

# Entwicklung eines Evaluierungswerkzeuges für barometrische Luftdruck-Sensoren

## Verteidigung der Bachelorarbeit

Benedikt Kleinmeier

Hochschule München

Fakultät für Informatik und Mathematik

23. März 2012



# Übersicht

- 1 Einführung
- 2 Die Komponenten des Evaluierungswerkzeuges
- 3 Der Softwareentwicklungsprozess
- 4 Live-Demonstration

# Einführung

## Thema

Entwicklung eines Evaluierungswerkzeuges für barometrische Luftdruck-Sensoren

## Themenwahl

- Praxissemester bei Infineon Technologies AG.
- Dabei zwei Projekte mit absoluten und relativen Drucksensoren bearbeitet.
- Projekt mit absoluten Drucksensoren als Werkstudent weitergeführt.
- Dieses Projekt ist Gegenstand der Bachelorarbeit.

# Barometrische Luftdruck-Sensoren

## Einsatzgebiet der Sensoren

U.a. im Automobilbereich.

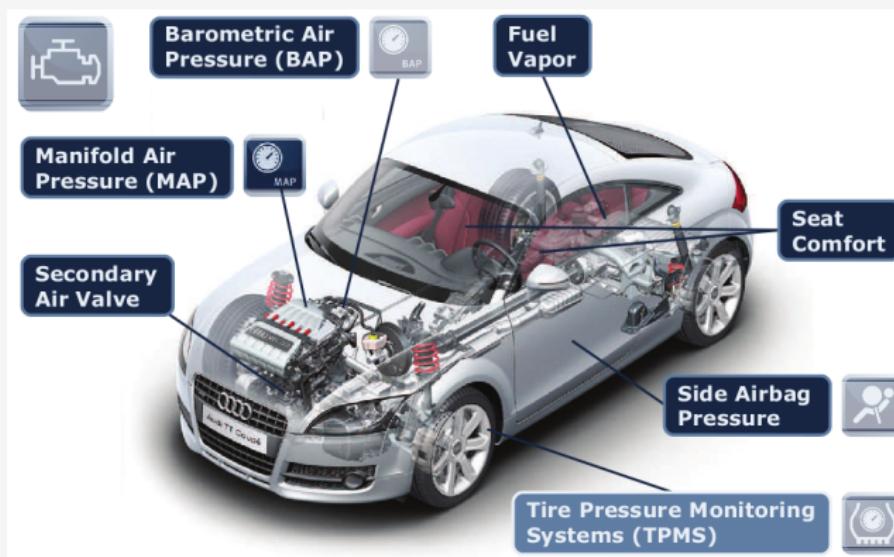


Abbildung: Infineons Drucksensoren

# Bedarf für Evaluierungswerkzeug

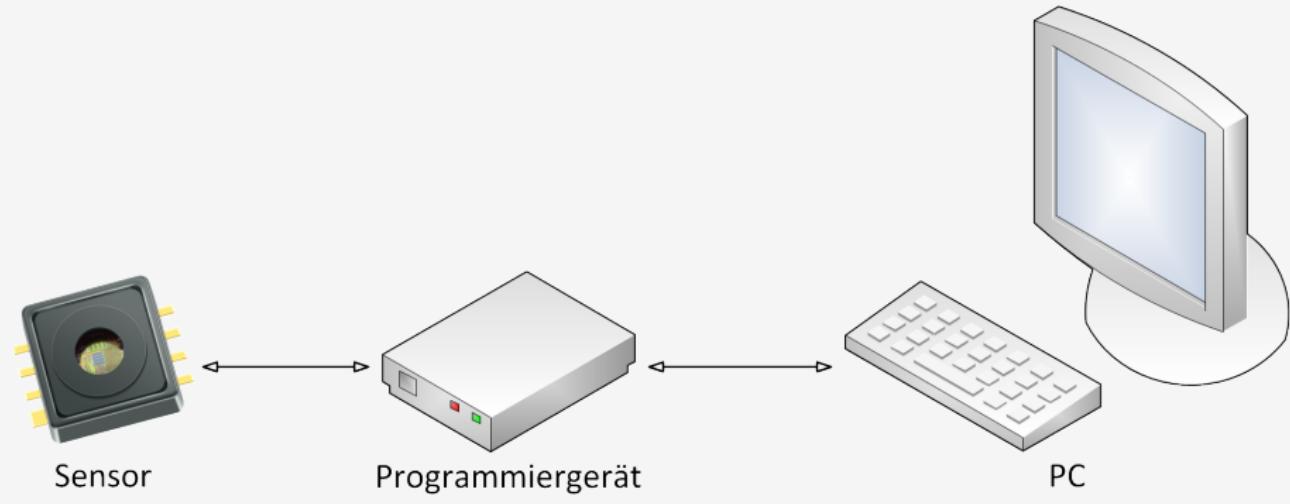
## Kunden

durch grafische Anzeige von Druckwerten die Entwicklung von Bauteilen unterstützen.

## Infineon

- Eigene Drucksensoren prüfen.
- Verhalten (EEPROM) der Sensoren umprogrammieren.

