Patrik Pfaffenbauer

Portfolio

patrik.pfaffenbauer@p3-software.eu | +43 660 49 08 028 | Software Engineer
Untere Dorfstraße 14, 4622 Eggendorf | ATU73971525

Patrik Pfaffenbauer
31.01.2019

Contents

BeKa-Software GmbH

Entwickelt in Pasching/Wien Individualsoftware für verschiedenste Kunden. (www.beka-software.at)

SystemTera

Entsandt aus einem Projekt für die Firma OÖ. Gas-Wärme GmbH. Hierbei geht es um die Überwachung von rund 500 Heizwerken für Mehrparteien-Häuser, Gewerbeparks, Hotels und Industriebetriebe im Raum Oberösterreich. Anschließend wurde aus dem Projekt ein Produkt. (www.systemtera.com)

Server

Dazu wurde ein Embedded Linux Controller entwickelt, dieser nutzt ein Gentoo Linux als Betriebssystem. Die Anwendung wurde mit C++ und Qt entwickelt. Der Controller sammelte die Daten über verschiedenste Bussystem (KNX, ModBus, MBus,…) und übertragt diese in die Cloud.

Cloud

Die Cloud Anwendung verwaltet die Server im Feld und speichert die Daten in der Datenbank mittels Entity Framework. Diese Anwendung wurde mit C# .NET entwickelt. Die Kommunikation zwischen Server & Cloud wurde mit SignalR realisiert um auch Echtzeitdaten abfragen zu können. Hierzu habe ich einen SignalR Client in C++ als Open Source Software entwickelt (https://github.com/p3root/signalr-qt). Die Server im Feld wird über den Manager verwaltet.

Manager

Der Manager ist die Verwaltungsoberfläche für das ganze System. Hier konnten neue Server angelegt werden und die Daten der Server Visualisiert werden. Die Applikation wurde in WPF implementiert. Die Kommunikation zwischen Manager & Cloud wird über WCF & REST aufgebaut. Die Kommunikation zum Server wurde mit SignalR realisiert. Es wurden auch verschiedenste Reports über die Effizienz der Anlagen erstellt.

Aufgaben/Technologien

- C# .NET
  - WCF
  - WPF
  - REST
  - SignalR
  - Entity Framework
- C++
  - Qt 4.8.2
  - SignalR
- Embedded Linux
  - Gentoo
- SVN

Verbund Stromzähler

Es wurde gemeinsam mit der Firma Verbund ein Stromzähler für das **Verbund Eco-Home** Produkt entwickelt. (https://www.verbund.com/de-at/privatkunden/smart-home/eco-home-zusatzgeraete/verbund-strommessmodul)

Dieses Projekt läuft auf einem Atmel SAM Prozessor und wurde mit C/C++ implementiert. Herausforderung hier war klarerweise die Stabilität und Zählergenauigkeit. Es gibt hier keine Möglichkeit den Stromzähler über das Internet upzudaten.

Der Stromzähler bittet drei Schnittstellen

- ModBus
  - Um auch anderen Anbietern die Möglichkeit zu geben den Stromzähler auszulesen.
- PLC (Power Line Communication)
  - Um mit dem Eco-Home Gateway zu kommunizieren
- USB
  - Um den Stromzähler zu kalibrieren, und die ModBus Einstellungen vorzunehmen

Aufgaben/Technologien

- Embedded
- C/C++
- Git & Bitbucket

NETxAutomation Software GmbH

Zählt du den führenden Anbietern für Software in der Gebäudeautomatisierung.
(www.netxautomation.com)

Web Manager

Ist die Konfigurationsoberfläche wo Kunden das Gebäude überwachen und konfigurieren können. Hierzu zählt das Alarming, Scheduling, Trending & Visualisierung. Diese Anwendung wurde vor 2 Jahren unter meiner Hand von Grund auf mit Angular 2 neu entwickelt. Inzwischen wird hier Angular 7 genutzt. Die Kommunikation erfolgt über REST & SignalR. Das Backend ist in C# entwickelt und nutzt für die Datenübertragung zum Server ein COM Interface.

Server

Ist das Herzstück der Produktpalette, dieser Sammelt die Daten von verschiedenen Bussystemen und verteilt diese je nach Konfiguration. Der Server ist in C++ implementiert.

Plugins

Der Server kann mittels einem Plugin-System erweitert werden. Hierzu gibt es eine C++ und eine .NET Schnittstelle. Neue Treiber werden jedoch nur mehr in .NET geschrieben.

Hierzu habe ich einige „Treiber“ entwickelt.

