PROFITIPPS ANTRIEBSSTEUERUNG (ART. 180290)

ACHTUNG: Diese Anleitung richtet sich an Personen, die das entsprechende Know-how haben. Faller stellt die folgende Information ohne irgendwelche Garantie und ohne der Möglichkeit Support in Anspruch zu nehmen zur Verfügung.
Der Faller Containerkran wird bekanntlich über einen Webbrowser von einer html-Seite her gesteuert. Diese Seite wandelt Benutzer-Interaktion, wie z.B. Taste drücken und / oder loslassen in HTTP GET Requests um und sendet diese ab.
Wer die Steuerung (z.T.) automatisieren oder in bestehende Systeme einbinden will, oder wer die zur Verfügung gestellte Seite ersetzen will, kann das mit der folgenden Information tun.

ALLGEMEINES

Die gesamte Steuerung erfolgt über http GET Requests. Zu beachten ist aber, dass es 2 Steuerungen gibt, jede mit eigener IP-Adresse. Um z.B. Seil oder Katze zu bewegen, muss die zweite Steuerung angesprochen werden, für die Hauptfahrwerk jedoch die erste. Die erste Steuerung ist die von der die Faller-seitig zur Verfügung gestellte html-Seite geladen wird.
Obwohl möglich, raten wir dringend davon ab, einmalige Vorgänge, wie z.B. das Eingeben oder Ändern eines Passwortes oder die Konfiguration von Zusatz-Funktion, wie z.B. eines 3. Servo-Motors oder einer zusätzlichen LED, selbst zu programmieren: Dies würde sehr komplex und bringt (wegen der „Einmaligkeit“) nicht viel. Deshalb stellen wir die dafür notwendige Information auch nicht allgemein zur Verfügung. Selbstverständlich können Sie aber das Ansteuern aller Komponenten selbst in die Hand nehmen.
WICHTIG: nicht alle Requests sind zu jeder Zeit freigeschaltet. Deshalb sollten Sie die oben erwähnten einmaligen Konfigurationen soweit vollständig durchgeführt haben, bevor Sie Ihre eigene Steuerung in Betrieb nehmen. Andererseits können Sie im täglichen normalen Betrieb komplett auf die von uns zur Verfügung gestellte Webseite verzichten, wenn Sie das möchten. Sie können aber auch jederzeit letztere parallel zu Ihrer eigenen Steuerung verwenden.

NOTWENDIGE VORAUSSETZUNGEN UND INFORMATIONEN

Wir werden im Folgenden kein Wissen über Basistechnologie, wie TCP/IP, WLAN, HTTP oder auch Programmierung in (Script-) Sprachen weitergeben. Wenn Sie aber solches Wissen haben, können Sie auf allen Geräten, welche die eingesetzten Basis-Technologien unterstützen, Ihre eigene Steuerung erstellen, beispielsweise Android- oder I-Phone, Microsoft-Windows (TM) oder MAC Rechnern oder auch Linux-Geräte wie der Raspberry PI bis hinunter zu Arduino (oder ähnlichen) Micro-Controllern. Im Falle Arduino sollten Sie aber mindestens auf „MEGA“ oder ESP8266 o.ä. zurückgreifen da z.B. ATMEGA328 zu schwach für die Aufgabe sein könnte.
Wenn Sie etwas selbst machen möchten, aber ggf. vorhandene Wissenslücken nicht im Selbststudium füllen möchten oder können: Bei genügend Interesse planen wir Kurse, in denen Sie Gelegenheit bekommen, einige Beispiele durchzuarbeiten und zwar unter Anleitung des Architekten der Software auf den Steuerungen!
Alle Kommunikation erfolgt über WLAN (2,4 Ghz). Dazu haben beide Steuerungen ihre eigenen WLAN-Interfaces. Im Normalbetrieb (d.h. wenn Sie ein WLAN-Passwort eingegeben haben) agiert die primäre Steuerung (Hauptfahrwerk) zugleich als Web-Server und als WLAN-Access-Point, während die sekundäre Steuerung nur als Web-Server fungiert. In diesem Falle hat die primäre Steuerung immer die IP (4) Adresse 172.17.217.217. Die sekundäre Steuerung bekommt Ihre Adresse über DHCP im Bereich ab 172.17.217.60. Ihre Steuerung(en) erhalten so ebenfalls ihre Adressen in diesem Bereich. ACHTUNG: Der WLAN-AP in der primären Steuerung unterstützt maximal 4 Knoten gleichzeitig. D.h. Neben den beiden Steuerungen ist nur noch Platz für max. 2 Computer!

