Inhalt

[03.10.2024 – Protokoll 3](#_Toc189752737)

[Motorsteuerung mit Webserver 3](#_Toc189752738)

[09.01.2025 - Protokoll 12](#_Toc189752739)

[Motoransteuerung über API-Requests 12](#_Toc189752740)

[Ziel 12](#_Toc189752741)

[Login mit Postman 12](#_Toc189752742)

[API-Calls mit Python 12](#_Toc189752743)

[Login-Funktion in Python 12](#_Toc189752744)

[Funktionen durch Browse entdecken 13](#_Toc189752745)

[Motor starten 15](#_Toc189752746)

[Motor stoppen 16](#_Toc189752747)

[Erste Applikation 16](#_Toc189752748)

[06.02.2025 Protokoll 17](#_Toc189752749)

[Daten von API auslesen und in Datenbank schreiben 17](#_Toc189752750)

[Ziel 17](#_Toc189752751)

[DB-Handler 17](#_Toc189752752)

[Daten auslesen 18](#_Toc189752753)

[Lesen und Schreiben 18](#_Toc189752754)

[Abbildungsverzeichnis 19](#_Toc189752755)

# 03.10.2024 – Protokoll

## Motorsteuerung mit Webserver

Aufbau:

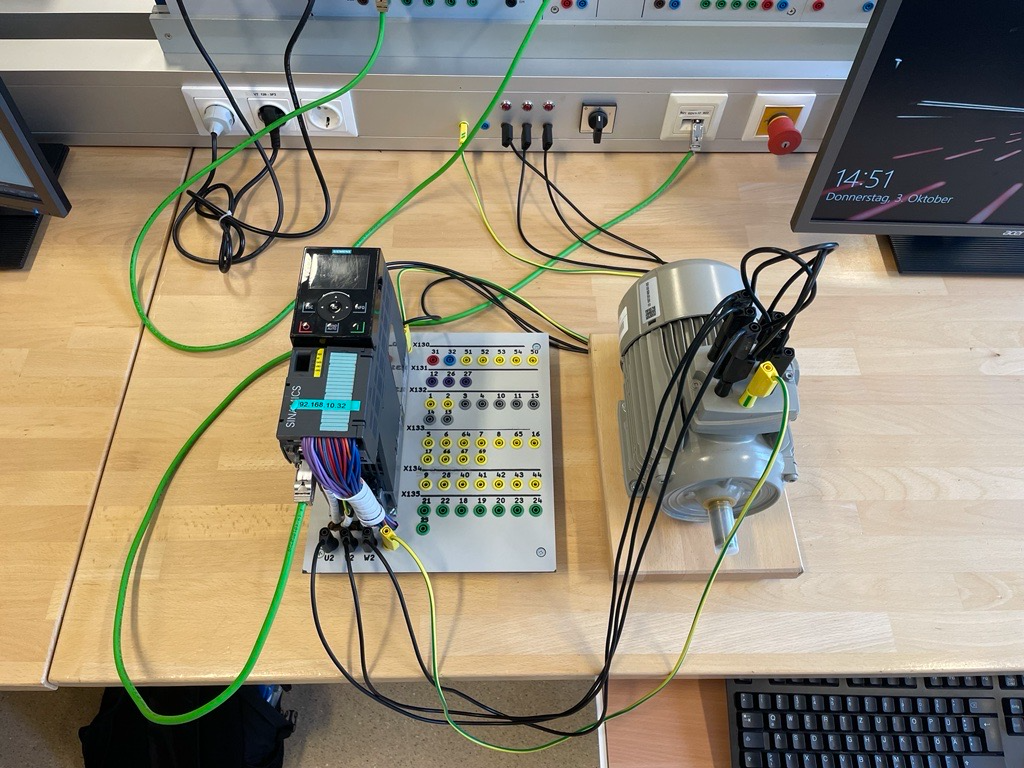


Figure 1 Aufbau

Zuerst Namen der CPU und des FU’s (Frequenzumrichter) ändern 🡪 aufgrund der IPs die auf den Geräten stehen.