## Die drei Komponenten des Evaluierungswerkzeuges

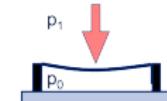


**Abbildung:** Die drei Komponenten des Evaluierungswerkzeuges

# Die drei Komponenten in der Übersicht

## Sensor

Digitale und analoge Sensoren nach kapazitivem Prinzip.



## Programmiergerät

- Besitzt Mikrocontroller und serielle Schnittstellen (USB und RS232).
- Hat Chip für USB-RS232-Übersetzung.
- Schnittstelle zwischen Sensor und PC.

## PC

- Mit grafischer Benutzeroberfläche (GUI).
- Konfiguriert Programmiergerät nach Sensorvorgaben.

# Projektumfang

## Aufgaben

- Firmware für 16- und 8-Bit-Mikrocontroller der Programmiergeräte in C.
- GUI in C#/.NET-Framework.

# Unterstützte Sensoren

## Sensoren

- Digitale Sensoren: KP253, KP256 und KP256 (KP25x)
- Analoge Sensoren: KP234, KP235 und KP236 (KP23x)

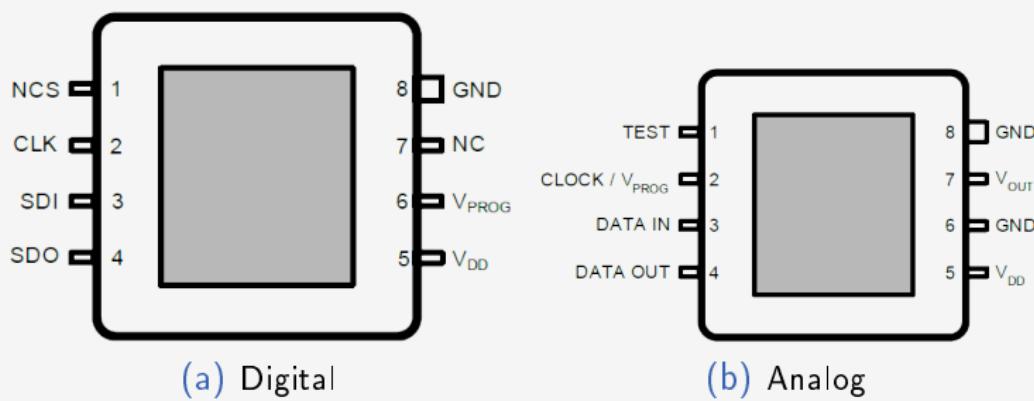


Abbildung: Pin-Konfiguration der digitalen und analogen Sensoren

# Kommunikation mit den Sensoren

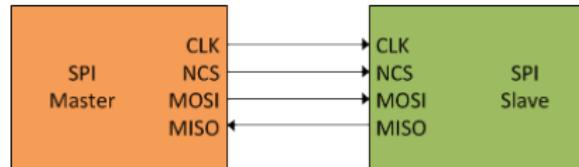
## Digitale und analoge Sensoren

- Sensoren KP25x nutzen Serial Peripheral Interface (SPI).
- Sensoren KP23x geben Druckwert an einem einzelnen Sensor-Pin aus.