Sollten Sie die Steuerungen in ein eigenes WLAN Netz integriert haben, etwa weil Sie dieses für weitere Steuerungsaufgaben nutzen, müssen Sie beiden Steuerungen ermöglichen ihre IP (4) Adressen über DHCP zu beziehen. Am besten richten Sie auf Ihrem WLAN-Access-Point dazu über die MAC-Adresse reservierte IP-Adressen ein, sodass die

Steuerungen immer die gleiche Adresse erhalten.
In jedem Falle reagieren die beiden Steuerungen aber auf mDNS Anfragen und geben so Namen und zugehörige Adressen bekannt. (Auf diesem Wege finden sich die beiden Steuerungen auch „intern“). Leider unterstützt Microsoft-Windows (TM) bisher mDNS noch nicht selbst. D.h. Windows-user müssten dafür auf weitere Software wie z.B. AVAHI oder „Bonjour“ -Browser zurückgreifen. Andererseits können Sie natürlich immer über Ihren Access-Point die Adressen herausfinden. Die mDNS Namen sind immer in der Form: fallernodexx.local, wobei xx zwei hexadezimal Ziffern sind (z.B. : „f9“). Die beiden Endziffern der primären Steuerung sind die gleichen, wie die der SSID des WLANs nach der ersten Passwort Eingabe (z.B. „fallerWIFIf9“). Manche Access-Points (wie z.B. neuere Fritz boxen) stellen die Namen auch in ihrem eigene DNS-Server zur Verfügung, allerdings oft in der eigenen Domäne (z.B. „fallernodef9.fritz.box“).

Um die eigene Steuerung zu erstellen, können Sie auf alle Software zurückgreifen, welche in der Lage ist HTML GET Requests abzusetzen. Nach und nach werden wir kleine Beispiele liefern für wget (und damit batch oder shell Scripte), CURL, Python und javascript (nodejs). Sollten Sie einen Web-server in ihrem Steuerungs-Netzwerk haben, können Sie natürlich Javascript auch aus einer eigenen HTML-Seite heraus nutzen, auch dafür geben wir Beispiele. Bitte verstehen Sie, dass wir die ganzen Beispiele aber erst nach und nach erstellen können. Sollten Sie z.B. einen FHEM Server betreiben, können Sie unsere Beispiele auch einfach in PERL übersetzten.

EINSTIEG

Aller Anfang ist kinderleicht! Sie brauchen dafür nichts was Sie nicht schon haben und kennen:
Damit wir loslegen können, brauchen Sie einen funktionierenden Containerkran und ein Notebook, Desktop-Computer oder ein Smartphone oder Tablet. Dabei spielt das Betriebssystem praktisch keine Rolle. Nur ein Web-Browser muss vorhanden sein und natürlich müssen Sie alles im gleichen WLAN-Netzwerk haben. Also, wenn Sie mit der von uns zur Verfügung gestellten Steuerung (html-Seite) den Containerkran steuern können kann es los gehen. Selbst wenn wir einen Touchscreen haben und damit unsere Seite nicht (richtig) funktioniert macht das für diesen Einstieg nichts!
In allen Beispielen verwenden wir keine IP-Adressen, sondern die Namen: fallernode86.local für die primäre Steuerung und fallernode46.local für die sekundäre. Diese müssen Sie immer durch Ihre Namen oder besser die entspr. IP-Adressen ersetzen.
Geben Sie nun als URL in ihrem Browser ein:
http://fallernode86.local/startM?sNr=1&turn=1&tio=1000
Das Hauptfahrwerk sollte nun 1 Sekunde nach rechts laufen. Mit turn=-1 sollte Sie 1 Sekunde nach links laufen.
Sollte etwas nicht funktionieren, können Sie einmal &sync=0 anhängen. Das schaltet die Synchronisierung der beiden Hauptfahrwerk Motoren aus (Achtung das könnte zu „Verspannungen“ führen!).
Wenn es funktioniert hat, wie wäre es dann mit:
http://fallernode46.local/startM?sNr=1&turn=1&tio=1000
Jetzt sollte sich die Laufkatze (oder der Spreader?) eine Sekunde bewegen.
Ein weiteres Beispiel:
http://fallernode46.local/fswitch?nr=3&blink=500
Jetzt sollte die (grüne) LED auf der Anschluss-Platine der sekundären Steuerung blinken. Mit blink=0 kann das wieder ausgeschaltet werden. Das funktioniert aber nur, wenn der switch 3 (Pin 16) in der Konfiguration eingeschaltet ist!
Sie können nun noch weitere Versuche machen, besonders mit den http Requests, welche Sie später in Ihrer eigenen Steuerung verwenden wollen. Achten Sie dabei auch immer auf die (ziemlich kleine!) Anzeige im Browser. Wenn der Befehl erkannt und ausgeführt wurde, sehen Sie dort ein kleines „ok“. Bei falschen Parametern oder wenn der Wert außerhalb des vorgesehenen Wertebereiches ist (oft!) „Bad ARGS“. Die Liste der HTTP GET Requests mit Parametern und kurzer Beschreibung finden Sie ganz am Ende! Falls Sie sich verschreiben, werden Sie übrigens oft auf unsere Steuerungsseite umgeleitet, anstatt der sonst üblichen 404 Seite.

