basysKom

QT QUICK ODER WEBTECHNOLOGIE?

ZWEI HMI-STACKS IM VERGLEICH

HMI-Forum – 11/2019 – Stuttgart

AGENDA

- Motivation und Kurzvorstellung
- Gegenüberstellung
- Zusammenfassung & Ausblick

MOTIVATION UND KURZVORSTELLUNG

Motivation

Für eine neue Produktgeneration wird für das HMI die Technologiefrage gestellt (Maßgabe Plattformunabhängig und Offen)

Wir als Dienstleister sind erstmal neutral

- Qt / Qt Quick
- Web-Technologien

Was ist für UNS die bessere Lösung? Nicht, was ist der bessere Stack

- in welchem Kontext findet die Entwicklung/der Einsatz statt?
- was gibt es an (nicht-technischen) Rahmenbedingungen?
- womit „fühlen“ wir uns besser?
- wo wollen wir uns hin entwickeln?

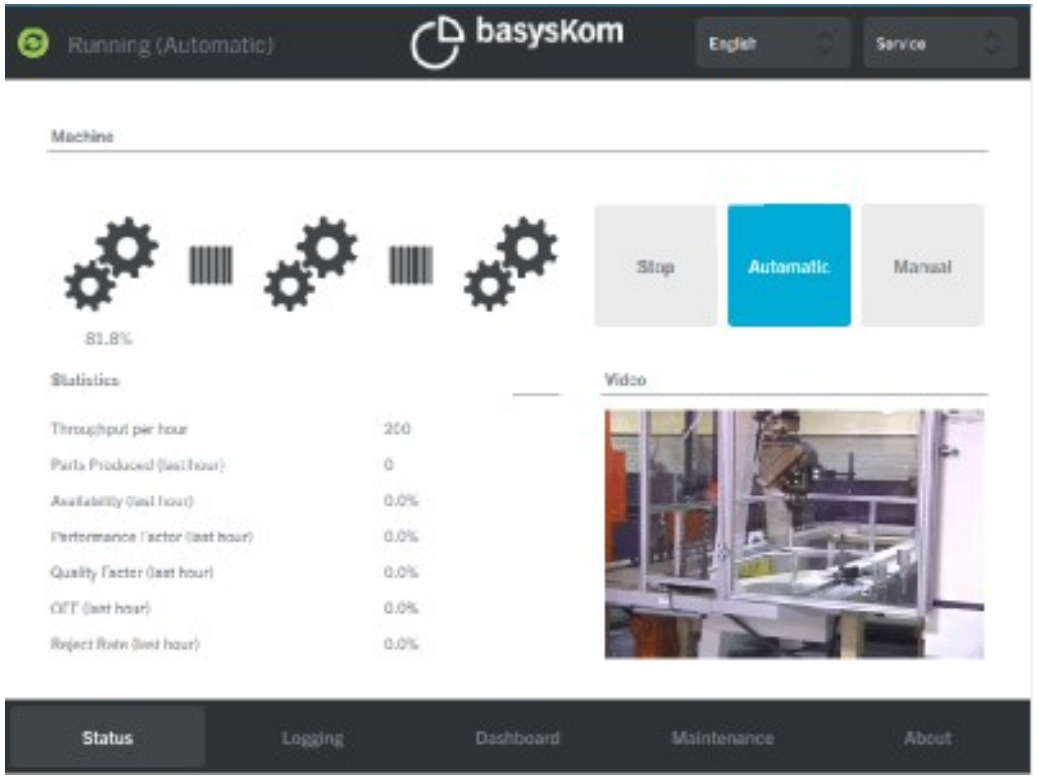
Kurzvorstellung Qt Quick

Qt ist ein Cross-Plattform C++ Framework für HMI-Entwicklung

Qt versucht das bessere C++ zu sein

Qt Quick ist Teil von Qt

- Anwendungsframework um schnell individuelle, animierte Touch HMIs zu entwickeln
- ~2010 zu Nokia-Zeiten konzipiert, deutlich gereift
- Daumenregel: wenn es nicht wie ein Desktop aussehen soll (flächig, animiert, touch)