## Serial Peripheral Interface (SPI)

= synchrones, serielles Bussystem mit vier Leitungen und Master-Slave-Prinzip.

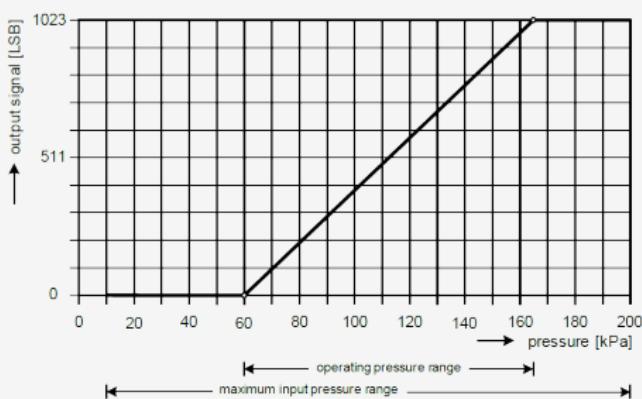
- Serial Clock (SCK bzw. CLK)
- Chip Select (CS bzw. NCS)
- Serial Data Out (SDO)
- Serial Data In (SDI)



# Transfer-Funktion der Sensoren

## Ausgegebener Druck-/Temperatur-Wert

- liegt digital in Bit bzw. analog in Volt vor.
- Wird mittels einer linearen Transfer-Funktion in die Größe Pascal umgerechnet.
- Jeder Sensor besitzt eigene Transfer-Funktion.



$$f(x) = m \times x + t,$$

$m$ : Steigung in  $\frac{\text{Bit}}{\text{kPa}}$

$x$ : Druck in kPa

$t$ : y-Achsenabschnitt

$f(x)$  liefert das Ausgangssignal in Bit.

# Unterstützte Programmiergeräte

## Programmer System Infineon Sensor Interface 2 (PGSISI2)

= Leiterplatine mit integrierten Schaltkreisen:

- 16-Bit-Mikrocontroller (Infineon XC164CS-16F)
- USB-RS232-Übersetzer
- Serielle Anschlüsse: USB-Port, 9-polige D-Sub-Buchse (RS232)



Abbildung: Anschlüsse der PGSISI2 (Vorderseite)

# Unterstützte Programmiergeräte

## Universal Wireless Link (UWLink)

= Leiterplatine mit integrierten Schaltkreisen:

- 8-Bit-Mikrocontroller (Infineon XC886CLM-8FF)
- USB-RS232-Übersetzer
- Serieller Anschluss: USB-Port

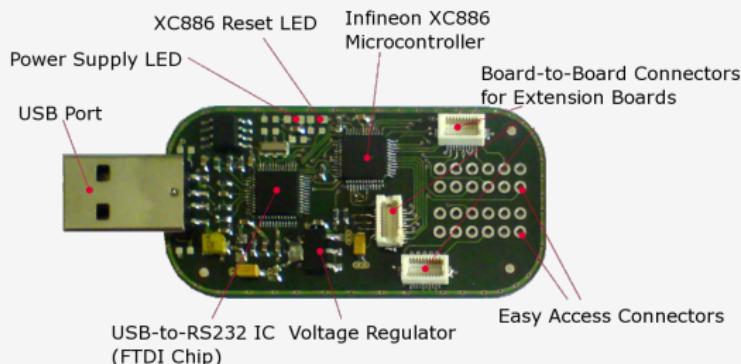


Abbildung: Komponenten des UWLinks

# PC (GUI)

Mindestanforderungen  
an Hard- und Software.

## Hardware

- 32-Bit-Prozessor
- 30 MB RAM
- 5 MB Festplattenspeicherplatz

## Software

- Microsoft Windows 2000
- .NET Framework 2.0

# Der Softwareentwicklungsprozess

## Ziele

- Kommunikation zwischen Komponenten ermöglichen.
- Daten von Sensor in GUI anzeigen.
- Sensor-EEPROM programmieren.