SHELL SCRIPTE / WINDOWS BATCH

Die einfachste Art etwas zu Automatisieren oder als Abfolge zu Steuern, ist es ein Windows Batch File oder MAC/ Linux Shell Script zu erstellen. In Verbindung mit „CRON“ oder dem AT-Kommando bei Windows (bzw. neuer: schtask) lässt sich z.B. auch zeitgesteuert etwas ausführen.
Die einfachste Art und Weise auf der Kommandozeilen Ebene zu Arbeiten ist mit „wget“. Gleich zeigen wir ein Beispiel. Natürlich können Sie, wenn Sie ein etwas mächtigeres Werkzeug brauchen auch „cURL“ verwenden. Es gibt auch Kommandozeilen Browser. Nehmen Sie das was Sie sowieso schon benutzen. Wenn das alles neu ist, sollten Sie sich auf wget konzentrieren.

Wget sollte bei Linux oder MAC OSX schon dabei sein. Wenn nicht oder wenn Sie Microsoft Windows (TM) verwenden, müssen Sie sich eine passende Installation suchen. Für Windows reicht es aus, das ZIP-File herunterzuladen und das .exe und evtl. vorhandene DLLs in Ihrem Ausführungs- Pfad zu kopieren (z.B. \windows\system32). Ihre Suchmaschine ist Ihr Freund! Auch wenn Sie wget und/oder cURL (Alternativen) für Android oder I-Phone suchen.

Nehmen wir nun an, dass Sie es geschafft haben, wget auf Ihren Computer zu bringen und dieser Computer jetzt im gleichen WLAN wie der Containerkran ist. Wir öffnen ein Kommandozeilen Fenster: Bei Windows cmd.exe oder bei MAC/Linux ein Terminal. Jetzt sollte es genauso einfach sein, wie beim Einstieg zum Test einige Kommandos an den Containerkran abzusetzen:
wget “http://fallernode86.local/startM?sNr=1&turn=1&tio=1000“ -q -O NUL
sollte wieder das Hauptfahrwerk für eine Sekunde in Bewegung setzen. -q können Sie probehalber weglassen, dann sehen Sie ein Protokoll des Vorganges. -O NUL (bei Windows) verhindert, dass ein Ausgabefile geschrieben wird.
Probieren Sie auch noch:
wget “http://fallernode46.local/startM?sNr=1&turn=1&tio=1000“ -q -O NUL
und
wget “http://fallernode46.local/startM?sNr=2&turn=1&tio=1000“ -q -O NUL
für Laufkatze und Spreader.

Jetzt können Sie einige wget Kommandos in ein Batch- oder Shell Script File schreiben, vielleicht mit einigen „sleep“-s dazwischen, oder vielleicht sogar mit einigen if Abfragen. Vielleicht gibt es ja Bedingungen, wie etwa die Einfahrt eines Zuges, welche Sie mit Kommandozeilen-Parametern abfragen können. Das Ganze lässt sich auch Zeitgesteuert abrufen (z.B. mit „cron“).

Sie sehen, mit wget (z.B.) haben Sie schnell eine einfache Automatisierung erreicht. Sollten Sie aber komplexere Bedingungen abfragen wollen, oder zum Beispiel das Eintreffen bestimmter Events reagieren wollen, sollten Sie es mit einer mächtigeren Script-Sprache versuchen. Sie haben sicher schon erkannt, dass die Script- oder Programmiersprache, die Sie schon kennen dazu gut geeignet ist. Als Beispiel nehmen wir im nächsten Kapitel PYTHON.