IP von FU eingeben und Subnetz auf 255.255.254.0 stellen

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

Automatisch generierte Beschreibung

Figure 2 IP Eingabe & Subnetz

Danach auf „Online Verbinden“ klicken und hier die richtigen Schnittstellen einstellen

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

Automatisch generierte Beschreibung

Figure 3 verbinden

* Dann auf „Suche starten“ klicken, danach auf „Verbinden“
* Soll die IP-Adresse hinzugefügt werden?: JA
* Danach Reiter „Inbetriebnahme auswählen und durchklicken bis zu den Motordaten
* Motordaten anhand von der Plakette auf dem Motor eingeben und weiter durchklicken
* Motor mit Steuertafel in Betrieb nehmen

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

Automatisch generierte Beschreibung

Figure 4 Motor in Betrieb

* FU mit CPUverbinden
* Rechtsklick auf FU und Gerätenamen zuweisen
* Liste aktualisieren, dann Gerät auswählen und Namen zuweisen
* Gerätekonfiguration der CPU öffnen

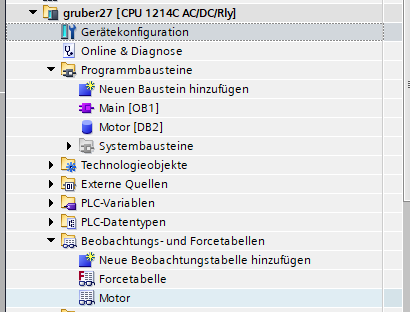


Figure 5 Gerätekonfiguration - Motor

* Neue Beobachtungstabelle „Motor“ erstellen
* Programmbausteine Motor[DB2] auswählen

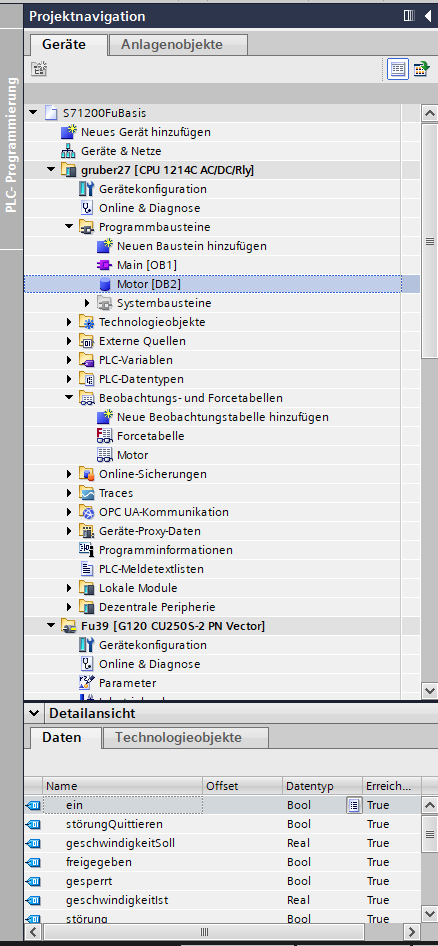


Figure 6 Programmbaustein - Motor

* Felder „**ein**“, „**störungQuittieren**“, „**geschwindigkeitSoll**“ und „**geschwindigkeitIst**“ in die Beobachtungstabelle reinziehen

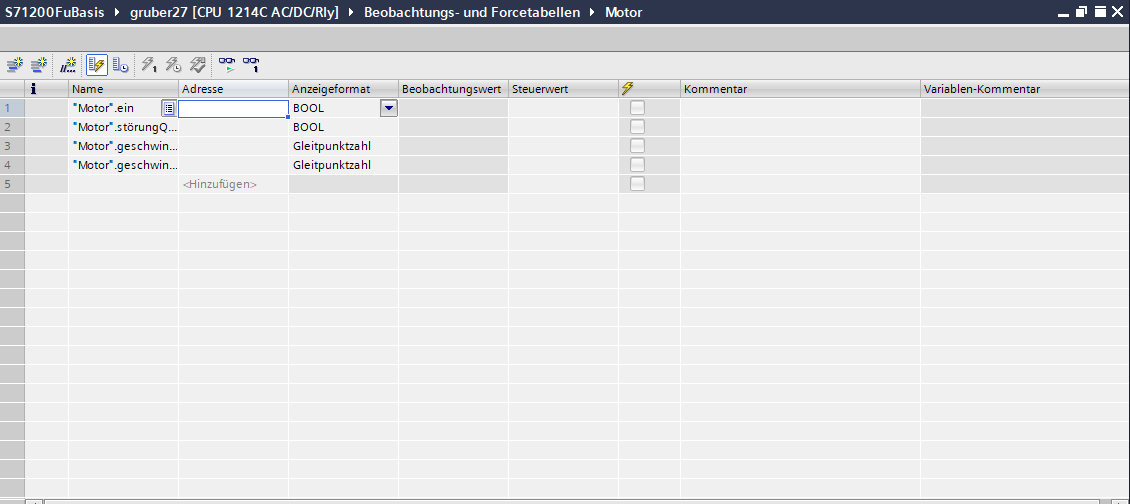


Figure 7 Beobachtungstabelle

* Über „Gerätekonfiguration“ 🡪 „Eigenschaften“ 🡪 „Webserver“ 🡪 „Allgemein“ den Webserver auf allen Modulen dieses Geräts aktivieren

