basysKom

Qt and the web – where are we today?

Embedded World Conference 2020

basysKom

About us

basysKom

- Is a Qt Service Partner since 2004
- Is located in Darmstadt and Nürnberg
- Is employing ~30 people
- Is part of the UX Gruppe
- Provides services (consulting, training, coaching and development) around Qt and Web-based HMIs
- Focuses on technical/industrial applications (HMI and application development)

Myself

- Development Lead @ basyskom

basysKom

Introduction

The web is now around 30 years old

Hypertext, HTML, HTTP(S), web browser, web server, (...) are omnipresent

The web with its immense success/massif adoption had a strong influence on technology

For a lot of use-cases web-technologies are the default

basysKom

Is the web the Holy Grail?

Problems with web technologies

- Strongly fragmented ecosystem
- Lots of churn (short-lived APIs, fashion-driven)
- Offline is a special case
- Limited use on (low-end) embedded hardware
    - System integration can be a mess
    - Performance issues

basysKom

It's a web's world

So even in 2019 there are good reasons to develop native applications

Still a native framework has to prove itself capable in a „web world“

- Consuming web services (Qt as a client)
- Providing web services (Qt as a server)
- Rendering web content with Qt
- Getting Qt into the browser

basysKom

Qt as a client

basysKom

Use-Cases: Qt as a client

Task:

- Interface web-focused services

Examples

- Upload/download of files
- Interacting with REST- and/or WebSocket-APIs

Things to consider

- Serialisation of API data
- Transport security
- Authentication/authorization

basysKom

HTTP(s)-client

QNetworkAcccessManager

- C++ API
- Part of QtNetwork
- A bit of an abstract name for a HttpClient? ;-)
- API can feel a bit dated when coming from more modern asynchronous environments
  - In Qt since 4.4

XmlHttpRequest

- JS-API that can be used from QML
- Mostly analog to the browser API of the same name
- Can be used to pull data into a QML-only/-focus Qt Quick application

Both support SSL/TLS (via OpenSSL)

basysKom

WebSocket-client

WebSockets == bidirectional connection between client and server

- Enables the server to push events into the client (browser)

WebSockets in Qt

- QWebSocket provides a client
- Part of the QWebSockets module
- Provides a C++ and QML API
- Supports TLS (wss://)

basysKom

Serialization/Deserialization

JSON

- Typical format used on Web-APIs
- Supported through QJsonDocument and friends
- Part of QtCore
- No JSON-Schema support (there are add-ons that work with QJsonDocument though)

XML

- Old-school, but still around

Various options in Qt

- QXMLStream{Reader|Writer} in Qt Core
- QDomDocument in Qt XML
- Qt XML Patterns for a validating parser

basysKom

Authentication/Authorization

Clients need to authenticate itself against (web) services

- Ideally without giving the client the username/password
- Ideally without giving the client wholesales access

Token-based authentication/authorization

Supported in Qt through the QNetworkAuthorization module

- OAuth1, 2

basysKom

Qt as a server

basysKom

Use-Case: Qt as a server

Task

- Provide REST and/or WebSocket-APIs with Qt

Examples

- Interfaces for test-automation
- Headless Qt (Qt as a friendly/better C++ to implement an embedded firmware)
- Integration point for other applications (e.g. order import)
- Integration of legacy applications

basysKom

Use-Case: Qt as a server

Solution: QHttpServer!

- A Qt component that has been missed by a lot of people
- There are quite a few Qt-ish HttpServer projects on github, none that has critical mass is properly maintained

Now there is an „official“ take by The Qt Company

- https://code.qt.io/cgit/qt-labs/qthttpserver.git/
- Currently still a labs project!
- GPL + commercial licensing

Solution: QWebSocketServer

- Part of the Qt WebSocket module
- Available since Qt 5.4
- C++ and QML bindings

basysKom

Rendering web content with Qt

basysKom

Use-case: embedding web content

Task

- Existing web content needs to be rendered within a Qt application
- Examples
  - generic web browser
  - walled gardens
  - games
  - reports
  - output of authoring tools that produce web content

Solution

- QWebEngine
  - Embeddable browser component based on Chromium
  - QML & C++ APIs
  - Complex API – enables complex applications
- QWebView
  - Wraps a native browser component on Android/iOS
  - Falls back to QWebEngine on other platforms
  - QML only
  - Simple API – high level integration

basysKom

Use-case: Web-HMI & Qt/C++ backend

Task

- A mixed mode application where the frontend is done with web-technologies and the backend with Qt/C++
- Deeper integration compared to a classic client/server scenario

Motivation

- Performance
- Legacy code (aka. „valuable business logic“)