PYTHON ALS BEISPIEL FÜR SCRIPT UND PROGRAMMIERSPRACHEN

Die kleinen Beispiele im Folgenden beziehen sich auf PYTHON 3 (mindestens 3.8). Falls Sie lieber mit PYTHON 2 arbeiten, gibt es einige kleinere syntaktische Änderungen. Es sollte aber nicht zu schwierig sein, das anzupassen.
Zu Anfang zeigen wir ein Beispiel mit einem minimalen Python Script, welches geeignet ist im Hintergrund dauernd zu laufen. Deshalb gibt es so gut wie keine Benutzer Interaktion und vor allem kein graphisches Benutzerinterface. Danach zeigen wir aber noch ein Beispiel, wie mit einer graphischen Benutzeroberfläche, eine einfache Fernbedienung erzeugt werden kann. Die könnte dann die von uns zur Verfügung gestellte html-Seite im Normalbetrieb ersetzen.

Für das Absetzten der HTTP GET Requests brauchen Sie das Modul: „requests“. Für das graphische Beispiel verwenden wir das Modul „tkinter“. Normalerweise werden solche Module einfach mit pip (bzw. pip3 in unserem Falle) nach installiert. Aber auch dies sprengt den Rahmen unserer kleinen Anleitung und wir gehen davon aus, dass Sie ggf. andere Quellen finden, um die PYTHON Installation und die Modul-Installation zu erlernen.

Auch für jemanden der Python noch nie verwendet hat (aber ein klein wenig Programmiererfahrung hat) sollten die folgenden Beispiele machbar sein, sowie auch einige Anpassungen an eigene Vorstellungen. Im Gegensatz dazu, wollen wir auf andere Sprachen (z.B. PERL oder BASIC) verzichten, weil wir glauben, dass Sie selbst die PYTHON Beispiele leicht adaptieren können, wenn Sie eine andere Script- oder Programmiersprache beherrschen.
Auch das „Event-Management“ simulieren wir nur über kleine Files („flags“), denn wir wissen nicht, was Sie sonst als Steuerung anderer Komponenten benutzen.

Hier unser Beispiel1. Für die ersten Versuche starten Sie das Programm aus einer Console (cmd.exe bzw. Terminal) mit:

python3 example.py	oder einfach
./example.py	bzw.\example.py (windows)

So können Sie es mit ^C jederzeit stoppen.

```
#!/usr/bin/python3
# Beispiel:       Endlose Schleife
#         Wenn das File : myFlag.txt existiert
#         wird die LED (sek. Steu.) eingeschaltet
#         3 Sekunden später fährt das Hauptfahrwerk für 5 Sekunden
#         das File myFlag.txt wird gelöscht
#         und kurz darauf die LED wieder gelöscht.
#         Dann warteten wir auf die nächste Runde
import requests
import os.path
from os import path
import datetime
from time import *

#die folgenden Definitionen müssen angepasst werden:
URLmain = „http://192.168.178.39“       # die IP Adresse der primären Steuerung
URLseco = „http://192.168.178.33“       # die IP Adresse der SEKUNDÄREN Steuerung
LED = „3“                                                       # die fswitch Nummer der LED der sekundären Steuerung
hPhase = 500
FFile = „./myFlag.txt“
#ab hier benutzen as is

def stopM(node,servo):
        CMD = „/stopM?sNr=“+str(servo)
        URL = URLmain if node == 1 else URLseco
        r = requests.get(URL + CMD)

def startM(node,servo,richtung):
        CMD = „/startM?tio=5000&sNr=“+str(servo)+ „&turn=“ + str(richtung)
        URL = URLmain if node == 1 else URLseco
        r = requests.get(URL + CMD)

def fswitchB(onOff):
```

```
            CMD = „/fswitch?nr="+ LED +"&blink="
            if onOff :
                  CMD += str(hPhase)
            else :
                  CMD += „0"
            URL = URLseco + CMD
            r = requests.get(URL)

now = datetime.datetime.now()
print(„Pyhton example 1\nEndlosschleife\nwenn File myFlag.txt existiert action")
print (str(now)[:-7] + „: Beginn, wir warten!")
Richtung = 1
while True :
            if path.exists(FFile) :

now = datetime.datetime.now()
                  print (str(now)[:-7] + „: action")
                  fswitchB(True)                        # show that something happens
                  sleep(3)                        # 3 Sekunden warten
                  startM(1,1,Richtung)      # fahre 5 Sekunden in eine Richtung
                                                      # ACHTUNG: start wartet nicht auf stop /
timeout!
                  Richtung = -1 if Richtung == 1 else 1       # Drehe um!
                  os.remove(FFile)
            sleep(6)                                    # idle 10 seconds
            fswitchB(False)                       # Blinklicht stoppen

print („never here!")
```

Das Beispiel2 mit der graphischen Oberfläche wird in Kürze separat nachgeliefert.