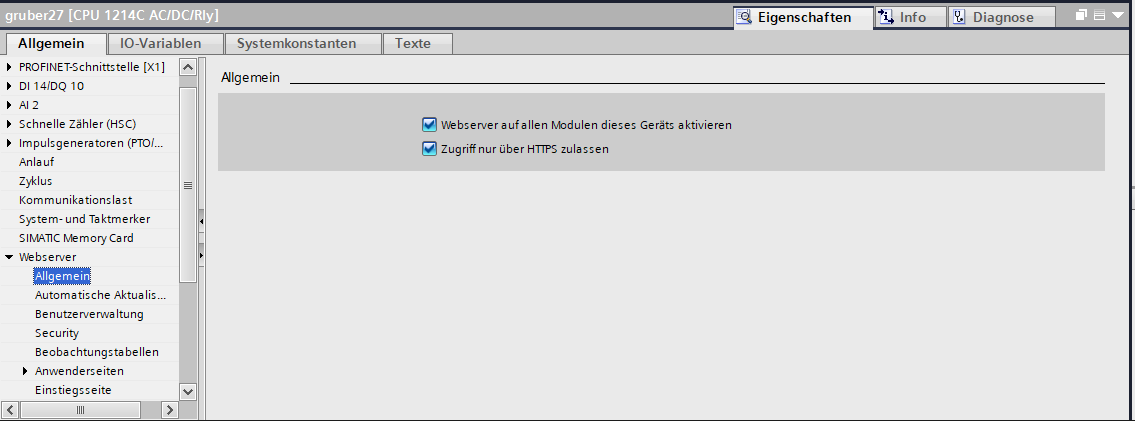


Figure 8 Webserver aktivieren

* In der Benutzerverwaltung einen neuen Benutzer mit administrativen Rechten erstellen

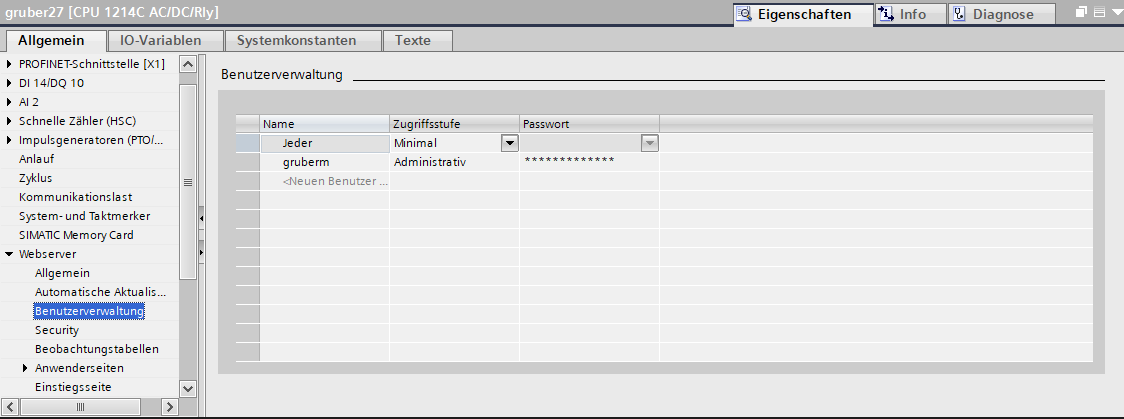


Figure 9 Benutzer erstellen

* Ein Bild, das Text, Software, Computersymbol, Webseite enthält.

  Automatisch generierte BeschreibungBei „Security“ den Zertifikatstyp auf „Hardwaregeneriert“ umstellen

Figure 10 Zertifikat

* Beobachtungstabelle „Motor“ unter „Beobachtungstabellen“ mit Lese/Schreib-Rechten hinzufügen

