

Patrik Pfaffenbauer

Portfolio

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

Contents

About me	4
Quote	4
Technology Stack	4
Programming Languages	4
Technologies	4
Cloud	4
Concepts	4
Passion	4
Book recommendations	4
BeKa-Software GmbH	5
SystemTera	5
Server	5
Cloud	5
Manager	5
Tasks/Technologies	5
Verbund Stromzähler	6
Tasks/Technologies	6
NETxAutomation Software GmbH	7
Project time	7
Web Manager	7
Server	7
Plugins	7
Vision	7
Tasks/Technologies	7
AMV Networks GmbH	8
Project time	8
Autologg	8
Autologg Connected	8
Tasks/Technology	8
Arolla.io	9
Tasks/Technologies	9
P3bble	10

Automatica.Core	11
Project time	11
Server	11
Drivers	11
Logics	11
Cloud.....	12
CLI	12
CI/CD.....	12
Plugins	12
Raspberry PI Image.....	12
Tasks/Technologies	12
Screenshots	14
Schnittstellen Konfiguration.....	14
Logik Konfiguration	14
Prolog	14

About me

My name is Patrik Pfaffenbauer, I live in Upper Austria and work as a self-employed Software Engineer/Architect.

You can find some information about my technology stack, projects I have made and some more stuff. But let me start with my favorite quote.

Quote

If you think good architecture is expensive, try bad architecture.

Brian Foote & Joseph Yoder

Technology Stack

Programming Languages

- C#
- JavaScript
- C/C++
 - + Embedded micro controller
- Unix Bash

Technologies

- .NET Core
- Angular >= 2
- Qt >= 4.8
- Docker

Cloud

- Microsoft Azure
- Amazon Web Services

Concepts

- Clean Code
- SOLID principle

Passion

- Building automation aka smart home

Book recommendations

- **Clean Code/Architecture/Coder/Agile** by Robert C. Martin
- **Lean Startup** by Eric Ries

BeKa-Software GmbH

Develops individual software for different customers in Vienna and Pasching. (www.beka-software.at)

SystemTera

This project involves monitoring of around 500 heating plants for multi-party housed, commercial parks, hotels, and industrial companies in the Upper Austria area. The project then became a product. (www.systemtera.com)

Server

We developed a proprietary Linux embedded controller, which run on Gentoo Linux. The application was developed with C++ and Qt. The controller collected data via various bus systems (KNX, ModBus, MBus, ...) and transfers them to the "cloud".

Cloud

The cloud application manages the servers in the field and stores the data in the database using the Entity Framework. This application was developed with C # .NET. Communication between the server and the cloud was implemented with SignalR in order to be able to query real-time data. I developed a SignalR client in C ++ as open source software (<https://github.com/p3root/signalr-qt>). The servers in the field are managed by the manager.

Manager

The manager is the administrative interface for the entire system. Here new servers could be created and the server data visualized. The application was implemented in WPF. Communication between Manager & Cloud is established via WCF & REST. Communication with the server was realized with SignalR. Various reports on the efficiency of the systems were also created.

Tasks/Technologies

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

Verbund Stromzähler

Together with the Verbund company, an electricity meter was developed for the Verbund Eco-Home product. (<https://www.verbund.com/de-at/privatkunden/smart-home/eco-home-zusatzgeraete/verbund-strommessmodul>)

This project runs on an Atmel SAM processor and was implemented with C/C++. The challenge here was clearly stability and meter accuracy. There is no way to update the electricity meter over the Internet.

The meter provide 3 different interfaces:

- ModBus
 - To provide an open standard, so that 3rd party applications can read data from the meter.
- PLC (Power Line Communication)
 - For the Verbund Eco-Home system.
- USB
 - To calibrate and update the meter.

Tasks/Technologies

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

NETxAutomation Software GmbH

Is one of the leading providers of software in building automation. (www.netxautomation.com)

Project time

2016-12 – 2019-02

Web Manager

Is the configuration interface where customers can monitor and configure the building. This includes alarming, scheduling, trending and visualization. This application was developed from scratch with Angular2 **under my responsibilities**. Angular 7 is now used here. Communication takes place via REST & SignalR. The backend is developed in C and uses a COM interface for data transmission to the server.

Server

Is the **core** of the product range, this collects the data from different bus systems and distributes them depending on the configuration. The server is implemented in C++.

Plugins

The system can be expanded using a plug-in based system. This can be done via C++ and .NET. New drivers where only written in .Net.

For this I have developed some “drivers”:

- LaMPs
 - To bring different DALI KNX gateways to one common denominator
- KNX IP Secure
- MBus UDP

Vision

NETxVision is the name of the mobile app that is used for the visualization. This was developed using Xamarin.

Tasks/Technologies

- Teamleiter der Softwareentwicklung
- C# .NET >= 4
- C# .NET Core >= 2.1 (Reporting)
- C++
- Angular >= 2 + Typescript
- Git / Bitbucket
- Xamarin

AMV Networks GmbH

AMV Networks GmbH are working on solutions for the mobility of tomorrow. For this I have participated in 2 projects.

Project time

2019-02 - aktuell

Autologg

Autologg is a digital driver logbook, which also complies with the Austrian/German tax administrations. To do this, the customer must install the Autologg Box in their vehicle and can use it to log his tracks digital. However, since new vehicles are already delivered with onboard connectivity, install the box has become obsolete. The first step was to develop a digital logbook for Tesla owners. (www.autologg.com)

Autologg Connected

Tesla offers the possibility to read vehicle data in real-time. For this purpose, a .NET core application was developed that connects to Tesla and reads the vehicle data. As soon as a trip is recognized, it is transferred to the Autologg system.