ANHANG: CONTAINERKRAN HTTP GET REQUESTS

Hier jetzt die Sammlung der Requests mit Kurzbeschreibung und möglichen Parametern:
Wie schon mehrfach erwähnt sind alles GET Requests und sehen so aus:
http://fallernodexx.local/request?parameter=wert&pam2=wert...
Bzw.:
http://192.168.111.222/request?parameter=wert&pam2=wert...
Meistens gibt es als Rückgabe Wert nur (200) „ok" oder (500) „BAD ARGS". Es gibt Requests ohne Parameter und solche mit (1 oder mehrere); unter Umständen auch dynamisch, d.h. manche Parameter sind von anderen abhängig oder optional. Die Liste zeigt nur den Teil nach der Adresse, mit den optionalen Parametern in eckigen Klammern [], also z.B.:
/request?parameter=wert[&p2=wert2][&p3=wert3]
Alle Requests werden von beiden Steuerungen honoriert.

```
        CMD = „/fswitch?nr="+ LED +"&blink="
        if onOff :
            CMD += str(hPhase)
        else :
            CMD += „0"
        URL = URLseco + CMD
        r = requests.get(URL)

now = datetime.datetime.now()
print(„Pyhton example 1\nEndlosschleife\nwenn File myFlag.txt existiert action")
print (str(now)[:-7] + „: Beginn, wir warten!")
Richtung = 1
while True :
        if path.exists(FFile) :

now = datetime.datetime.now()
            print (str(now)[:-7] + „: action")
            fswitchB(True)                          # show that something happens
            sleep(3)                            # 3 Sekunden warten
            startM(1,1,Richtung)      # fahre 5 Sekunden in eine Richtung
                                                    # ACHTUNG: start wartet nicht auf stop /
timeout!
            Richtung = -1 if Richtung == 1 else 1       # Drehe um!
            os.remove(FFile)
        sleep(6)                                    # idle 10 seconds
        fswitchB(False)                         # Blinklicht stoppen

print („never here!")
```

Das Beispiel2 mit der graphischen Oberfläche wird in Kürze separat nachgeliefert.

ANHANG: CONTAINERKRAN HTTP GET REQUESTS

Hier jetzt die Sammlung der Requests mit Kurzbeschreibung und möglichen Parametern:
Wie schon mehrfach erwähnt sind alles GET Requests und sehen so aus:
http://fallernodexx.local/request?parameter=wert&pam2=wert...
Bzw.:
http://192.168.111.222/request?parameter=wert&pam2=wert...
Meistens gibt es als Rückgabe Wert nur (200) „ok" oder (500) „BAD ARGS". Es gibt Requests ohne Parameter und solche mit (1 oder mehrere); unter Umständen auch dynamisch, d.h. manche Parameter sind von anderen abhängig oder optional. Die Liste zeigt nur den Teil nach der Adresse, mit den optionalen Parametern in eckigen Klammern [], also z.B.:
/request?parameter=wert[&p2=wert2][&p3=wert3]
Alle Requests werden von beiden Steuerungen honoriert.

/getOtherEsp
retourniert (200) die IP-Adresse der jeweils anderen Steuerung

/getBat
retourniert die Batterie Spannung in Millivolt

/startM?sNr=n&turn=d[&tio=t][&sync=y]
startet den Servomotor Nr=n, 1<=n<=3, in Richtung=d, mit d = -1 links oder d = 1 rechts.
Nach t Millisekunden (<= 10000) stoppt der Motor selbständig, wenn nicht inzwischen ein weiterer Start-Request abgesetzt wurde. Lässt man tio weg, stoppt der Motor nach etwa 1 Sekunde. Der Parameter sync wirkt nur auf die Hauptfahrwerk Servos: mit sync=1 oder ohne den Parameter werden die beiden Motoren synchronisiert, mit sync=0 werden sie nicht synchronisiert (ACHTUNG, das kann zu „Verspannung" führen!)
Bei dem Hauptfahrwerk bewirken sNr=1 und sNr=2 das gleiche und wirken auf beide Motoren
Sonst wirken sNr=1 auf die Laufkatze und sNr=2 auf den Spreader. Die sNr=3 bewirkt nur etwas, wenn ein zusätzlicher Servo (etwa für das Drehen der Katze) installiert ist.
Bei dem Hauptfahrwerk wird versucht das Ende zu erkennen. Dann stoppen die Motoren und können zunächst nur in die andere Richtung gestartet werden.