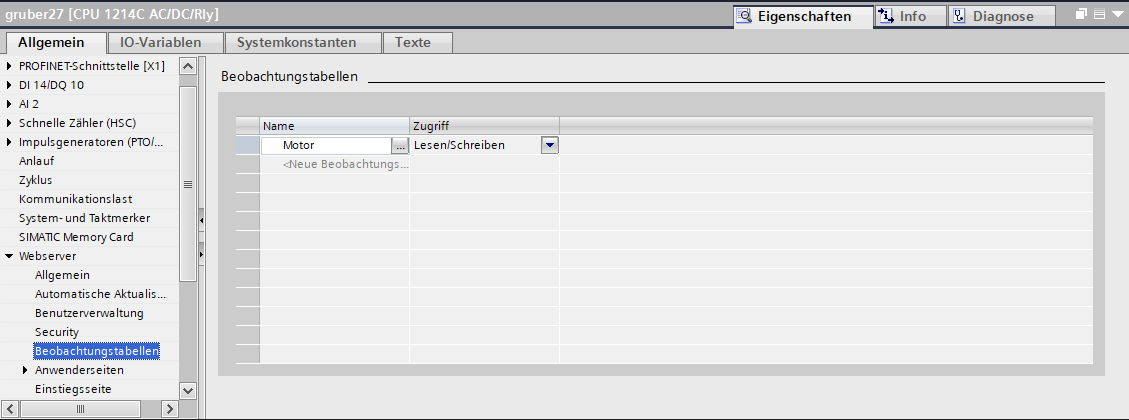


Figure 11 Lese- Schreibrechte

* Unter der IP-Adresse den Webserver aufrufen

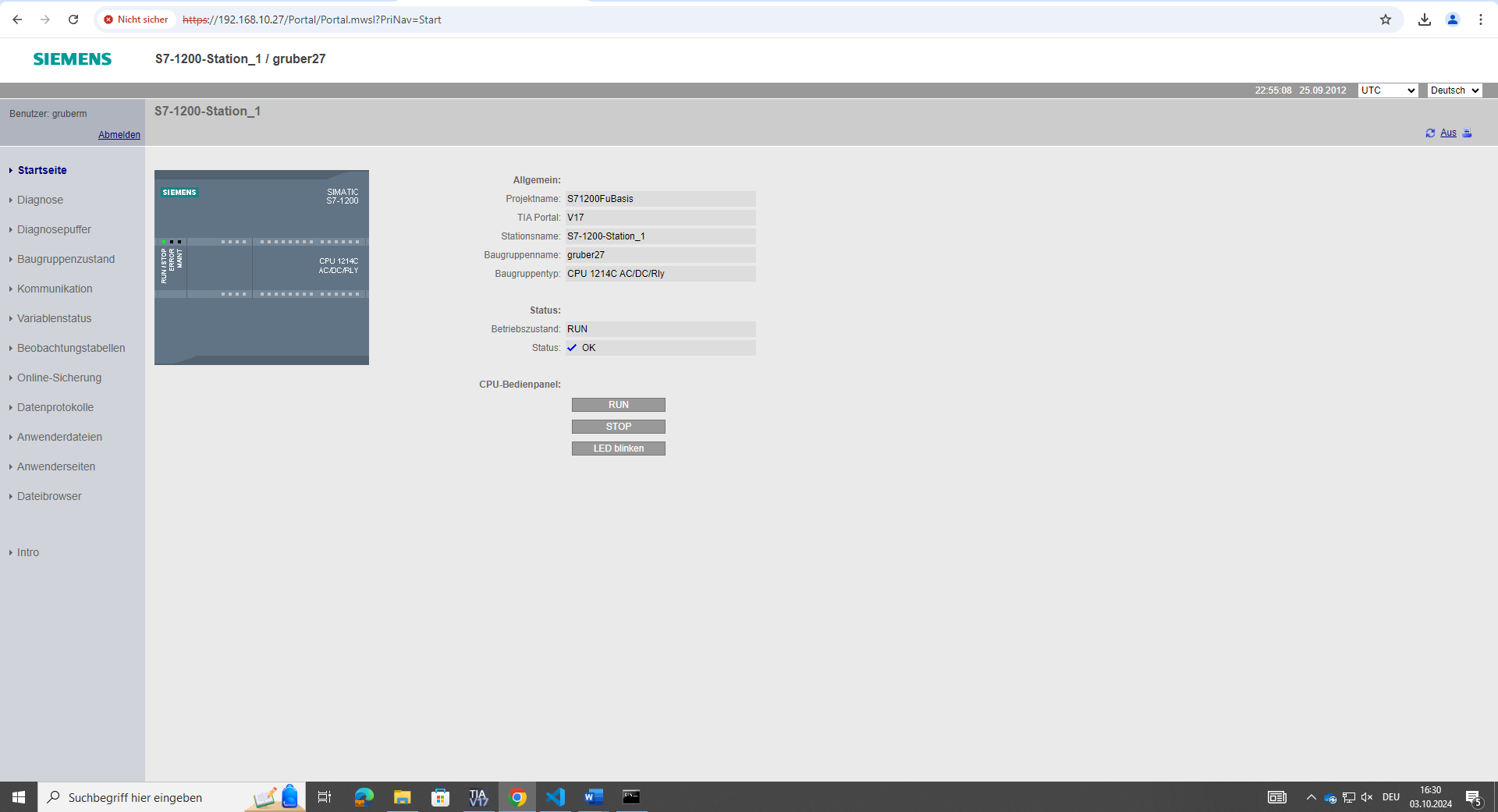


Figure 12 Webserver aufrufen

* Unter „Systembausteine/Webserver“ „DB333“ auswählen
* Rechts unter „Kommunikation/Webserver“ das Modul „WWW“ reinziehen
