

Anleitung : mit WICAN (Meatpi) den SOC vom Peugeot 208e bekommt
(funktioniert vielleicht auch mit OPEL Corsa E)

Description : getting the SOC on Peugeot 208e by using WICAN
(works maybe as well for OPEL Corsa E)

0) Meine IT – Umgebung / my IT environment

Auto / car : Peugeot 208e
Smarthome Appl: IO-Broker (on Raspberry Pi)
Wallbox : OpenWB (on Raspberry Pi)
CAN-BUS Adapter: WICAN-OBD-C3

1) Anwendungsfall (Ziel) / Use Case (objective)

Der SOC soll vom Peugeot 208e an IO-Broker / Wallbox gesendet werden sobald ich zuhause ankomme

Get the the SOC from Peugeot 208e when arriving at home (car still switched on) and send it from WICAN via Wifi to IO-Broker / Wallbox.

2) Anschluss des WICAN an OBD2-port / Connection WICAN to OBD2-port

Ich benutze ein Adapterkabel, bei dem ich lediglich Power V+ (PIN-16 am WICAN) mit dem Zündungs + (PIN-1 am 208e) verbunden habe, um sicher zu gehen, dass der WICAN bei ausgeschaltetem Auto auch aus ist.

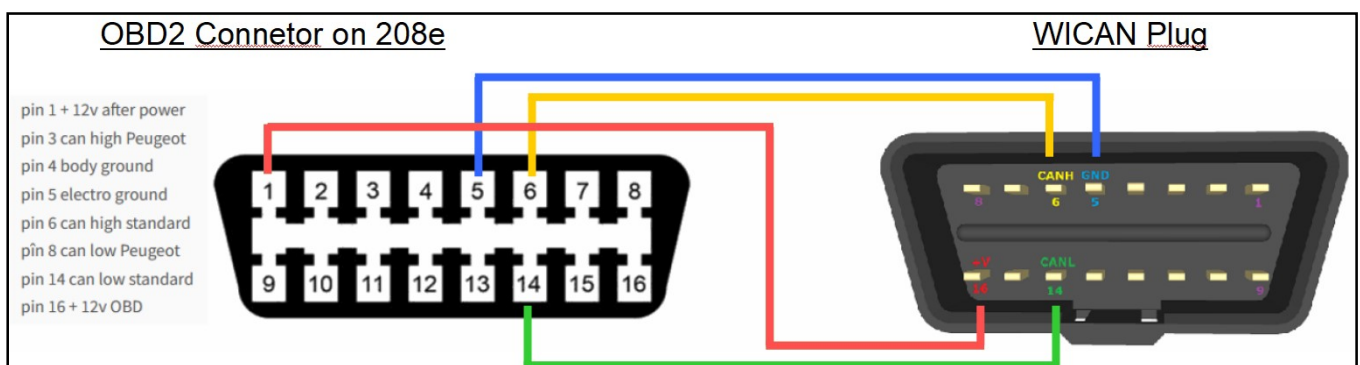
=> kein weiterer Stromverbrauch

Weiter PINs-Verbindungen (grün / gelb) für die Kommunikation muessen seit einem Software-Update seitens Peugeot nicht mehr verändert werden.

I'm using an adapter-cable and I only changed the Power V+ (PIN-16 on WICAN) to Ignition + (PIN-1 on 208e) to ensure that the WICAN is switched off when the car is off
=> no additional and possible power-consumption.

The 208e even does not send nay SOC-information during charging.

Changing other PINs for communication purpose (green / yellow) is not needed anymore since a software-update some years ago.



3) Verbinde den WICAN mit deinem WLAN / Connect WICAN to your Wifi

Ich denke ich brauche diese Kapitel nicht detailliert auszuführen

I think, that I don't need to detail this chapter

3) MQTT-Server installieren und einrichten (IO Broker)

The screenshot shows the 'Instanzeinstellungen: mqtt.3' configuration page in the IOBroker interface. The left sidebar contains navigation links: Übersicht, Adapter, Instanzen (selected), Objekte, Aufzählungen, Protokolle, Benutzer, Hosts, and Dateien. The main content area is divided into three tabs: VERBINDUNG, SERVER SETTINGS, and MQTT EINSTELLUNGEN. The 'VERBINDUNG' tab is active, showing fields for 'IP-Adresse' (set to '[IPv4] 0.0.0.0 - Zugriff von allen IPs zulassen') and 'Port' (set to '1885'). There is a checkbox for 'WebSockets benutzen'. Below this is the 'Authentifizierungseinstellungen' section with fields for 'Benutzer' and 'Passwort' (masked with dots), and a 'Passwort (wiederholen)' field.