## Kommunikation Sensor $\longleftrightarrow$ Mikrocontroller

- Digitale Sensoren: Vier SPI-Leitungen mit SSC-Modul ansprechen („Synchronous Serial Channel“).
- Analoge Sensoren: Sensor-Pin mit Analog-Digital-Konverter auslesen.

## Kommunikation Mikrocontroller $\longleftrightarrow$ PC (GUI)

- PC ist Master, Mikrocontroller Slave.
- Protokoll notwendig.

# Die Mikrocontroller-Firmware

## Mikrocontroller

= integrierter Schaltkreis mit

- klassischen Befehls- und Speicherkomponenten: z.B. CPU, RAM, ROM etc.
- weiteren Komponenten: z.B. SSC für SPI-Kommunikation, UART für RS232-Kommunikation

## Firmware

genauso modularisieren wie Hardware.

- D.h. Modul „xy“ in Dateien *xy.h* und *xy.c* deklarieren/definieren.
- Z.B. Modul SSC → *SSC.h* und *SSC.c*.  
*SSC.h/.c* enthält z.B. Methoden zum Senden und Empfangen.
- ⇒ Hohe Wiederverwendbarkeit von Code.

# Modularisierung der Mikrocontroller-Firmware

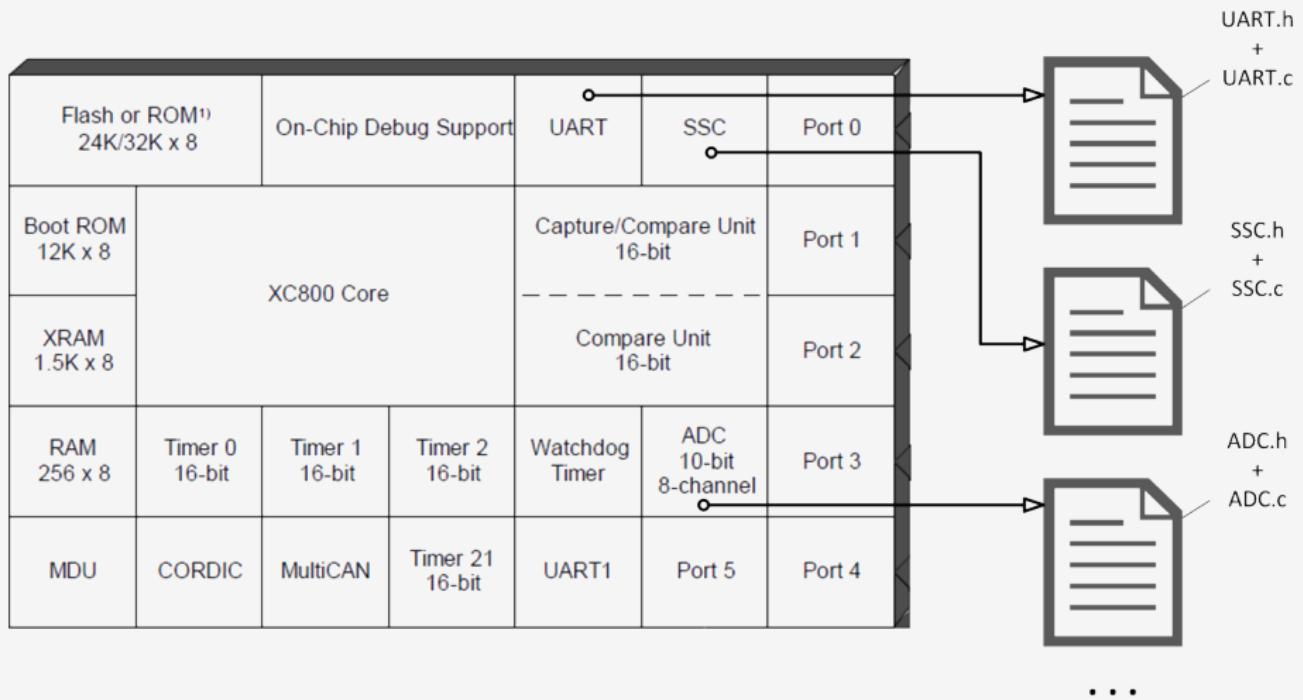


Abbildung: Abbildung der Mikrocontroller-Module in Software

# Software-Werkzeuge

## Versionsverwaltung

Zentrale Versionsverwaltung mit Subversion (Client TortoiseSVN).