- LaMPs
  - Um verschiedene DALI KNX Gateways auf einen Nenner zu bringen
- KNX IP Secure
  - Kümmert sich um den gesicherten Aufbau einer IP Verbindung zu einen IP Secure Gateway/Router
- MBus UDP
  - MBus Protokoll über ein UDP-MBus Gateway

Vision

NETxVision heißt die Mobile App welche für die Visualisierung benutzt wird. Diese wurde mittels Xamarin entwickelt.

Aufgaben/Technologien

- Teamleiter der Softwareentwicklung
- C# .NET >= 4
- C# .NET Core >= 2.1 (für das Reporting)
- C++
- Angular Typescript
- Git / Bitbucket
- Xamarin

P3bble

Ist ein Framework um mit der Pebble Smartwatch zu kommunizieren. Diese wurde für Windows Phone 8 entwickelt (C# .NET). Ich habe die Arbeit aber dann eingestellt da Microsoft bekanntlich Windows Phone ebenso eingestellt hat. (https://github.com/p3root/p3bble)

Automatica.Core

Automatica ist ein selbst entwickeltes Automatisierungssystem. Der Server wurde mit .NET Core entwickelt und kann somit auf Windows, Linux und Mac ausgeführt werden. Die Konfigurationsoberfläche wurde mit Angular entwickelt und wird auf dem im Server betriebenen Webserver ausgeliefert. (https://www.automaticacore.com, https://docu.automaticacore.com)

Server

Der Server ist das zentrale Stück, dieser kümmert sich um die Schnittstellen nach außen (Web API, REST & SignalR für Realtime Date) und lädt die verschiedenen Plugins. Dieser verarbeitet auch die Daten von den Plugins (Treibern und Logiken) und gibt diese an das nächste Plugin, je nach Konfiguration weiter.

Visualisierung

Es kann auch am Server eine Visualisierung genutzt werden. Hierzu gibt es vorgefertigte Blöcke welche in verschiedenen Seiten genutzt werden können. Die Visualisierung kann aber auch anhand der Konfiguration generiert werden.

Treiber

Ich hab bereits eine Menge Treiber implementiert, darunter:

- KNX IP (+ IP Secure)
- MBus UDP
- ModBus TCP / RTU
- Automatica Remote
  - Um andere Automatica Instanzen zu konfigurieren (Master/Slave)
- Konstanten
- EnOcean
- Fronius Symo Wechselrichter (Solar API)
- Amazon Alexa, Logitech Harmony
  - Hier simuliere ich eine Hue Bridge und kann somit von verschiedenen Produkten erkannt und verwendet werden
- Apple HomeKit
  - Um das iPad, iPhone oder auch Apple TV für die Visualisierung zu verwenden
- Ikea Tradfri
- Loxone Miniserver (WebSocket API)
- OpenWeatherMap
  - Für die Abfrage von Ortspezifischen Wetterdaten
- Times/Sun
  - Stellt aktuelle Uhrzeit bzw. Datum als Datenpunk bereit
  - Stellt auch den Sonnenstand berechnet anhand der Geo Location zur Verfügung
- Wake on Lan
- ZWave (Work in Progress)
- ZigBee (Work in Progress)

Treiber werden in .NET Core entwickelt und über die Automatica.CLI paketiert und in die Cloud hochgeladen wo der Benutzer am Server diese dann installieren kann.

Logiken

Hier habe ich eine Basis Logiken bereits implementiert wie:

- Messenger
  - Um Emails zu senden
  - Später sollen auch SMS und Voice Notifikationen eingebaut werden
- Compare
  - Bigger, Equals,…
- Logic
  - And, Or,…
- Math
  - Addition,…
- Time
  - Zeitschaltuhr,…

Cloud

Die Cloud Anwendung ist ebenso in .NET Core entwickelt und wird in Azure gehostet. Diese Anwendung dient derzeit nur für die Lizenzverwaltung (war ein Proof of Concept), die Plugin Verwaltung und die Verwaltung der Updates. Hierzu gibt es auch eine Managementoberfläche welche in Angular implementiert ist.

CLI

Die CLI (Command Line Interface) ist ein kleines Hilfswerkzeug für die Entwicklung von Plugins. Diese kann z.B. ein Plugin Boiler Plate generieren wo alle Abhängigkeiten und Basis Implementierungen bereits vorhanden sind. Mit dieser CLI kann auch ein Plugin gebaut und in die Cloud deployed werden.

