Einbindung von Webcams in TrainController

Hardware Kamera

Unsere Modellbahn-Anlage ist über viele Räume verteilt, da macht es Sinn, dass man die Bauabschnitte mit Hilfe von Kameras überwachen kann. Wir haben derzeit insgesamt 4 Robocams in 4 Räumen installiert. Jede der Kamera hat eine IP-Adresse und kann über „Presets“ auf vordefinierte Positionen fahren.

Eine der Ideen war nun, dass TrainController diese Kameras steuern soll, denn wer, wenn nicht TC weiß, wo gerade Züge unterwegs sind.

Um den ganzen Raum überwachen zu können, bedarf es einer PTZ-Kamera. PTZ steht für „Pan-Tilt-Zoom“, also eine Kamera, die (ferngesteuert) sich drehen, neigen und zoomen kann. Die Kamera sollte Kommandos per http-Protokoll verstehen bzw. auch Voreinstellungen, so genannte „Presets“ speichern können.

Bei der Wahl der Kamera ist darauf zu achten, dass man Kommandos über eine frei zugängliche API zu steuern ist, d.h. der API-Code sollte quelloffen bereit liegen.

Einbindung in TrainController

Und damit kommen wir zur Schnittstelle zu TrainController: TC kann ja Windows-Programme (z.B. auch Batch-Files) starten. Ich starte dann „einfach“ ein Curl-Kommando. Curl erlaubt auf Kommandozeilenebene HTTP-Aufrufe. Somit könnte man Schalter, Taster, … im Gleisbild ablegen, die dann ein Batch-Skript aufruft, das wiederum ein Curl-Kommando aufruft. Weitere Knöpfe können die Presets der Kamera, also die vorher eingestellten Positionen und Zoom-Einstellungen der Kamera, aufrufen.

Wenn das einmal im Stellwerk (oder einem weiteren Stellwerk) hinterlegt ist, dann bedarf es einer Automatik, die solche Knöpfe für uns „drücken“. Denkbar ist, dass das z.B. Aktionsmarkierungen in einem Blockplugin-autotooltip__default plugin-autotooltip_bigBlöcke im TrainController

Blöcke (Fahrdienstleiter)

Der Fahrdienstleiter steuert den Zugverkehr auf der Basis eines Blocksystems. Zu diesem Zweck wird die Modellbahnanlage gedanklich in Blöcke aufgeteilt. Überall dort, wo Loks oder Züge kontrolliert, angehalten, abgestellt oder überwacht werden sollen, wird ein übernehmen. ES können aber auch andere Elemente sein, die so eine Aktion ausführen.

Was genau aufgerufen wird, hängt immer von der Kamera ab, da jeder Hersteller da seine eigenen Parameter definiert. Die Funktionalitäten sind zwar immer gleich, aber die „Sprache“ ist es nicht. Deswegen habe ich beim Kauf darauf geachtet, dass die „API“ der Kamera dokumentiert und frei verfügbar ist.

Ich habe eine „HiLook Mini PTZ Camera“ und die Dokumentation dafür kann man hier einsehen:

hikvision_cgi_ipmd_v1.5.9.pdf

Andere Hersteller dokumentieren ihre Kameras nicht. Bei einigen - z.B. D-Link - kann man sich die Kommandos aus dem Internet zusammensuchen. Ich habe zuerst alte D-Link Kameras benutzt, die ich noch von einem anderen Projekt übrig hatte. Die lösten aber nicht hoch genug auf und hatten keinen optischen Zoom. Eine neue HD-Kamera von D-Link hatte die gleiche API, aber D-Link hat deren Benutzung deutlich erschwert.

Prinzipiell lässt sich aber jedes IoT-Gerät steuern, das über eine (REST-)API verfügt. Das könnten z.B. auch Philips Hue Lampen sein. Tag/Nacht-Wechsel mit „Abendrot“ usw. würde also auch gehen durch reine Programmierung.

Ein Einstieg dazu findet sich hier: https://developers.meethue.com/develop/get-started-2/

Das wird mein nächstes IoT-Projekt. Ich habe in TC eine Programmerung, die Probleme beim Zugverkehr erkennt. Die Anlage wird dann gestoppt. Die Beleuchtung der Schattenbahnhöfe soll dann automatisch angeschaltet werden.