Kurzvorstellung Qt Quick

Qt Quick besteht aus

- Sprache (QML, Declarativ) + Javascript
- Runtime
- SceneGraph (beschleinigtes Rendering)

Backend in C++ (über das Meta-Object-System gekoppelt)

- Qt Quick Anwendungen bauen auf Qt auf (+ beliebige C/C++ Bibliotheken)
- Typischerweise befinden sich Frontend und Backend im selben Prozess

```
Image {
    id: lowerLeftGear

    height: Constants.space50 * container.scalingFactor
    width: height
    fillMode: Image.PreserveAspectFit

    anchors.left: parent.left
    anchors.bottom: parent.bottom
    source: "qrc:/assets/images/gear01.png"
    sourceSize.width: sourceSize.width * scalingFactor
    mipmap: true
}
```

Kurzvorstellung Web-Technologie

Was ist überhaupt „Web-Technologie“?

- HTML5 + CSS + Javascript im Browser
- Client/Server-Model
- Kommunikation mit dem Server per HTTP/WebSockets

Und das Backend?

- Wir benötigen noch „etwas“ für die Server-Seite
- Womit ist die Server-Seite umgesetzt?
- Ist das „auch“ Web-Technologie?

Was bedeutet „Javascript“?

- From-Scratch?
- jQuery forever?
- $FrameworkX?

Exkurs Web-Technologie im Wandel der Zeit

Klassisch

- Server-Seitiges Rendering von HTML
- Request-Response getrieben
- PHP, Perl, Ruby, ...

Neo-Klassisch

- Server-Seitiges Rendering von Seiten
- Javascript lädt (partiell) dynamisch Inhalte nach („AJAX“)
- Nicht jede Interaktion mit dem Server hat einen Reload zur Folge

Modern

- SPA – Single Page Application
- Server liefert initial die Anwendung (HTML/CSS/Javascript/Assets als Bundle) und stellt APIs zur Verfügung (Anwendungsserver)
- Anwendung ist in JS geschrieben, Rendering im Client
- Fühlt sich an wie eine Anwendung, nicht wie eine Web-Seite

Web-Technologie im Kontext HMI

Klassisch bis Modern

- keine Wertung!
- im Kontext (Touch-)HMI ist SPA meist die beste Lösung

Zurück zum Backend...

- HTTP-Server der das App-Bundle ausliefert
- Application-Server der per REST und/oder WebSocket API ansprechbar ist

Mehr

- https://blog.basyskom.com/2016/cross-platform-application-development-for-desktop-mobile-and-embedded-with-modern-web-technology/

Technologie für das Backend?

Node.js

- nicht per-se Web-Technologie, aber „gehört dazu“
- Großes Ökosystem
- Gedanke: Einsatz von Javascript/Typescript im Frontend/Backend
- kann Sinn machen, wenn die Verantwortung für HMI (Frontend/Backend bei einer Gruppe liegen soll)
- Einbindung von nativen Bibliotheken oder Dienste-Schnittstellen wie zb. D-Bus möglich
- Nativer Ansatz ist u.U. schlanker

C++ oder Qt oder ...

- Gedanke: Einsatz von Web-Technologie soll auf das Frontend beschränkt bleiben
- Die Gruppe die den Anwendungskern betreut, betreut auch das Backend („alles was auf dem Gerät läuft“)
- kein großes etabliertes Ökosystem (mit Web-Bezug), (nötig?).

Welches Javascript-Framework?

SPA braucht ein geeignetes Framework

- die Anwendung wird primär darin geschrieben

Angular vs. React vs. . . .?

- In den letzten Jahren die beiden populärsten Frameworks
- Äpfel und Birnen

Angular ist ein vollständiges Framework

- Komplex
- Einstiegskurve ist steiler

React ist ein UI-Framework, muss mit weiteren Bibliotheken kombiniert werden

- Einfacher im Einstieg, braucht später ggfls. mehr Erfahrung
- Mehr Freiheit, weniger Struktur

Unsere Erfahrung

- die höhere Einstiegshürde von Angular amortisiert sich
- ein mehr an Struktur und Vorgaben ist hilfreich

GEGENÜBERSTELLUNG

Tooling?