/stopM?sNr=n
Stoppt den Servo Nr n (1..3). Nummerierung siehe /startM, 0 stoppt alle Servos dieser Steuerung

/fswitch?nr=n&v=v[&dim=d]
oder nächste Zeilen!
Zum Ein und ausschalten zusätzlicher Schalter. Diese müssen entsprechend konfiguriert sein!
n=1..3, v=0..100 ausschalten bis einschalten zu 100% , dim=10..1000 = 1/100 Sekunden fade in oder fade out, z.B. dim=100 => komplettes fade in / out in 100/100 = 1 Sekunde

/fswitch?nr=n&blink=b
n=1..3 (wie oben!) schaltet Blinklicht ein mit b = ½ Periode, d.h. Anphase = Ausphase = b in Millisekunden, z.B. b=500 => ½ Sekunde an und ½ Sekunde aus, b=0 schaltet Blinklicht aus

/fswitch?nr=n&toggle=1
Schaltet aus, wenn an war und ein (100%) wenn aus war.

Befehle, um die Geschwindigkeit der Servos anzupassen liefern wir noch nach.

PYTHON BEISPIEL MIT GUI

Das Beispiel 2 kreiert ein „Mini-Pad“, d.h. eine Fernsteuerung mit 9 Knöpfen:

Für die ersten Versuche starten Sie das Programm aus einer Console (cmd.exe bzw. Terminal) mit:

python3 example2.py oder einfach
./example2.py bzw .\example2.py (windows)

In der 9. und 10. Zeile müssen Sie vor dem Start die Adressen anpassen! Am Besten schliessen Sie das Beispiel über die graphische Oberfläche. Bei diesem einfachen Beispiel stoppen die Servomotoren automatisch nach max. 8 Sekunden oder wenn „Stop“ gedrückt wird. Wird der „Start“ innerhalb von 8 Sekunden widerholt, beginnen die 8 Sekunden von vorne.

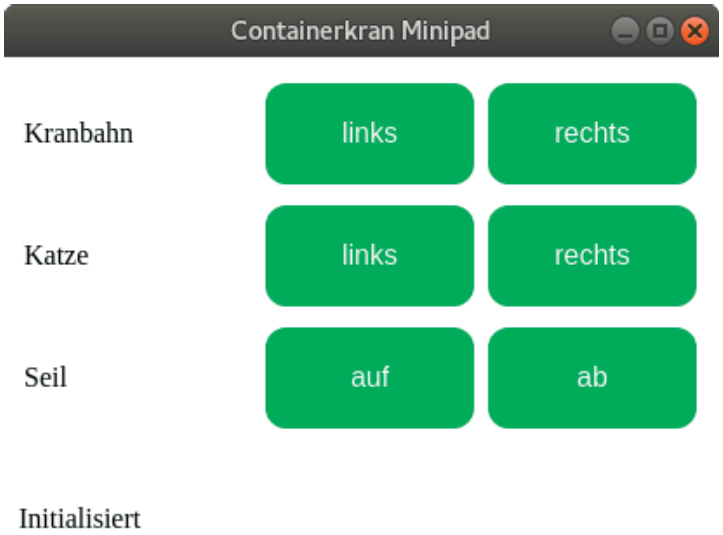
```
#!/usr/bin/python3
import requests                 # pip3 install requests  ist evtl. vorher notwendig
import tkinter                  # normalerweise bei neueren python versionen enthalten
from tkinter import *

#die folgenden definitionen müssen angepasst werden
URLmain = „http://192.168.178.39“
URLseco = „http://192.168.178.33“
#ab hier benutzen as is

top = Tk()
top.title(„Steuerung 1“)
top.geometry(‚400x150+400+100‘)

def stopM(node,servo):
        CMD = „/stopM?sNr=“+str(servo)
        URL = URLmain if node == 1 else URLseco
        print ( URL + CMD )
        r = requests.get(URL + CMD)
```

```
def startM(node,servo,richtung):
        CMD = „/startM?tio=8000&sNr="+str(servo)+ „&turn=" + str(richtung)
        URL = URLmain if node == 1 else URLseco
        print ( URL + CMD )
        r = requests.get(URL + CMD)

def startM110():
        startM(1,1,0)

def stopM1():
        stopM(1,1)

def startM11m1():
        startM(1,1,-1)

f0 = Frame(top, width=100)
f0.pack(side=LEFT)
l1= Label(f0,text="Krahnbahn")
l1.pack(anchor='w',side=LEFT,padx=5,pady=5)
l2 = Label(f0,text="Katze   „)
l2.pack(anchor='w',side=LEFT, padx=5,pady=5)
l3 = Label(f0,text="Seil   „)
l3.pack(anchor='w',side=LEFT, padx=5,pady=5)


f1 = Frame(top)
f1.pack(side=RIGHT)

startB = Button(f1, text = „Start rechts", command = startM110)
startB.pack(side=LEFT, padx=5, pady=5)

stopB = Button(f1, text = „Stop", command = stopM1)
stopB.pack(side=RIGHT, padx=5, pady=5)

startB2 = Button(f1, text = „Start links", command = startM11m1)
startB2.pack(side=RIGHT, padx=5, pady=5)
f2= Frame(top)
f2.pack(side=RIGHT)

startB21 = Button(f2, text = „Start rechts", command = startM110)
startB21.pack(side=LEFT, padx=5, pady=5)

stopB2 = Button(f2, text = „Stop", command = stopM1)
stopB2.pack(side=RIGHT, padx=5, pady=5)
```

```
startB22 = Button(f2, text = „Start links“, command = startM11m1)
startB22.pack(side=RIGHT, padx=5, pady=5)
f3= Frame(top)
f3.pack(side=RIGHT)

startB31 = Button(f3, text = „Start rechts“, command = startM110)
startB31.pack(side=LEFT, padx=5, pady=5)

stopB3 = Button(f3, text = „Stop“, command = stopM1)
stopB3.pack(side=RIGHT, padx=5, pady=5)

startB32 = Button(f3, text = „Start links“, command = startM11m1)
startB32.pack(side=RIGHT, padx=5, pady=5)
mainloop()
```

