~~DISCUSSION:off~~

JSON-API

Übersicht über unterstützte Befehle:

- GPIO-Befehle, evtl. mit leicht geänderten Symbolen (? durch anderes Symbol ersetzt)

Designziele

- einfach zu parsen, v.a. sollte die Bedeutung gelesener Zeichen nicht von noch nicht gelesenen Zeichen abhängen
- Kompatibilität zu alten GPIO-Befehlen
  - Buchstaben a bis m dürfen nicht am Anfang eines Befehls stehen, sonst Widerspruch zum einfachen Parsen
  - Problem bei Erhöhung der Anzahl der IO-Pins: entweder alte GPIO-Syntax wird inkonsistent (z.B. Verwendung von Zahlen statt Buchstaben für neue Pins) oder es wird kompliziert zu Parsen (z.B. wenn ein Befehl der JSON-API mit n beginnt wäre nicht sofort klar, ob damit der Befehl oder der PIN mit der Bezeichnung n gemeint ist) → Lösungsvorschlag: Buchstaben n, o und p reservieren für eventuelle Erweiterung und damit ein hexadezimale Kodierung neuer Pins ermöglichen (z.b. Pin 15 entspricht p, Pin 16 entspricht aa)

Comment by svesch: Ist ok, 16 Bit ([a-p] + [x] als Joker) werden im Alphabet reserviert. Bei zukünftigen Implementierungen kann die Belegung ggf. dynamisch geändert werden oder auf - wie in Deinem Vorschlag - Doppelkombinationen erweitert werden.

- Kompatibilität zu Action-Script → kompatibel zu URL (Erlaubte Zeichen: [A-Z, a-z, 0-9, -, _, ., ~])
  - nicht case sensitive
  - möglichst keine Sonderzeichen
- Text basiert, möglichst ASCII

Vorschläge

- Befehle dürfen nur mit q bis w, y oder z beginnen
- hierarchischer Aufbau der Befehle: erster Buchstabe gibt Gruppe der Befehle an, weitere die Untergruppen bis schließlich ein Befehl ausgewählt wird (z.B. sb = 5000 heißt Baudrate des Serial Servers auf 5000 setzen; s → Befehl hinsichtlich des Serial Servers, b → Baudrate)

Partition (erster Buchstabe)	**Untergruppe**
s	Serial Server
q	Prozesszugriff
r	IP-Einstellungen

Partition (erster Buchstabe)	Untergruppe
t	Event trigger Einstellungen
...	...

Zwischenstand:

Command Name	Type	Description
q	submenu	Access to Process
qs	WORD	indicates if the process is running; setting to 1 starts the process; setting to 0 stops the process; 2 indicates an error and is no valid value to be written
y	submenu	pins; development only!
yX	submenu	automatically generated knot number 0
yXX	WORD	value of singel pin; development only!
s	submenu	Serial
su	submenu	UART-Config
sup	ENUM:"../Firmware/IO/serial.h":UARTParityModes_t	UART-Parity
sub	ENUM	UART-Bitrate
u	submenu	GPIO server
ue	submenu	Edge counter configuration of all GPIO ports (only ports 0 to 2 support edge counter)
uev	submenu	Values of edge counters of every pin
uevX	WORD	Value of single edge counter; only 0 may be assigned (i.e. counter reset)
ut	submenu	Types of all GPIO ports
utX	ENUM:"../Firmware/IO/gpio.h":PortType_t	Type of single GPIO port
uv	submenu	Values of all GPIO ports
uvX	WORD	Value of single GPIO port

Event trigger Einstellungen (Gruppe t)

Für die aktuelle Session werden die Event trigger eingestellt. Diese Kommandogruppe hat für JSON-Zugriff keine Bedeutung, da diese Kommunikation nur durch den Client initiiert werden kann.

Die Kommandos aktivieren oder deaktivieren Events (d.h. Flanken). Für digitale Eingänge wird 't' mit einer der IO-Kürzel zusammengesetzt. Ggf. kann das Prinzip auf digitale Ausgänge erweitert werden, wenn lokale Änderungen mitgeschnitten werden sollen. Dem kann einer der folgenden Werte zugewiesen werden:

Wert	Bedeutung
0	Kein Event
1	Steigende Flanken
2	Fallende Flanken
3	Alle Flanken

Events generieren Zustandstelegramme der Form 'a=1'. Für analoge Eingänge oder PWM-Ausgänge hat diese Einstellung keinen Effekt.

Trigger für den Prozesspeicher können mittels tq=value erfolgen. value ist eine der 16 Adressen, die aboniert werden sollen.