## Integrierte Entwicklungsumgebung (IDE)

$\mu$ Vision 4.10 von Keil.

## API-Dokumentation

Doxxygen für Inline-API-Dokumentation mit Ausgabemöglichkeit in HTML, L<sup>A</sup>T<sub>E</sub>X uvm..

# Die Firmware-Architektur

```
1 void main()
2 {
3     // Initialize hardware modules.
4     ...
5     while(TRUE)
6     {
7         // Receive commands from PC.
8         ...
9         switch(command)
10        {
11             // Execute command.
12             case CMD_GET_FIRMWARE_VERSION:
13                 ...
14                 break;
15             case CMD_CONFIG_HARDWARE_SSC:
16                 ...
17                 break;
18         }
19         // Send data back to PC.
20         ...
21     }
22 }
```

Listing 1: Pseudocode für Firmware-Architektur (Main.c)

# Die Projektstruktur mit µVision

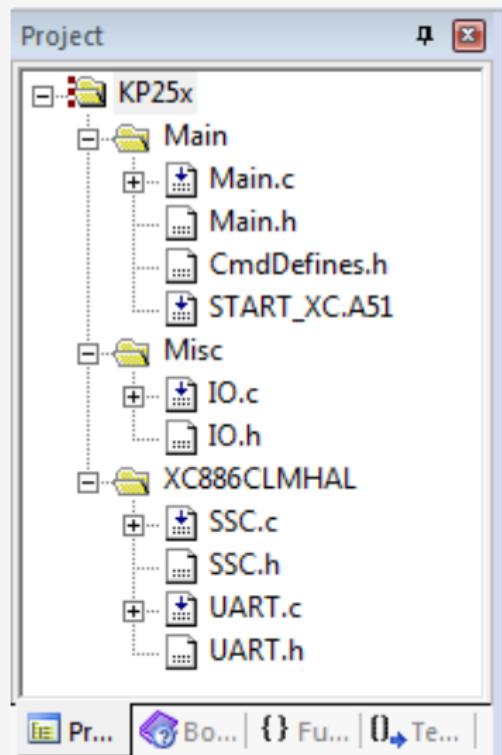


Abbildung: Projektstruktur für 8-Bit-Mikrocontroller

# Beispiel für eine Modul-Implementierung

## Notwendige Methoden für SSC

- `ubyte SSC_GetByte()`
- `void SSC_SendByte(ubyte OneByte)`
- `uword SSC_SendData(uword Data)`
- `void SSC_Init(SSC_Init_Parms const * const Params)`
- `void SSC_SetSSC_CONH_P(ubyte TEN, ...)`
- `void SSC_SetSSC_CONL_P(SSC_Data_Width BM, ...)`

# Beispiel für eine Modul-Implementierung

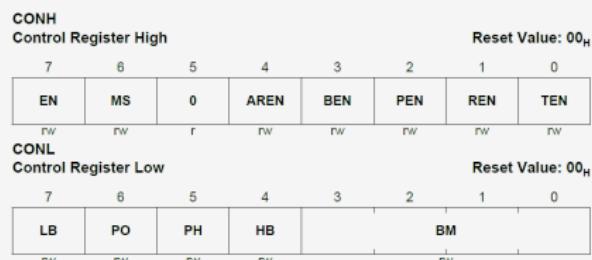


Abbildung: SSC Control Register beim 8-Bit-Mikrocontroller

```

1 void SSC_SetSSC_CONH_P(ubyte TEN, ubyte REN, ubyte PEN,
2   ubyte BEN, ubyte AREN, ubyte MS)
3 {
4   SSC_CONH_P &= SSC_CLEAR_SSC_CONH_P_MASK;
5
6   SSC_CONH_P |= TEN;
7   SSC_CONH_P |= (REN << 1);
8   SSC_CONH_P |= (PEN << 2);
9   SSC_CONH_P |= (BEN << 3);
10  SSC_CONH_P |= (AREN << 4);
11  SSC_CONH_P |= (MS << 6);
}
```