CI/CD

Das Build-System liegt hier bei Azure DevOps. Es gibt verschiedenste Build/Release – Pipelines. Wichtig war mir hier eine gute Testabdeckung, ein Fehler in der Core macht das System unbrauchbar. Daher habe ich eine Testabdeckung von >= 70%.

Plugins

Jedes Plugin hat eine eigene Pipeline, welche aber von der Automatica.CLI beim Anlegen des Projektes generiert wird. Nach jedem Build wird automatisch das Paketierte Plugin in die Dev-Cloud übertragen.

Core

Hier wird das Core System + Web gebaut und getestet. Mittels einer Release Pipeline wird das Update in die Cloud übertragen.

Raspberry PI Image

Es wird auch ein Image für den Raspberry PI in Azure DevOps generiert welches nur mehr auf eine SD Karte gespielt werden muss und der Anwender ist Ready-to-Go.

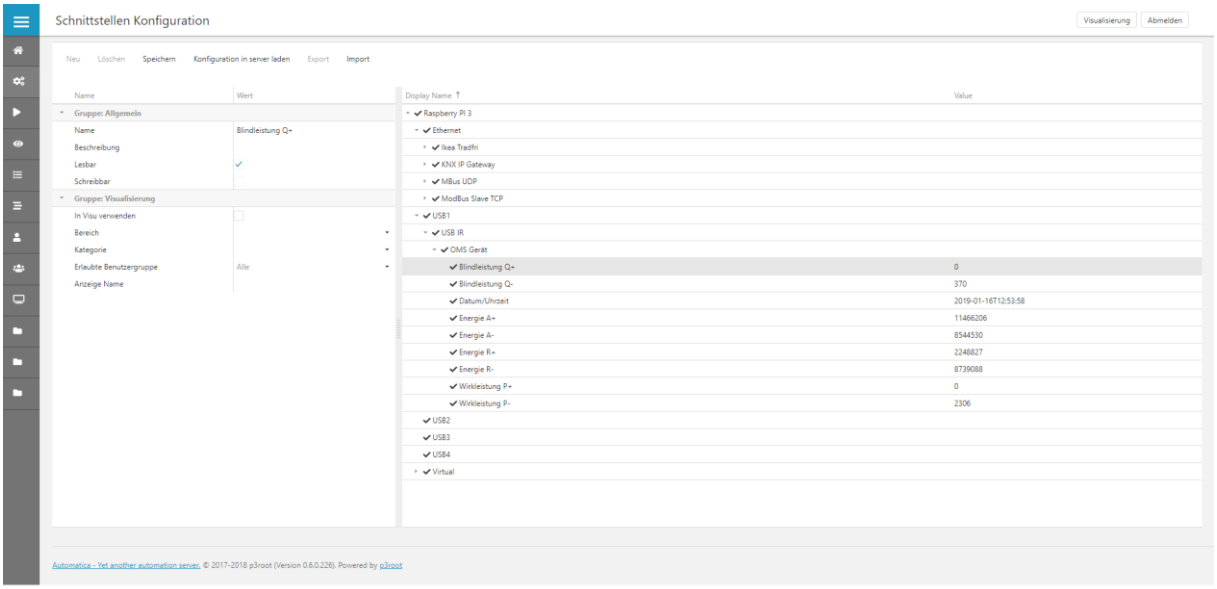
Aufgaben/Technologien

Hier wurde Design, Architektur und Implementierung vor mir vorgenommen.