Solution

- QWebEngine to „host“ the HMI
- QWebChannel to enable access from browser-side JS to QObjects
- See: doc.qt.io/qt-5/qwebchannel.html

basysKom

Use-cases: embedded web browser

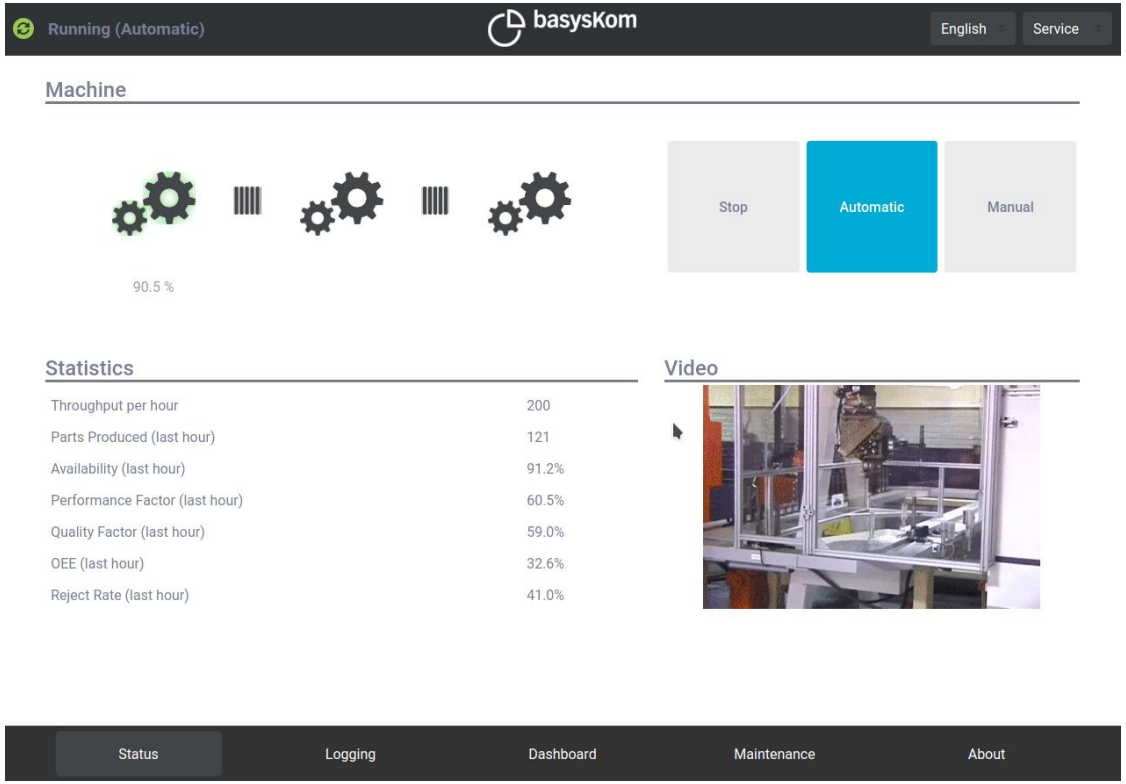
Task

- Web browser for an embedded system
- Full-screen, no Window-Manager
- Not actually a „Qt application“, just taking advantage of the pre-existing system integration work

Solution: Qt WebEngine (again)

Caveats

- System-Integration can be tricky, large
- Requires more powerful hardware (compared to Qt Quick)



basysKom

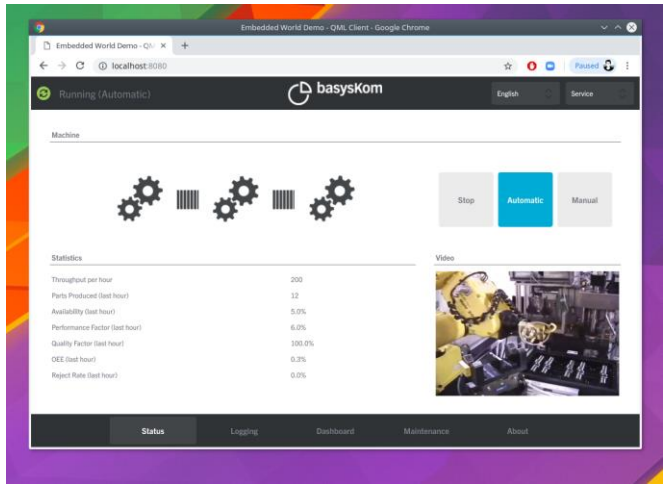
Getting Qt into the browser

basysKom

Use-case: the browser as a remote display

Task

- An HMI/application that is written in Qt should render into a browser
  - Device without a display
  - Remote access to a device (which has a display)

