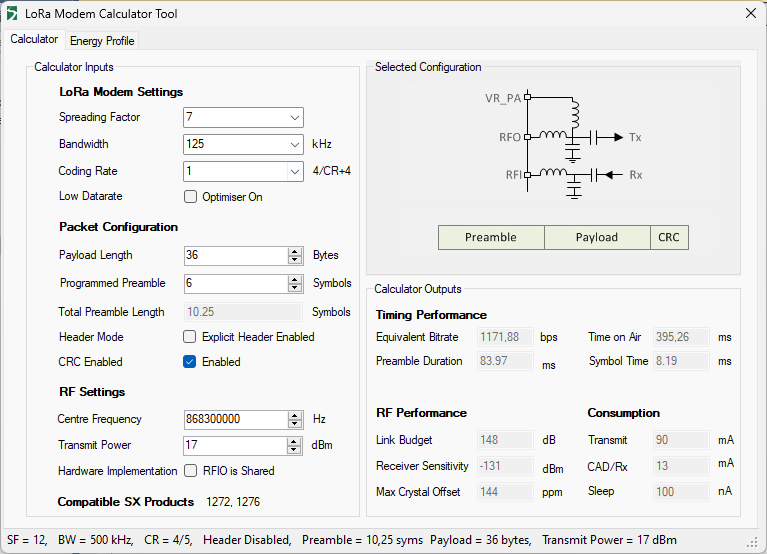
Abbildung 3: Microchip Studio ISP-Einstellungen

Um das Sendeintervall zu verändern, muss die gewünschte Zeit in der Zeile

uart\_sendStr("$PSRF103,4,0,20,1\*12\r\n");

angepasst werden. Das Zeitintervall ist das vorletzte Feld (in diesem Fall 20). Die Zeit wird in Sekunden angegeben. Das Zeitintervall kann zwischen 0 und 255 Sekunden lang sein. Die letzten 2 Zeichen vor dem \r\n sind eine Prüfsumme. Diese muss nach verändern der Zeit neu berechnet werden. Diese kann mit der Website <https://nmeachecksum.eqth.net/> oder durch xor-verknüpfung aller Ascii Werte zwischen $ und \* berechnet werden. Wichtig ist, dass der gesetzliche Duty Cycle nicht überschritten wird. Dazu muss die Dauer der Übertragung mit dem LoRa Calculator berechnet werden.

<https://www.semtech.com/products/wireless-rf/lora-connect/sx1276#_tools_software>

****

Die Payload length muss auf 36 Bytes gesetzt werden. In den Modem Settings müssen die Einstellungen für die Übertragung eingesetzt werden. Unter der Timing-Performance sieht man bei der time on air die Zeit, die eine Übertragung benötigt. Alle Übertragungsvorgänge dürfen nicht länger als 1% einer Stunde (=36s) dauern.

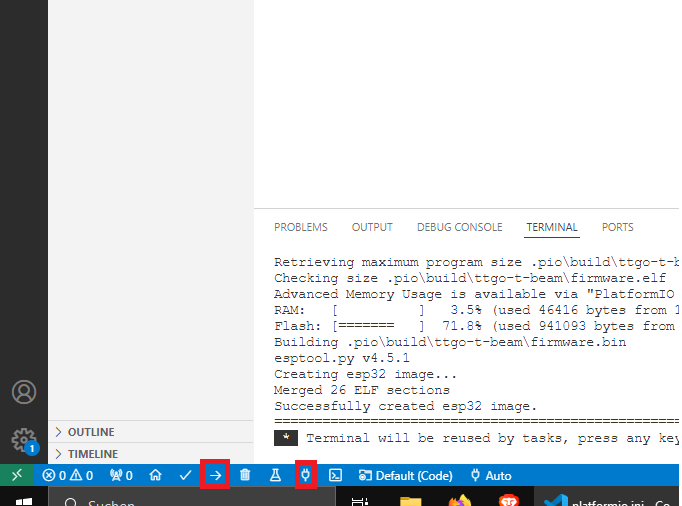
# Basisstation

Bei der Basisstation, dem TTGO T-Beam, muss die LPRS-Antenne mit dem SMA zu N Kabel angeschlossen werden. Versorgt wird die Platine mit einem Micro-USB Kabel.

Der Code für die Basisstation muss in Visual Studio Code mit PlatformIO geöffnet werden. Dazu muss das PlatformIO Plugin in Visual Studio Code installiert werden [https://marketplace.visualstudio.com/items?itemName=platformio.platformio-die]

Danach kann das Projekt im PlatformIO Menü geöffnet werden.

PlatformIO sollte damit beginnen, die benötigten Compiler und Bibliotheken herunterzuladen. Nachdem herunterladen kann mit dem Pfeil in der Leiste unten im Fenster das Projekt auf die Platine hochgeladen werden. Mit dem Steckersymbol kann der Serielle Monitor geöffnet werden.



Vor dem Hochladen müssen die Details für die WLAN-Verbindung, sowie die Adresse des Servers angepasst werden.



Abbildung 5: WLAN-Verbindung Basisstation



Abbildung 6: Adresse des Servers

Nach hochladen des Codes sollte sich das Board mit dem WLAN-Netzwerk verbinden und auf Daten vom Sender warten. Die vom Sender empfangenen Daten werden auf den Server geschrieben.

# App

Zuerst muss die Flutter Umgebung eingerichtet werden (wir empfählen über VS-Code)

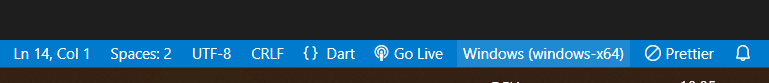
<https://docs.flutter.dev/get-started/install/windows/mobile?tab=first-start>

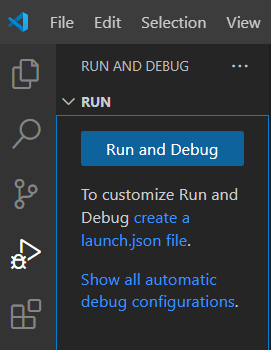
Ein Bild, das Text, Screenshot, Schrift, Software enthält.

Automatisch generierte BeschreibungWurde Flutter erfolgreich eingerichtet muss nun der Ordner Handy App aus dem GitHub Repository heruntergeladen werden und in VS-Code ausgewählt werden

Als nächstes wird unter ctrack/lib die main Datei ausgewählt.

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

Automatisch generierte BeschreibungNun muss ausgewählt werden worüber die App laufen soll:

Hier kann ausgewählt werden, ob die App auf einen Handy Emulator, über Chrome oder auf ein Verbundenes Android Handy geladen werden soll.

Wurde die gewünschte Umgebung ausgewählt kann die App über die Run and Debug Funktion hochgeladen werden.

Wichtig: Im Code muss die richtige Server Url und der richtige Token ausgewählt werden.