BEISPIEL 3: JAVASCRIPT / NODEJS / ELECTRON

Das 3. beispiel ist ebenfalls ein Mini-Pad, diesmal mit nur 6 Knöpfen:



Bei diesem Beispiel können die 3 Servomotoren jeweils nach links oder rechts (drehend) gestartet werden. Die Bedeutung des Knopfes wechselt dann von „start“ auf „stop“ und die Farbe auf rot. Das (java)-Script repetiert den Start alle 350 Millisekunden, bis „stop“ gedrückt wird.
Das Beispiel besteht aus einer html-Seite, welche das (java)-Script enthält und einem separaten script (main.js). Dazu kommt die Applikation-Konfiguration: package.json. Hat man eine Konsole im Installationsverzeichnis (d.h. dort wo die drei Files abgelegt sind) startet man das Minipad mit: npm start.

Natürlich müssen Sie eine (genügend neue) node js / electron Umgebung installiert haben. Diese gibt es für Windows, Linux oder MAC-OS. Die Installation sprengt aber den Rahmen dieser Anleitung. Das Beispiel ist in dieser Form nur für Personen mit node js Erfahrung gedacht. Allerdings kann das File index.html auch einfach auf einen beliebigen Webserver in Ihrem Netz geladen und von dort via Browser genutzt werden. Nodejs könnte (z.B. mit IONIC) auch zur Programmierung einer Android oder IOS App genutzt werden.
Die anzupassenden IP Adressen stehen in Zeile 24 des Files index.html.
Package.json:

```
{
 „name“: „electron-ckminipad“,
 „version“: „1.0.0“,
 „description“: „mini Fernsteuerung fuer containerkran“,
 „main“: „main.js“,
 „scripts“: {
  „start“:“electron .“,
  „test“: „echo \“Error: no test specified\“ && exit 1“
 },
 „author“: „Steffen Norbert“,
 „license“: „ISC“
}
```