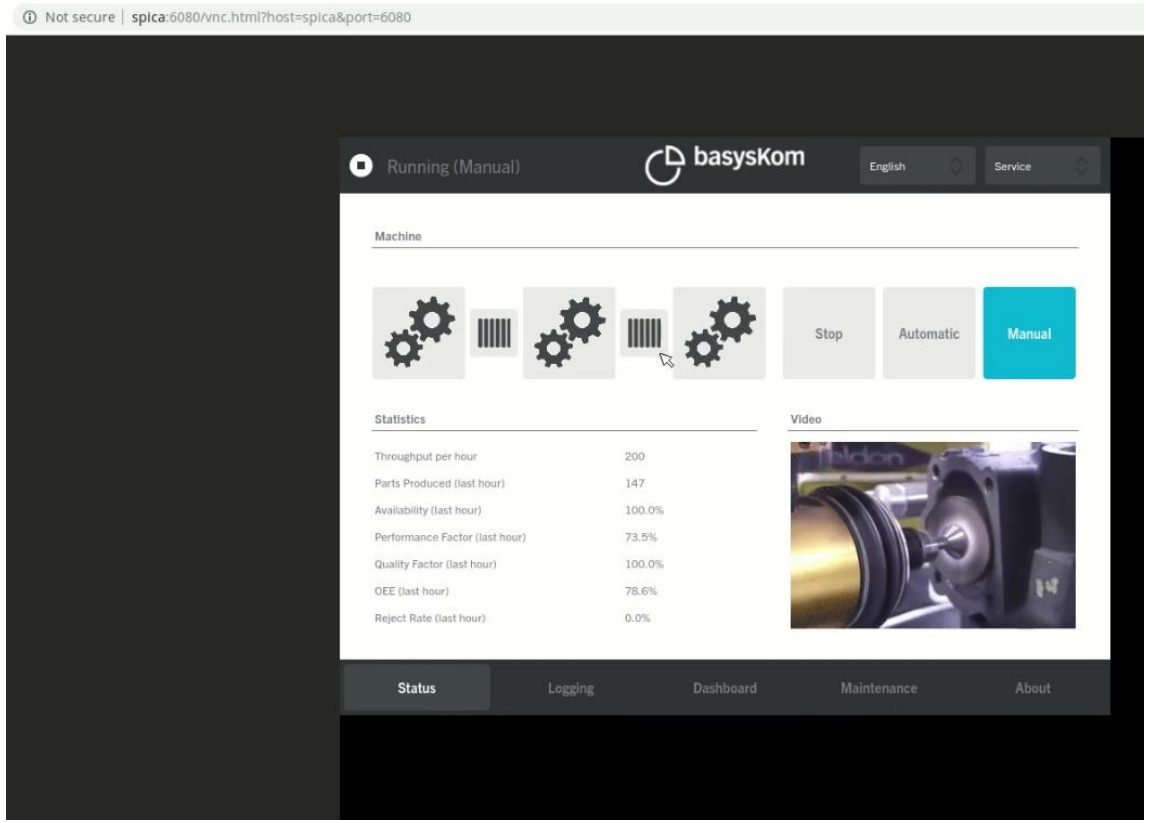


Solution #1: WebGL streaming

- A Qt QPA that is streaming webgl commands
  - myApp -platform webgl
- Qt Quick only
- GPL + Commercial licensing
- Resolution independent
- Latency is critical
- One user at a time
- Example: https://www.qt.io/bosch-built-with-qt

basysKom

Use-case: the browser as a remote display



Solution #2: VNC

- VNC QPA + Websocket-Proxy + NoVNC (JS-Client)
- QWidget & Qt Quick (when using the sw-renderer)
- Is sending bitmaps
- Resolution is fixed!
- Latency is critical
- Also one user at a time

basysKom

Use-case: running Qt in the browser

Task

- An application written in Qt needs to run inside an unmodified web browser
- Nothing is to be installed

Examples

- Headless devices
- Inhouse software distribution

Excursion: WebAssembly

- Portable runtime (byte code, APIs)
- Implemented by all major browsers
- Security wise the runtime has the same limitations as a regular website

basysKom

Use-case: running Qt in the browser

Solution: Qt for WebAssembly

- Official platform with Qt 5.13
- www.qt.io/qt-examples-for-webassembly

Relatively large downloads (~20MB for a Qt Quick application)

Link time is slow…

Application needs to deal with browser limitations

- File access
- Network access

basysKom

Wrap up

basysKom

Wrap up

Four perspectives on Qt and the web

- Client
- Server
- Rendering
- Qt in the browser

More?

- ADL, Swagger, openAPI
- Templating, HTML generation
- Complete WebApplications in Qt?
- …

basysKom

Questions?

Come talk to us about your „Qt and the web“ projects

You can find us at in Hall 4, booth 258

Frank Meerkötter

Development Lead

frank.meerkoetter@basyskom.com