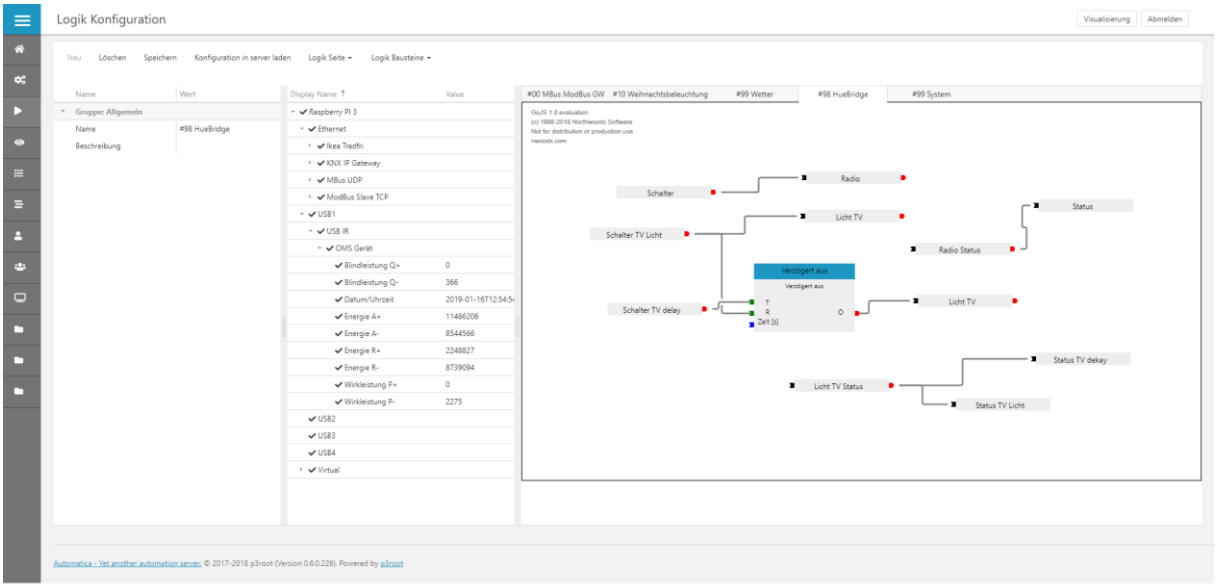
- C# .NET Core
  - Entity Framework Core
  - SignalR
  - Web API (REST)
  - Message Pack
- Angular
  - DevExtreme UI Components
  - Message Pack
- Embedded Linux
  - Raspberry PI (Debian)
- DocFX
  - Für die Code Dokumentation
- Azure Cloud
- Azure DevOps
- GitHub
- Git
- Im Projekt genutzte Verschlüsselungen/Passwort Transfer/Hash Algorithmen
  - Secure Remote Passwort
  - HDKF-SHA512
  - ED25519 Key Exchange
  - ChaCha20-Poly1305 AEAD
  - Curve25519
  - ED25519
  - AES128
  - PBKDF2

Screenshots

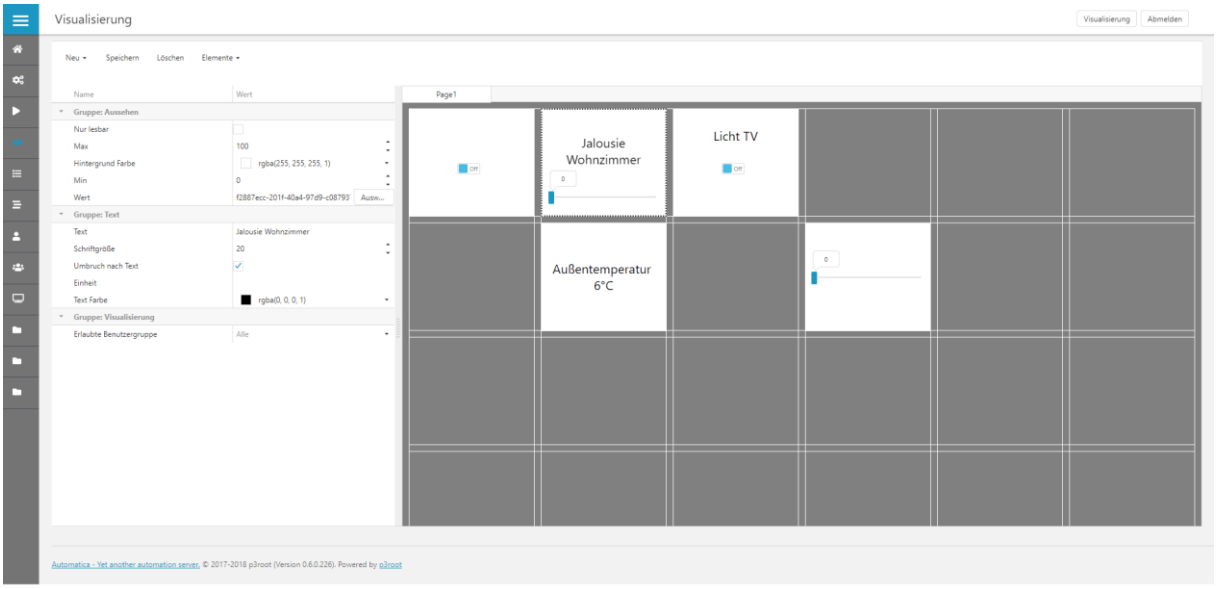
Schnittstellen Konfiguration



Logik Konfiguration



Visualisierung Konfiguration



Prolog

Das System wird bei mir im eigenen Haus bereits Produktiv genutzt.

Der Code für das System kann unter https://github.com/automatica-core & https://www.automaticacore.com eingesehen werden.