* RET\_VAL als retval setzen und die Variable mit Rechtsklick definieren

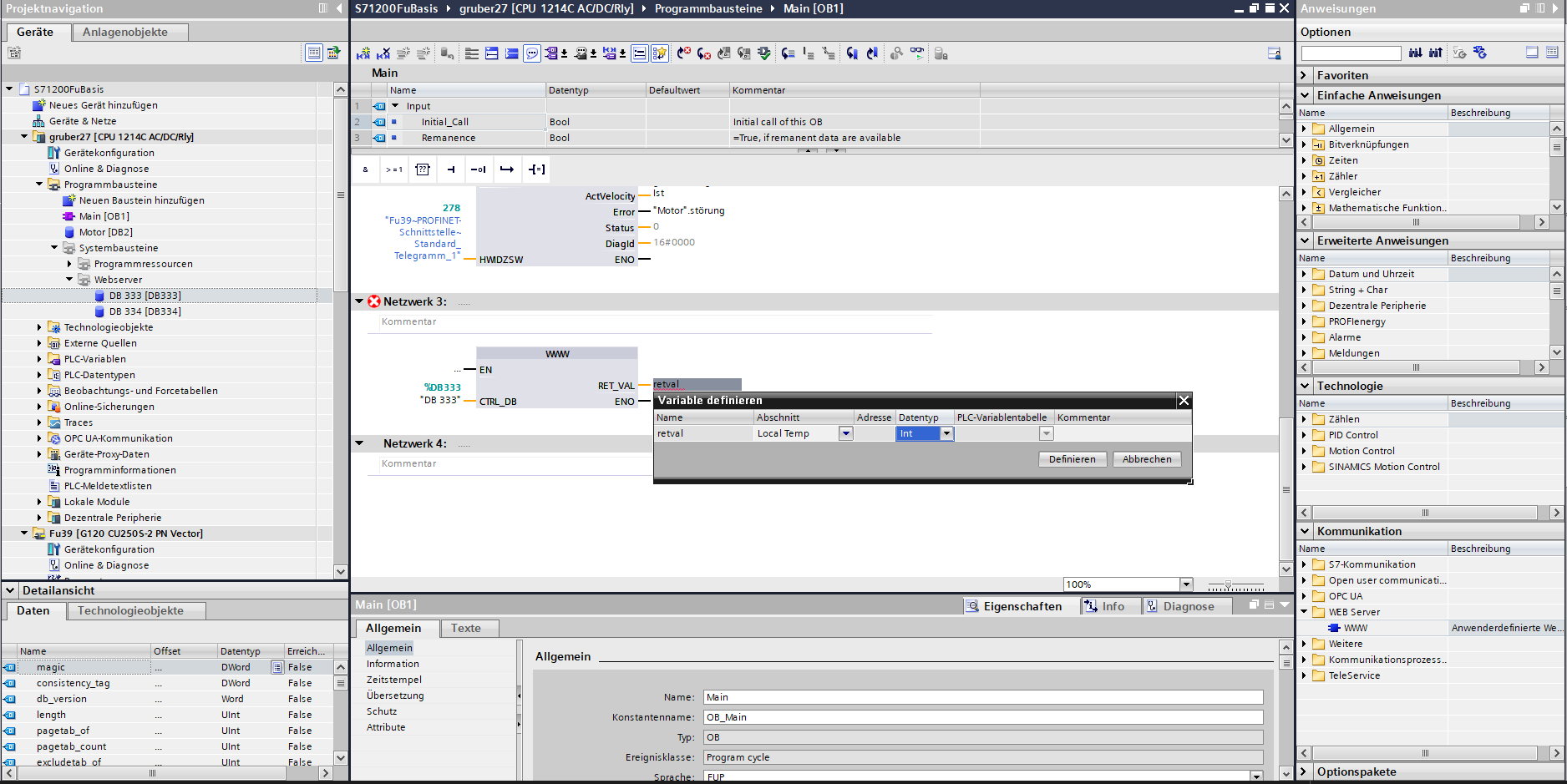


Figure 13 Variablen definieren

* Im Project ornder unter UserFiles den Ordner „Web“ erstellen und die Startseite die wir vom Fachlehrer bekommen haben reinkopieren

