

The background of the slide is a photograph of an industrial factory floor. A large, grey robotic arm is the central focus, positioned on the left side. It has a complex jointed structure with various cables and hoses attached. The arm is reaching towards the center of the frame. In the background, there are various industrial structures, including metal frames, pipes, and other machinery. The lighting is bright, typical of a factory environment. A large, semi-transparent yellow banner covers the right side of the image, containing the main title and subtitle in dark blue text.

# **BUILDING MODERN INDUSTRIAL APPLICATIONS WITH OPEN STANDARDS AND OPEN SOURCE SOFTWARE**

**Embedded World Conference, 2018**

# Contents

- Introduction
- Traditional HMI Software
- Modern HMI Software Development
- Conclusion





# INTRODUCTION



# Who are we?

## basysKom

- Located in Darmstadt & Nürnberg
- Software Engineering Services (Consulting, Training, Coaching & Development)

## Myself

- Development Lead
- [frank.meerkoetter@basyskom.com](mailto:frank.meerkoetter@basyskom.com)

## Why are we here today?

Software Engineering Services (Consulting, Training, Coaching & Development)

Focused on industrial applications



A great deal of experience with  
Application/HMI development (Qt & HTML5)  
and connectivity (OPC UA/MQTT/REST)

# What is this talk about?

HMI for industrial applications are often built with proprietary solutions

Showcase for an industrial HMI built with

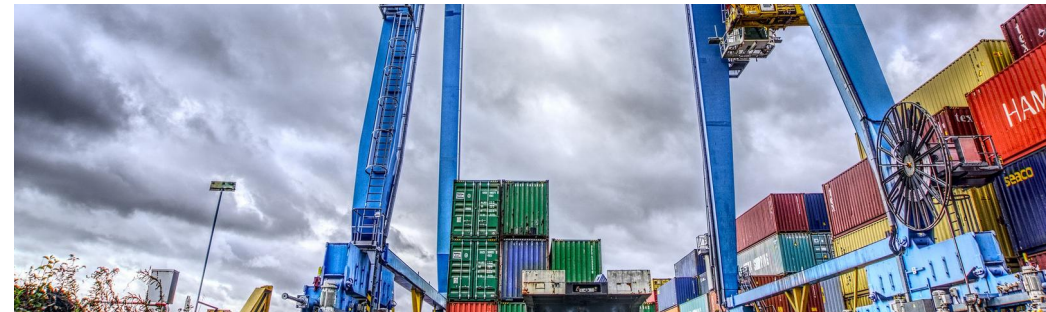
- ➡ Open Standards
- ➡ Open-Source Software

## Target audience



### Machine manufacturers

- Medium to large series
- The same or nearly the same application/HMI is shipped with the machine
- Complex, large applications



### Not plant manufacturers / industrial system integrators

- An HMI is created (configured) for a specific, individual installation
- Often a limited budget for an HMI
- Well served by HMI tools



# TRADITIONAL HMI SOFTWARE





# Overview

## HMI Software (specific for Industrial applications)

- Graphical editor
- Runtime

## Communication driver

- Proprietary
- OPC classic, OPC UA

### Goal:

- ➡ “configuration, not programming”
- ➡ Engineering focused

# Pros

## Many application specific features (on top of the core HMI functionality)

- Recipe-Management
- Alarms
- Import of symbols/machine variables
- Logging/Historical data

## Quick results (for the standard case)

- No deep software development skills needed

## Cons

**Hard to create high quality HMIs**

**Becomes a burden for large/complex applications**

**Hard to scale over a range of machines**

- Version control can be tricky

**Often only a limited number of platforms for the runtime**

- Which limits the choice of hardware
- This is becoming more flexible

**Licenses**

- Windows, HMI tool / runtime & communication driver(?)

**Lock-in**

**Dependencies on the product life cycle of others**

- Windows, communication driver & HMI software



## Upshot

- ➔ Works as long as one stays within the “comfort zone” of that tool
- ➔ Good choice for one-offs and pure automation projects

# MODERN HMI SOFTWARE DEVELOPMENT





# Goals for our showcase

## Be flexible

- Have the option to create something special/a real application

## Reduce hardware costs

- Be able to use a cheap ARM device for the HMI

## Reduce license costs

## Future proof

- Reduce the dependency on other peoples product life cycles





## An example stack

Qt

- QML/Qt Quick
- C++

OPC UA

- open62541
- Qt OpcUa

(Embedded)-Linux

- Actually cross platform

## Qt & Qt Quick

**Cross platform C++ framework**

**Qt Quick**

**Dual licensed (FOSS and Commercial)**

**Vast set of modules**

- SQL, Printing, XML, Networking, HTTP, CAN, ModBus, ...

**Long term commitments on API and ABI stability**

## OPC UA & Qt OpcUa

Communication standard for industrial applications

Platform independent (unlike OPC Classic)

open62541

Qt OpcUa is a new Qt module developed by basysKom

- Qt-API based on existing OPC UA stacks



## Pros

**Allows to build high quality HMIs**

**Scalable across**

- machine variants
- application complexity

**Flexibility to implement individual requirements**

**Cross platform**

**No vendor lock-in**

**Opportunity to reduce license fees and hardware cost**

## Cons

### **Less guidance by an industry specific tool**

- Less predefined structure
- Less pre-packaged functionality (no recipe management, no alarm-management, ...)

### **Requires a different skill set (software development focused)**

### **Does not scale for one-off scenarios**

# CONCLUSION



# Conclusion

## **Working with open standards and open source software shines when**

- creating large and complex applications no longer fitting into the structures provided by traditional HMI software
- trying to reduce license fees
- future proofing a product



**THANK YOU!**

**QUESTIONS?**

**Frank Meerkötter**  
Development Lead  
[frank.meerkoetter@basyskom.com](mailto:frank.meerkoetter@basyskom.com)