Main.js:

```
const { app, BrowserWindow, Menu } = require(‚electron‘)
Menu.setApplicationMenu(false)

function createWindow () {
 // Erstelle das Browser-Fenster.
 const win = new BrowserWindow({
  width: 400,
  height: 350,
  webPreferences: {
   nodeIntegration: true
  }
 })

 // and load the index.html of the app.
 win.loadFile(‚index.html‘)

 // Öffnen der DevTools.
 // win.webContents.openDevTools()
}
```

```
// This method will be called when Electron has finished
// initialization and is ready to create browser windows.
// Einige APIs können nur nach dem Auftreten dieses Events genutzt werden.
app.whenReady().then(createWindow)

// Quit when all windows are closed.
app.on(‚window-all-closed‘, () => {
 // Unter macOS ist es üblich, für Apps und ihre Menu Bar
 // aktiv zu bleiben, bis der Nutzer explizit mit Cmd + Q die App beendet.
 if (process.platform !== ‚darwin‘) {
  app.quit()
 }
})

app.on(‚activate‘, () => {
 // Unter macOS ist es üblich ein neues Fenster der App zu erstellen, wenn
 // das Dock Icon angeklickt wird und keine anderen Fenster offen sind.
 if (BrowserWindow.getAllWindows().length === 0) {
  createWindow()
 }
})
```

Index.html:

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="UTF-8">
  <title>Containerkran Minipad</title>
  <meta http-equiv="Content-Security-Policy" content="script-src ‚self‘ ‚unsafe-inline‘;" />
<style>

.button {
 background-color:  #4CAF50; /* Green */
 border: none;
 color: white;
 padding: 20px;
 text-align: center;
 text-decoration: none;
 display: inline-block;
 font-size: 16px;
 margin: 4px 2px;
 cursor: pointer;
 width: 120px;
 border-radius: 12px;
}
```

```
</style>
<script>
var node = [„http://192.168.178.39“,“http://192.168.178.33“]
var buttons = []
var bstate = []
var bcolor = [„green“,“red“]
var rfunc = [null,null,null,null,null,null,null]

function init() {
        for (var i = 1; i < 7 ; i++) {
                buttons[i] = document.getElementById(„b“ + i.toString())
                bstate[i] = 0
        }
        document.getElementById(„result“).innerHTML = „Initialisiert“
}

function startServo(mNode,sNr,direction){
        var xmlhttp = new XMLHttpRequest();
        xmlhttp.onreadystatechange = function() {
                if (this.readyState == 4 && this.status == 200) {
                        document.getElementById(„result“).innerHTML = „Servo „ + sNr.toString() + „ gestar-
ted“
                }
        };
  xmlhttp.open(„GET“, mNode+‘/startM?sNr=‘+sNr+‘&turn=‘+direction, true);
        xmlhttp.send();
}

function stopServo(mNode,sNr) {
        var xmlhttp = new XMLHttpRequest();
        xmlhttp.onreadystatechange = function() {
                if (this.readyState == 4 && this.status == 200) {
                        document.getElementById(„result“).innerHTML = „Servo „ + sNr.toString() + „ ge-
stoppt“
                }
        };
  xmlhttp.open(„GET“, mNode + „/stopM?sNr=‘+sNr, true);
        xmlhttp.send();
}
```

```
function bClicked(nr) {
        bstate[nr] = (bstate[nr] == 0) ? 1 : 0;
        document.getElementById(„result“).innerHTML = „changed „ + nr + „ color: „ + bcolor[bstate[nr]];
        buttons[nr].style[„background-color“] = bcolor[bstate[nr]];
        var mNode = (nr < 3 ) ? node[0] : node[1]
        var mTurn = ( (nr&1) == 1) ? „-1“ : „1“
        var mServo = ( nr > 4 ) ? „2“ : „1“
        if ( bstate[nr] == 0 ) {
                clearInterval(rfunc[nr])
                stopServo(mNode,mServo)
        } else {
                startServo(mNode,mServo,mTurn)
                rfunc[nr] = setInterval(startServo,350,mNode,mServo,mTurn)
        }
}
</script>
</head>
<body onLoad=“init()“>
<table>
<tr><td width=“240px“>Kranbahn</td>
<td><button class=“button“ onclick=“bClicked(1)“ id=“b1“>links</button></td>
<td><button class=“button“ onclick=“bClicked(2)“ id=“b2“>rechts</button></td>
</tr>
<tr><td>Katze</td>
<td><button class=“button“ onclick=“bClicked(3)“ id=“b3“>links</button></td>
<td><button class=“button“ onclick=“bClicked(4)“ id=“b4“>rechts</button></td>
</tr>
<tr><td>Seil</td>
<td><button class=“button“ onclick=“bClicked(5)“ id=“b5“>auf</button></td>
<td><button class=“button“ onclick=“bClicked(6)“ id=“b6“>ab</button></td>
</tr></table>
<br><br>

<div id=“result“></div>
</body>
</html>
```