Hallo Tan Vo,
Hallo Yves Charlie Yomeni,

wie besprochen schick ich euch paar Punkte die für die erfolgreiche Bearbeitung des Supplements sehr hilfreich sind

Für die Verbindung mit der Raspberry Pi:

Hier soll der Pi für den Austausch und der Automatische Start/Stopp zuständig sein. Ein wesentliche Aufgabe der Pi ist die Anzeige auf einem Display.
Dieser Display soll eine schöne Graphische Darstellung der Zustände darstellen

(1- Akku leer à wird als Akku gezeigt welche in der Ladezustand befindet)
(2- Akku fast voll à langsamer Änderung der Akku Ladezustand, das simuliert das langsame Laden nach der Erreichung eines gewissen Levels z.b 85% der Akku ist Voll)

Dann muss man dann der Pi so Programmieren dass er die Pakete von dem UWB Module über Serielle Schnittstelle liest und anhand dessen wird dann entsprechend gezeigt.

Eine detaillierte Tutorial zur GUI Programmierung auf dem Raspberry Pi findet ihr hier

* Raspberry Pi GUI Tutorial [link](https://www.baldengineer.com/raspberry-pi-gui-tutorial.html)

Notes:

1- EVSE: electric vehicle charging station, EV: electric vehicle
2- IEC61851: gilt für Einrichtungen zum Laden innerhalb und außerhalb von Elektrostraßenfahrzeugen an genormten Wechselspannungen, dieser Norm definiert der Signale beim E-Fahrzeug Laden (wir machen was ähnliches aber Wireless, deswegen nenne ich alles mit Bezug auf diesen Norm)
3- Mit diesen Teilen werdet ihr arbeiten:
    a. 2x http://www.exp-tech.de/raspberry-pi-3-modell-b-1gb-ram (für beide Seiten)
    b. 2x http://www.exp-tech.de/smartipi-touch-stand-for-raspberry-pi-7-touchscreen-display-no-lego-studs-on-front
    c. http://www.exp-tech.de/raspberry-pi-display-7-touch-screen-display-with-10-finger-capacitive-touch
    d. Die UWB Module gebe ich euch später dann kann ich euch zeigen wie man mit den Umgeht.
4- GUI Programmierung ist mit Qt creator unter Linux am besten und am einfachsten
5- Hilfreich ist es wenn ihr euch schon mit Git vertraut macht (Pull, Push, Commit)
    a. http://dev.firegate.de/projects/public/wiki/GIT_Doku
    b. Vlt gibt es was einfaches als startup im Internet
    c. Ich werde euch dabei noch helfen wenn es nötig ist

Für die UWBs:

EV (Auto) hat ein UWB Module konfiguriert als Tag ßà EVSE hat ein UWB Module als Anchor konfiguriert
Die Datei im Anhang dient als Referenz.

Workflow für die Programmierung der UWB Knoten:

* Installation von LPCXpresso

* Import des Workspace aus dem bereitgestellten Git Repository [fw-dwm1000](https://gitlab.com/m1r3l/fw-dwm1000)

Ich habe schon die beiden UWB Knoten konfiguriert und auch die msg-Frame verlängert um 4 neue States (Zustände) zu übertragen, auch mit den entsprechenden set und get commands
Die sind:

iss: cmd_iec61851_set_ev_state
isb: cmd_iec61851_set_ev_battery
isv: cmd_iec61851_set_evse_state
isp: cmd_iec61851_set_evse_power

im **src/cli/cli_tbl.h** und **src/cli/commands/cmd_iec61851.c**
src/drivers/rf/dwm1000/instance.c , **src/drivers/rf/dwm1000/instance.h**
zu finden

das ist schon ein großer Schritt in der richtigen Richtung.

Der Code ist vertraulich, und er darf nicht zu dritten gegeben werden!

Frohes schaffen!

Mit freundlichen Grüßen

Myrel Alsayegh, M.Sc.

Wissenschaftliche Mitarbeiterin

Bergische Universität Wuppertal
Fakultät für Elektrotechnik, Informationstechnik und Medientechnik
Campus Freudenberg
Rainer-Gruenter-Straße 21
Raum: FD 2.14
D-42119 Wuppertal
Telefon: +49 (0)202 439-1517