4) WICAN Konfiguration / configuration of WICAN

Settings:

The screenshot shows the 'Settings' tab of the WICAN configuration interface. The interface has tabs for Status, Settings (active), Automate, Monitor, and About. The settings are organized into sections: AP Config, Station Config, CAN, MQTT, BLE, and Sleep Mode. Red annotations highlight specific settings: 'Protocol: AutoPID' in the CAN section is marked with '!!!'; 'MQTT URL: 192.168.1.67' is annotated with 'IP of IO-Broker' in red; 'MQTT Port: 1885' is annotated with 'port of MQTT-server (IO-Broker)' in red. At the bottom, a table lists CAN IDs with columns: CAN ID (dec), Name, PID, Index, Start Bit, Bit Length, Expression, and Cycle ms. The first row is highlighted with a red box and '!!!': CAN ID 1684, Name SOC, PID 212, Index 2, Start Bit 0, Bit Length 1, Expression B4/2, Cycle ms 500. A 'Delete' button is next to this row. Below the table is a note: 'Note: If you forget the AP password, connect usb cable to recover'. At the very bottom is a 'Submit Changes' button.

CAN ID (dec)	Name	PID	Index	Start Bit	Bit Length	Expression	Cycle ms
1684	SOC	212	2	0	1	B4/2	500

Automate:



Status **Settings** **Automate** **Monitor** **About**

Vehicle Specific: Disable ▾
Vehicle Model: ▾ ↻
Home Assistant Discovery: Disable ▾
Grouping: Disable ▾
Destination Topic:
Cycle Time(ms):
Vehicle Profiles: Durchsuchen... Keine Datei ausgewählt.
Store

Custom PIDs:
Custom Initialisation:

Name	Init	PID	Expression	Period(ms)	Type	Send_to	New
<input type="text" value="soc"/>	<input type="text" value="CSM1;ATCRA694;"/>	<input type="text" value="22D4101"/>	<input type="text" value="B4/2"/>	<input type="text" value="1000"/>	MQTT_Topic ▾	<input type="text" value="wican/208e/can/soc"/>	Delete
<input type="text" value="soh"/>	<input type="text" value="CSM1;ATCRA694;"/>	<input type="text" value="22D8601"/>	<input type="text" value="B4"/>	<input type="text" value="1000"/>	MQTT_Topic ▾	<input type="text" value="wican/208e/can/soh"/>	Delete

Submit Changes

Custom Initialization:
ATWS;ATi;ATE0;AT@1;ATSP6;ATAT1;ATL0;ATS0;ATH1;ATCAF1;ATSH79B;ATFCSH79B;ATFCSD300000;ATFCSM1;
Init: ATSH6B4;ATFCSM1;ATCRA694
Expression: B4/2

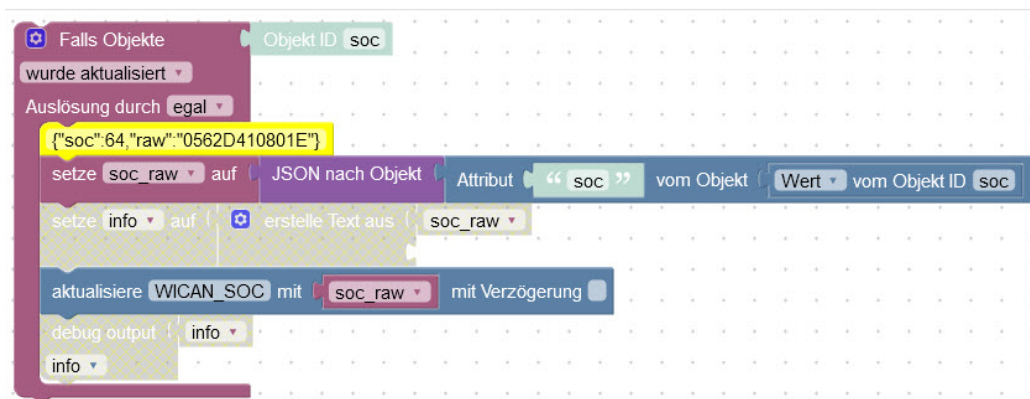
5) received SOC in IO Broker

📁 wican	wican	📁 folder		
📁 208e	208e	📁 folder		
📁 can	can	📁 folder		
📁 rx	wican/208e/can/rx	📁 state	variable	{ "SOC": 63.500000 }
📄 error	<small>matt server variable</small> wican/208e/can/rx/error	📁 state	variable	{ "error": "Timeout, pid: 22D8101" }
📄 soc	<small>Manuell erzeugt</small> soc	📁 state	state	JSON { "soc": 63.5, "raw": "0562D4107F9A" }
📄 soh	<small>matt server variable</small> wican/208e/can/soh	📁 state	variable	{ "soh": 1, "raw": "0662D8600105C5" }
📄 status	<small>matt server variable</small> wican/208e/can/status	📁 state	variable	{ "status": "offline" }
📄 tx	<small>matt server variable</small> wican/208e/can/tx	📁 state	variable	

6) Isolate SOC value via Blockly-Script

JSON auslesen, zum OBJECT konvertieren und den Wert des Attributs „soc“ auslesen und erneut isoliert abspeichern.

read the JSON , convert to an OBJECT and select only the value of the attribut „soc“ and store it as a single value



7) Der schwierigste Schritt ist geschafft / The most complicated step is ready

Der aktuelle Wert wird an IO-Broker übergeben, sobald der WICAN sich im WLAN angemeldet hat.

The current value is transmitted to IO-Broker as soon as the WICAN is linked to the WiFi.