Tasks/Technology

- Tesla Anbindung
 - AWS ECS
 - AWS DynamoDB
 - AWS Lambda
 - AWS API Gateway
 - Docker
 - .NET Core >= 2.0
- Erweiterung Frontend
 - Angular >= 2
- Erweiterung Autologg Backend
 - Java

Arolla.io

Arolla.io is an E-Scooter Sharing Platform. (www.arolla.io)

I was responsible for Backend and Frontend.

Tasks/Technologies

- Payment & Vouchers
 - React Native App
 - Heidelpay (Payment Provider)
 - AWS Lambda
 - NodeJS
 - Python
 - GraphQL
 - AWS Cloudformation

P3bble

Is a framework to communicate with the Pebble Smartwatch. This was developed for Windows Phone 8 (C # .NET). I stopped working because Microsoft is known to have stopped Windows Phone as well. (<https://github.com/p3root/p3bble>)

Automatica.Core

Automatica is a self-developed automation system. The server was developed with .NET Core and can therefore be run on Windows, Linux and Mac. The configuration interface was developed with Angular and is delivered on the web server operated in the server.

(<https://www.automaticacore.com>, <https://docu.automaticacore.com>)

Project time

2017-01 – aktuell

Server

The server is the central piece, this takes care of the interfaces to the outside (Web API, REST & SignalR for Realtime Data) and loads the various plugins. This also processes the data from the plugins (drivers and logics) and dispatches them on to the next plugin, depending on the configuration.

Visualization

The visualization is generated from the configuration properties.

Drivers

I already developed a wide set of drivers:

- KNX IP (+ IP Secure)
- MBus UDP
- ModBus TCP / RTU
- Consants
- EnOcean
- Fronius Symo Inverter (Solar API)
- Amazon Alexa, Logitech Harmony
- Apple HomeKit
- Ikea Tradfri
- Loxone Miniserver (WebSocket API)
- OpenWeatherMap
- Times/Sun
- Wake on Lan
- ZWave (Work in Progress)
- ZigBee (Work in Progress)

Logics

I already developed a wide range of logics:

- Messenger
 - Send emails
- Compare
 - Bigger, Equals,...
- Logic

- And, Or,...
- Math
 - Addition,...
- Time
 - Zeitschaltuhr, ...

Cloud

The cloud application is also developed in .NET Core and is hosted in Azure. This application is currently only used for license management (was a proof of concept), plug-in management and the management of updates. There is also a management interface for this, which is implemented in Angular.

CLI

The CLI (Command Line Interface) is a small auxiliary tool for the development of plugins. This can e.g. generate a plug-in boiler plate where all dependencies and basic implementations already exist. With this CLI, a plugin can also be built and deployed to the cloud.

CI/CD

We use Azure DevOps for build and release management. There are various build / release pipelines. Good test coverage was important to me here, an error in the core makes the system unusable. Therefore, I have a test coverage of $> 70\%$.

Plugins

Each plugin has its own pipeline, which is generated by Automatica.CLI when the project is created. After each build, the packaged plugin is automatically transferred to the dev cloud.

Raspberry PI Image

An image is also generated for the Raspberry PI in Azure DevOps which only needs to be copied on an SD card and the user is ready-to-go.

Tasks/Technologies

- C# .NET Core
 - Entity Framework Core
 - SignalR
 - Web API (REST)
- Angular ≥ 2
 - DevExtreme UI Components
- Embedded Linux
 - Raspberry PI (Debian)
- DocFX
 - Für die Code Dokumentation
- Azure Cloud

- Azure DevOps
- GitHub
- Git
- Crypto/Hashing
 - Secure Remote Password
 - HKDF-SHA512
 - ED25519 Key Exchange
 - ChaCha20-Poly1305 AEAD
 - Curve25519
 - ED25519
 - AES128
 - PBKDF2

Screenshots

Schnittstellen Konfiguration

The screenshot shows the 'Schnittstellen Konfiguration' window. It has a sidebar with icons for different functions. The main area contains a table with the following data:

Name	Wert	Display Name	T
Blindleistung Q+	0	Blindleistung Q+	0
Blindleistung Q-	370	Blindleistung Q-	370
Datum/Uhrzeit	2019-01-16T12:55:06	Datum/Uhrzeit	2019-01-16T12:55:06
Energie A+	11466206	Energie A+	11466206
Energie A-	8544530	Energie A-	8544530
Energie R+	2248827	Energie R+	2248827
Energie R-	8739088	Energie R-	8739088
Wirkleistung P+	0	Wirkleistung P+	0
Wirkleistung P-	2306	Wirkleistung P-	2306

Logik Konfiguration

The screenshot shows the 'Logik Konfiguration' window. It features a logic diagram on the right and a table of variables on the left. The table lists various variables and their values:

Name	Wert	Display Name	T
Blindleistung Q+	0	Blindleistung Q+	0
Blindleistung Q-	366	Blindleistung Q-	366
Datum/Uhrzeit	2019-01-16T12:54:55	Datum/Uhrzeit	2019-01-16T12:54:55
Energie A+	11466206	Energie A+	11466206
Energie A-	8544566	Energie A-	8544566
Energie R+	2248827	Energie R+	2248827
Energie R-	8739094	Energie R-	8739094
Wirkleistung P+	0	Wirkleistung P+	0
Wirkleistung P-	2275	Wirkleistung P-	2275

Prolog

Look here to get some more information.

<https://github.com/automatica-core>

<https://www.automaticacore.com>