Echtzeit-Steuerung+Messung über Webbrowser

Ziel: Z.B. ein RC-Auto über die Weboberfläche verzögerungsfrei steuern und Sensoren beobachten, ohne gegen ein Hindernis zu fahren.

Problem: PHP-Formular haben eine Latenz → Button Klick → Daten werden an Webserver übertragen → PHP Skript läuft → Steuerung passiert auf dem Server → Rückmeldung an den Browser → kann ein paar Sekunden dauern.

Lösungsansatz: Upgrade der Verbindung auf → Websockets!

Beispiel 1: Umschalten auf Websocket-Verbindung beim Apache-Webserver (in der sites-enabled/*.conf für den jeweiligen Webserver) für WorkAdventure:

```
RewriteCond %{HTTP:Upgrade} websocket [NC]
RewriteCond %{HTTP:Connection} upgrade [NC]
RewriteRule ^/?(.*) "ws://localhost:8000/$1" [P,L]
```

Was passiert?→ Bei einer „upgrade: websocket“ HTTP-Anfrage leitet der Webserver die Verbindung intern (/Option [P] für „passthrough“) auf die Websocket-Anwendung auf Port 8000 weiter. Der Client ist weiterhin mit Port 443 verbunden.

Beispiel 2: NoVNC Javascript Client

Hier: Starten von `x11vnc` auf dem laufenden Desktop am Pi (wird exportiert), `websockify` zum Durchreichen per Web, und HTML/Javascript-Client für die graphische Verbindung → NoVNC.

```
# Vorher: apt install websockify, oder Download von der NoVNC-Webseite
# Hier: Der Port 6080 auf allen Interfaces wird umgeleitet auf den laufenden
# VNC-Server auf Port 5900 (localhost), der Javascript-Client verbindet sich
# in diesem Fall mit Port 6080 statt wie im Beispiel 1 auf 443 zu bleiben.
websockify -D 0.0.0.0:6080 127.0.0.1:5900

# VNC-Server starten, der die bereits laufende (!) Grafik-Konsole übers Netz
# zugänglich macht.
x11vnc -loop -forever -noxdamage -rfbauth $HOME/.vnc/passwd -rfbport 5901 \
       >/dev/null 2>&1  &

# Auf der Client-Seite (bzw. in den HTML+Javascript-Dateien, die der Server
# an den Webbrowser als Client schickt) müssen die entsprechenden Ports
# als Ziel eingetragen werden.
[…]
// By default, use the host and port of server that served this file
host = WebUtil.getQueryVar('host', window.location.hostname);
// port = WebUtil.getQueryVar('port', window.location.port);
port = WebUtil.getQueryVar('port', 6080);
[…]
rfb.connect(host, port, password, path);
```

→ Der Javascript-Client verbindet sich über den `websockify`-Proxy mit dem VNC-Server.

Websockify für Debian: `sudo apt install websockify`

Beispiel Roboter-Steuerung: Gibt es ein Desktop-Tool für den Roboter? Dann einfach per VNC (s.o.) den Desktop freigeben, und per Webclient den Desktop im Browser steuern. :-)
Ansonsten → Javascript-Library für websockets verwenden. Möglicherweise unterstützt die Roboter-Software serverseitig schon websockets.