Listing 2: Methode SSC\_SetSSC\_CONH\_P in der Datei SSC.c

# Die Schnittstelle zwischen Firmware und GUI: Das Protokoll

## Protokoll-Anforderungen

- GUI → Firmware: Anfragen mit optionalen Parametern
- Firmware → GUI: Antworten unterschiedlicher Länge

<Command>    <Payload Length>                  <Payload>

1 Byte                      1 Byte                      „Payload Length“ Byte

Tabelle: Nachrichtenformat GUI zu Firmware

<Command>    <OK>    <Payload Length>                  <Payload>

1 Byte                      1 Byte                      1 Byte                      „Payload Length“ Byte

Tabelle: Nachrichtenformat Firmware zu GUI

# Protokollablauf

## Beispiel-Kommunikation

Anfrage (GUI): 1A 02 20 00

Antwort (Firmware): 1A 01 02 51 6F

## Erklärung

Anfrage (GUI):

1A „CMD\_RXTX\_HARDWARE\_SSC“ aus *CmdDefines.h*

02 2 Byte Payload mitsenden

20 00 Payload (KP25x-Anfrage zum Druckauslesen)

# Die grafische Benutzeroberfläche

## GUI

visualisiert Sensor-Funktionalitäten:

- Druck
- Temperatur
- EEPROM

## Druck und Temperatur

- Darstellung in 2-dimensionalen Graphen.
- Protokollierung in Liste.
- Exportierbar als Textdatei.

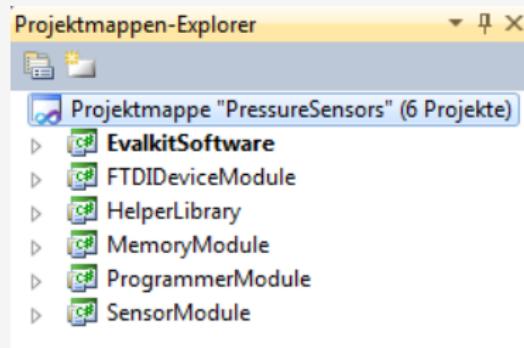
## EEPROM

Darstellung in separatem Fenster.

# GUI-Entwicklung

## GUI

- in C# mit .NET-Framework 2.0.
- Windows Forms



- EvalkitSoftware & SensorModule implementieren.
- Andere Module erweitern.

Abbildung: Die Projektstruktur der GUI

# Entwicklungsprozess

## 1. Schritt

### GUI-Prototyp

- Zuerst auf Papier
- Dann mit Windows Forms-Designer (IDE Visual Studio 2008)