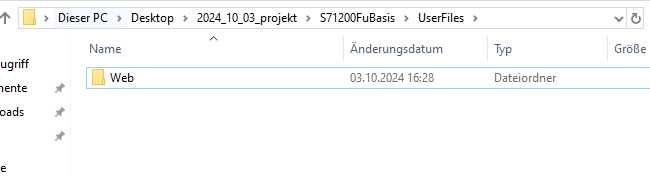


Figure 14 Startseite erstellen

* Unter „Webserver/Anwenderseiten“ das HTML Verzeichnis und den Applikationsnamen definieren

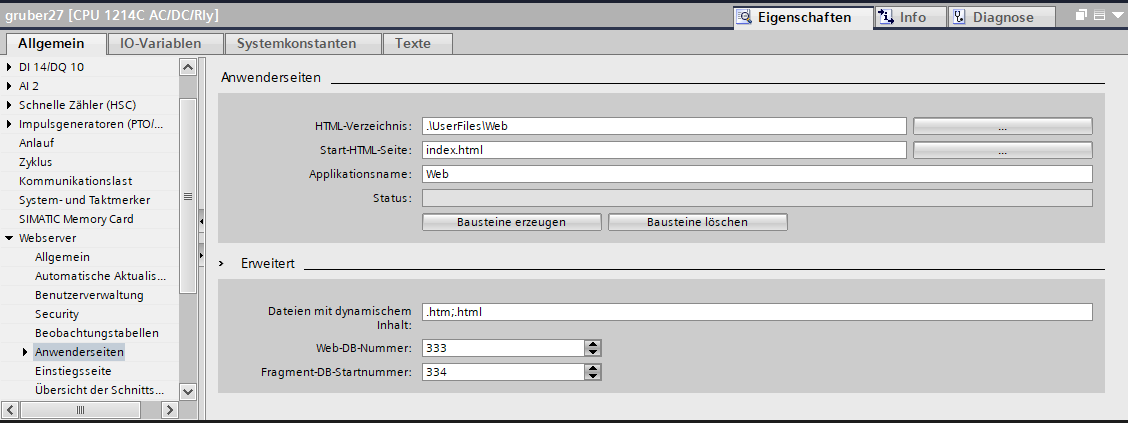


Figure 15 Verzeichnis und Namen definieren

* Folgende Ausgabe erscheint unter Allgemein wenn alles funktioniert hat

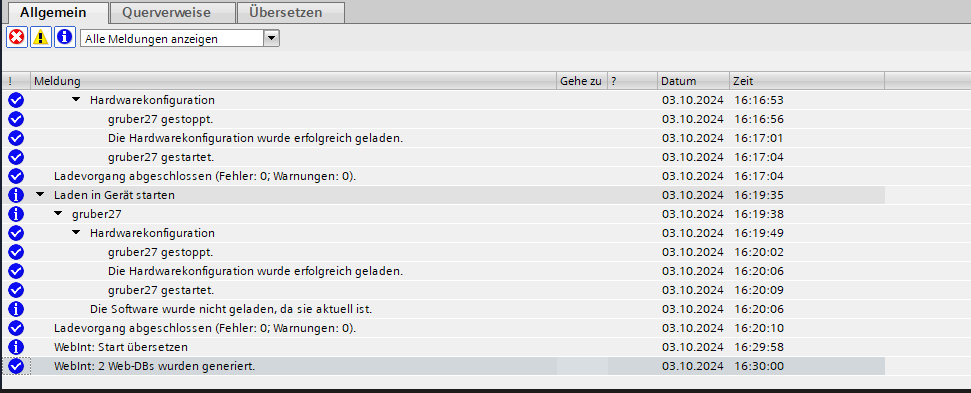


Figure 16 Output

* Auf der vorher generierten Seite unter Anwenderseiten kann man nun auf die erstellte Website zugreifen und den Motor über ebendiese steuern.



Figure 17 Webserver

# 09.01.2025 - Protokoll

## Motoransteuerung über API-Requests

### Ziel

Das Ziel dieser Übung ist es, einen Motor über API-Requests zu steuern. Dies wird zunächst mit Postman getestet, um die grundlegenden API-Funktionen zu verstehen, und anschließend in einem Python-Skript umgesetzt.