Qt / Qt Quick

- Qt Creator als moderne IDE (nicht zwingend)
- Diverse C++-Toolchains
- qmake (zukünftig CMake)
- Paketverwaltungen sind erst im kommen (mit Qt weniger wichtig?)

Web-Technologie

- Auswahl an modernen IDEs (VS Code, Atom, WebStorm)
- Build-System(!)
- Debugger im jeweiligen Browser (Dev-Tools)
- Paketverwaltung (zB. npm)

Layouting?

Qt / Qt Quick

- Layouting ist Screen-orientiert („platzieren von Widgets“)
- Inhalte werden nicht gescrollt
- Unterstützt auflösungsunabhängige Layouts
- Support für High-DPI Scaling

Web-Technologie

- Per CSS
- Traditionell Dokument-orientiert („füllen einer dynamischen Seite mit Inhalt“)
- mit modernen CSS-Mitteln einfacher
- Unterstützt auflösungsunabhängige Layouts
- Sehr gute Unterstützung für DPI-Unterschiede (ein px ist kein Geräte Pixel)

Styling?

Qt / Qt Quick

- Styling wird in QML beschrieben
- Ausgliedern von Styling-Informationen in zb. QML-Singletons, Property-Bindings

Web-Technologie

- Trennung von Layout, Inhalt und Styling
- Styling per CSS
- Sehr flexibel/modular

Lizenzierung

Qt / Qt Quick

- Dual / Tripple Licensed (GPL / LGPL / Commercial)
- Copy-left bringt einige Verpflichtungen / Dinge die zu beachten sind

Web-Technologie

- Sowohl im Node/NPM-Umfeld als auch bei den SPA-Frameworks sind „Permissive Lizenzen“ die Regel (MIT, BSD, Apache)
- Weniger Verpflichtungen (nicht keine!)
- Es schadet nicht (automatisiert) die Lizenzen im Auge zu behalten

Gedanke: ggfls. kein neues Thema wenn man Embedded-Linux einsetzt?

Langlebigkeit?

Qt / Qt Quick

- Gutes „Führungszeugnis“
- API werden innerhalb eines Major-Release beibehalten
- Lebensspanne => ~10 Jahre
- Gewisse Stabilität auch über Major-Releases hinweg

Web-Technologie

- Es kommt drauf an…
- Kern-Webtechnologie ist recht stabil (DOM, Browser-APIs)
- Bei den SPA-Frameworks gab es eine Weile sehr viel Reibung (wird besser)

Hardware-Anforderungen?

Primär für Embedded-HMIs spannend

Qt / Qt Quick

- Komplexe Anwendungen auch mit i.MX6 SoloX und ähnlichen SOCs
- mit oder ohne GPU
- Anforderungen werden tendenziell mit Qt6 / QML3 nochmal deutlich geringer

Web-Technologie

- Embedded-HMI: i.MX6-DualLite ist grenzwertig
- Fortlaufende Entwicklung auf dem Gerät ist wichtig => die Webentwickler benötigen jeder(!) ein Target.

Cross-Plattform?

Qt / Qt Quick

- Desktop/Mobile/Embedded
- Desktop als „der Ursprung“
- Starker Fokus auf Embedded in den letzten Jahren
- Windows, OSX, Linux, Android, iOS, Embedded-Linux, …
- Diverse Toolchains

Und im Browser?

Web-Technologie

- Überall da wo ein Browser zur Verfügung steht (rein HMI)
- Apps per Cordova
- Desktop-Anwendungen per Electron

Cross-Browser-Testing ist immer noch nötig (und ein Schmerz)

ZUSAMMENFASSUNG & AUSBLICK

Zusammenfassung und Ausblick

Qt Quick als auch „Web-Technologien“ sind die beiden dominanten Vertreter für HMIs mit Maßgabe – offen & plattformunabhängig

Kunden stellen für neue Produkte oft die Technologie-Frage – was ist besser?

- muss jeweils im Kontext betrachtet werden
- Web-Technologien sind im Embedded-Kontext oft schwierig – Qt ist dort wiederum stark
- Qt ist schwierig, wenn auch eine Browser-basierte HMI benötigt wird

Welche Aspekte wurden im Vortrag nicht betrachtet?

- Besuchen Sie uns an unserem Stand

DANKE SCHÖN!

FRAGEN?