Wie man dann wann die Kamera positioniert, bleibt jedem dann selbst überlassen. Das kann eine Operation zu Beginn einer Zugfahrtplugin-autotooltip__default plugin-autotooltip_bigZugfahrten

* Anpassung an das Anfahrverhalten * Aufenthalt, Startverzögerungen und Beschleunigung * Beschreibung der Zugfahrt Regeln * Zugfahrt bwegt mit Kuppeln * Zugoperationen im TrainController * Die Mindest-Beschleunigungszeit in Zugfahrtregel * induphil-01-rundfahrt * induphil * Zugfahrten: Nachfolger * Operationen bei Start und Stop * Zugfahrt Pendelfahrt * Sind alle Blöcke und Strecken für die Zugfahrt markiert? * Unterbrechen des Betriebs – Beenden von Zugfahr… sein oder eben eine Aktionsmarkierung. Ich habe dazu dann Makros definiert, die von der Aktion ausgelöst werden und letztendlich den Preset-Taster drückt. Makro deswegen, weil ich noch etwas Zusatzintelligenz drin habe:

Zuerst habe ich noch zwei Schalter definiert: 1. Steuerung generell an und aus. 2. Kamera ist grade in Benutzung.

Jedes Makro testet zuerst beide Schalter, schaltet dann den „Benutzungsschalter“ - womit alle anderen Makros gesperrt werden, drückt dann den Preset und gibt nach ein paar Sekunden die Kamera wieder frei. Damit gibt man der Kamera genug Zeit, den Zug durchs Bild fahren zu lassen ohne dass eine andere Aktion die Kamera übernimmt, also im Mehrzugbetrieb die Kamera wild hin- und herschwenkt.

Prinzipiell gibt es drei verschiedene Kommandos: Kamera bewegen, Preset setzen, Preset aufrufen.

Am einfachsten sind die Kommandos für das Aufrufen eines Presets. Hier ein Beispiel meiner Kamera zum Aufruf des Presets Nummer 3 (für alle Beispiele gilt, dass man die IP-Adresse der Kamera sowie Username und Passwort ersetzen muss):

curl -H „application/xml“ -X PUT -u „username:passwort“ „http://192.168.0.117/PTZCtrl/channels/1/presets/3/goto“

Hinter „/preset/“ schreibt man also einfach nur die Nummer des Presets.

Die Kamera hat ein eigenes Web-Interface, mit dem man die Kamera positionieren und die Presets speichern kann. Dann würden die „Go“-Knöpfe ausreichen.

Ich habe aber auch diese Funktionen in TC übernommen.

Beispiel für das Kommando hinter einem Speicher-Knopf für Preset 3:

curl -H „application/xml“ -X PUT -u „username:passwort“ „http://192.168.0.117/PTZCtrl/channels/1/presets/3“ -d „@C:\Users\a_gra\Documents\xmlset3.txt“

Hier müssen zusätzliche Parameter im XML-Format mitgeschickt werden. Diese stehen in einer Datei mit dem Namen „xmlset3.txt“:

<PTZPreset> <enabled>true</enabled> <id>3</id> <presetName>MyPreset3</presetName> </PTZPreset>

Gespeichert wird die Position, die die Kamera in dem Moment hat. Man muss sie also vorher forthin bewegen.

Beispiel für das Bewegen der Kamera:

curl -H „application/xml“ -X PUT -u „username:passwort“ „http://192.168.0.117/PTZCtrl/channels/1/momentary“ -d „@C:\Users\a_gra\Documents\xmlleft500.txt“

Die Datei „xmlleft500.txt“ enhält dann die Parameter für das Bewegen der Kamera für die Bewegung nach links:

```
<?xml version="1.0" encoding="UTF-8"?>
<PTZData>
<pan>-1</pan>
<tilt>0</tilt>
<zoom>0</zoom>

<Momentary>
<duration>500</duration>
</Momentary>
</PTZData>
```

Die Parameter pan, tilt und zoom definieren also die Richtung (-1 = links/unten/näher ran, 0 = keine Bewegung, 1 = rechts/oben/weiter weg) Duration ist die Bewegungszeit in Millisekunden.

Für alle Bewegungen habe ich dann entsprechend Batch- und XML-Dateien erstellt. Die Knöpfe rufen dann die entsprechende Batch-Dateien auf.

Wir brauchen ja auch ein Bild.

Der Videostream der Kamera lässt sich so anzeigen:

rtsp:*username:passwort@192.168.0.117/Streaming/Channels/101* ***Das ist das Bild in hoher Qualität und klappt z.B. mit VLC. In einem Webbrowser bekommt man auch ein einfaches Bild angezeigt: http://192.168.0.117/Streaming/Channels/102/httppreview Das hochauflösende Bild kann man z.B. mit der freien Software OBS aufzeichnen oder Streamen. Z.B. nach Youtube. So kann ich live von der Anlage auf Youtube senden. Man braucht auf jeden Fall auch das Webinterface der Kamera, um die Bildqualität, die Kompression usw. zu definieren. Das hängt natürlich auf davon ab, ob man streamen will***

und wie schnell die eigene Upload-Leitung ist. Auch Farbeinstellungen und andere Bildparameter lassen sich dort einstellen. Da hilft viel ausprobieren, bis es einem zusagt. LINK: https://www.andreas-graetz.de/modellbahn/automatische-webcam/