### Login mit Postman

* In Postman wird ein POST-Request an die /jsonrpc-Route gesendet.
* Das JSON-Body enthält folgende Struktur:

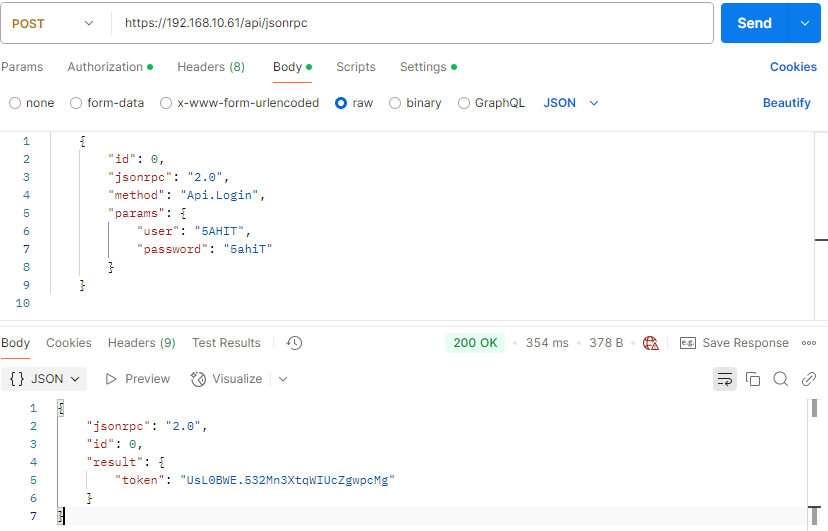


Figure 18 JSON-Body in Postman

* Als Antwort wird ein Token zurückgegeben, das für alle weiteren API-Requests erforderlich ist.

### API-Calls mit Python

Um die API einfacher zu verwenden, wird der Login in einer Funktion gekapselt.

### Login-Funktion in Python

Die folgende Funktion sendet den Login-Request und gibt den Token zurück:



Figure 19 Login API-Call Python

### Funktionen durch Browse entdecken

Um verfügbare Funktionen für den Motor herauszufinden, kann die Browse-Route verwendet werden.

Diese Route wird durch folgenden Code aufgerufen. In diesem Fall wird geschaut, welche Funktionen man für die Komponente „Motor“ ausführen kann.

Der vorher erhaltene Token muss angegeben werden.

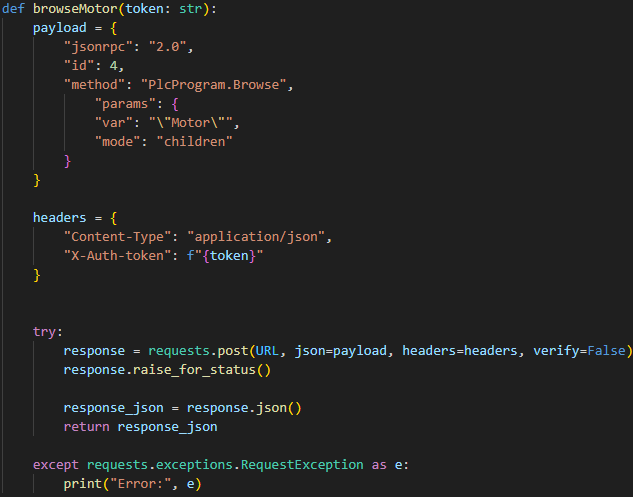


Figure 20 Browse Python

Folgende Ausgabe wird erzeugt, wenn man den Return-Wert dieser Funktion ausgibt.



Figure 21 Browse Output

### Motor starten

Durch das vorherige Browsen wurde festgestellt, dass der Motor ein „.ein“ Attribut hat, welches den Motor startet.

Setzt man dieses Attribut hier auf True wird der Motor beim Aufrufen dieser Route gestartet.

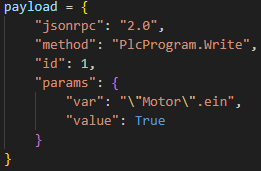


Figure 22 JSON für Motor starten

Folgendes Bild zeigt die ganze Funktion, die den Motor startet.