## 2. Schritt

### Konzept für Architektur („SensorModule“)

## 3. Schritt

### Implementierung

# 1. Schritt: GUI-Prototyp

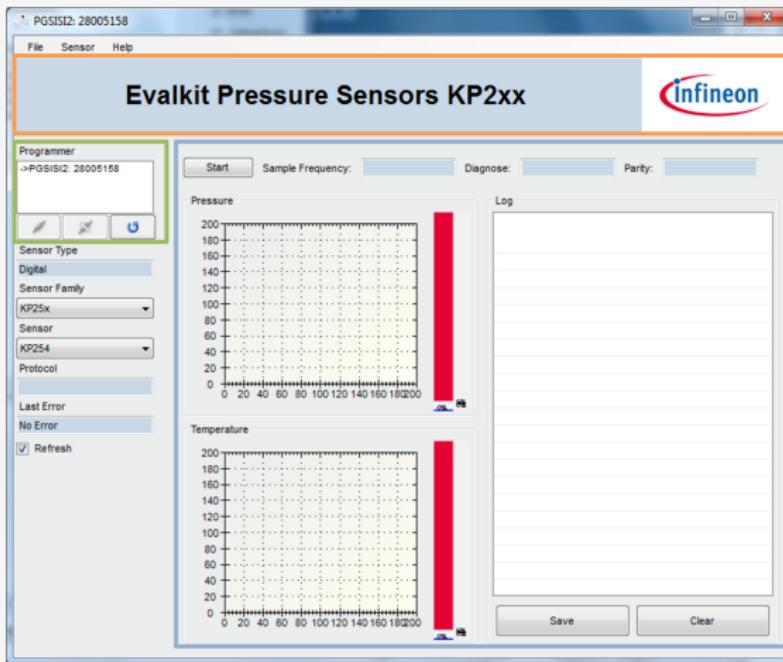


Abbildung: GUI-Prototyp mit Windows Forms-Designer

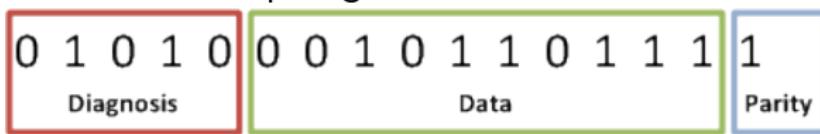
## 2. Schritt: Konzept für Architektur (Geschäftslogik)

### Unterste Ebene

- Anbindung an die Mikrocontroller-Firmware
- Klassen entsprechend Befehlen aus *CmdDefines.h*

### Mittlere Ebene

- Sende- und Empfangsrahmen für SPI-Kommunikation (KP25x):



### Oberste Ebene

- Sensor-Objekte mit Methoden wie GetPressure und GetTemperature

# Klassendiagramm Geschäftslogik (SensorModule)

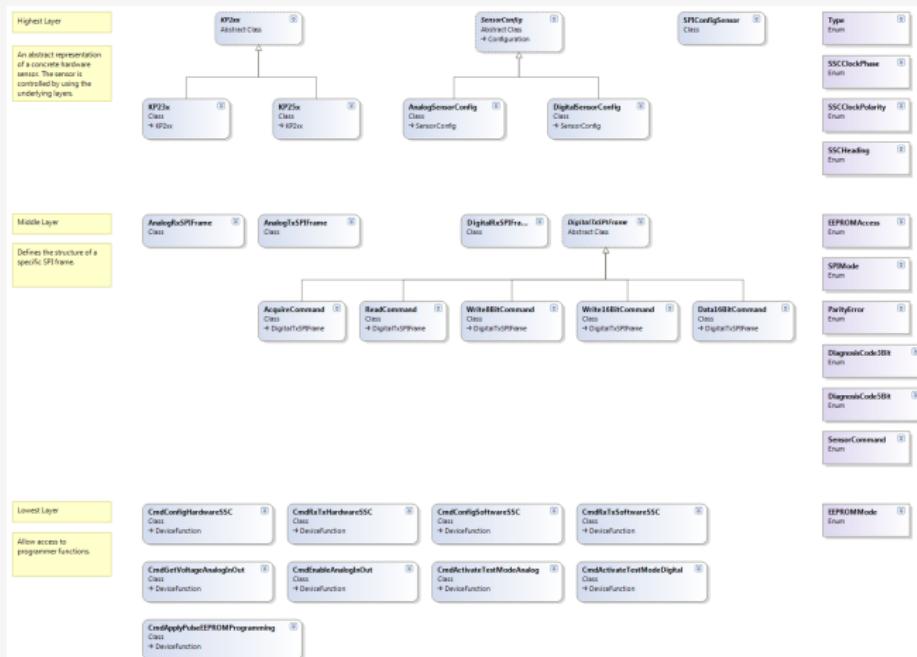


Abbildung: Vollständiges Klassendiagramm der Geschäftslogik

# Abschluss der Entwicklung

## Installationsprogramm

- Installationspaket (*.msi*) mit Windows Installer XML Toolset (WiX).
- Kopiert *EvalkitSoftware.exe* und andere Module als *.dll*.
- XML-Config-Dateien der Sensoren.

## Benutzerhandbuch

Mit Adobe FrameMaker 8 (WYSIWYG-Editor).

# Live-Demonstration

Danke für die Aufmerksamkeit.

Fragen und Diskussion.