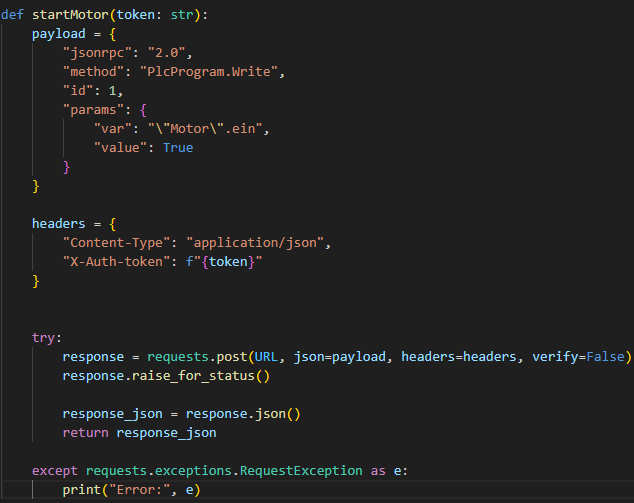


Figure 23 Motor starten Funktion

### Motor stoppen

Auf ähnlichem Weg kann der Motor auch wieder gestoppt werden.

Das Motor.ein Attribut muss auf False gesetzt werden.

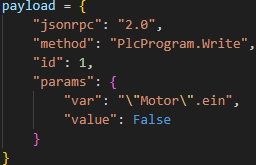


Figure 24 JSON für Motor stoppen

Der restliche Code ist gleich wie beim Starten des Motors.

### Erste Applikation

Durch folgenden Code wird der Motor für 5 Sekunden eingeschaltet und dann wieder aus.

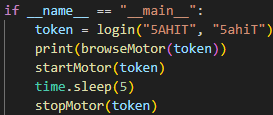


Figure 25 Main.py

# 06.02.2025 Protokoll

## Daten von API auslesen und in Datenbank schreiben

### Ziel

Ziel dieser Übung ist es, Daten von der im vorigen Protokoll bereits beschriebenen API auszulesen und in eine bereitgestellte MySQL Datenbank zu schreiben.

### DB-Handler

Die DB-Handler Klasse dient dazu, sich mit der MySQL Datenbank zu verbinden und Aktionen mit ihr auszuführen. In diesem Fall implementiert diese Klasse eine „insert\_data“ Methode, mit der Daten in die Datenbank geschrieben werden können.

Ein Bild, das Text, Screenshot, Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

Figure 26 DB Handler Klasse

Im Main wird diese Klasse dann initialisiert.

### Daten auslesen

Mit der readData Methoden werden Daten von der API gelesen, die dann eingefügt werden können.

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

KI-generierte Inhalte können fehlerhaft sein.

Figure 27 readData Funktion

### Lesen und Schreiben

Es wird ganz normal der Login ausgeführt, so wie beim letzten mal. Dann werden 100x die Daten für Dpublic.cosinus ausgelesen und direkt in die Datenbank geschrieben, bevor die Connection dann wieder geschlossen wird.

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Figure 28 main.py

# Abbildungsverzeichnis

[Figure 1 Aufbau 2](#_Toc189752667)

[Figure 2 IP Eingabe & Subnetz 2](#_Toc189752668)

[Figure 3 verbinden 3](#_Toc189752669)

[Figure 4 Motor in Betrieb 4](#_Toc189752670)

[Figure 5 Gerätekonfiguration - Motor 4](#_Toc189752671)

[Figure 6 Programmbaustein - Motor 5](#_Toc189752672)

[Figure 7 Beobachtungstabelle 6](#_Toc189752673)

[Figure 8 Webserver aktivieren 6](#_Toc189752674)

[Figure 9 Benutzer erstellen 6](#_Toc189752675)

[Figure 10 Zertifikat 7](#_Toc189752676)

[Figure 11 Lese- Schreibrechte 7](#_Toc189752677)

[Figure 12 Webserver aufrufen 7](#_Toc189752678)

[Figure 13 Variablen definieren 8](#_Toc189752679)

[Figure 14 Startseite erstellen 8](#_Toc189752680)

[Figure 15 Verzeichnis und Namen definieren 9](#_Toc189752681)

[Figure 16 Output 9](#_Toc189752682)

[Figure 17 Webserver 10](#_Toc189752683)

[Figure 18 JSON-Body in Postman 11](#_Toc189752684)

[Figure 19 Login API-Call Python 12](#_Toc189752685)

[Figure 20 Browse Python 13](#_Toc189752686)

[Figure 21 Browse Output 13](#_Toc189752687)

[Figure 22 JSON für Motor starten 14](#_Toc189752688)

[Figure 23 Motor starten Funktion 14](#_Toc189752689)

[Figure 24 JSON für Motor stoppen 15](#_Toc189752690)

[Figure 25 Main.py 15](#_Toc189752691)

[Figure 26 DB Handler Klasse 16](#_Toc189752692)

[Figure 27 readData Funktion 17](#_Toc189752693)

[Figure 28 main.py 17](#_Toc189752694)