netidee PROJEKTE

Open³ Toolbox
Konzept

Dokumentation | Call 15 | Projekt ID 5188

Inhalt

1 Einleitung

Nachfolgend wird die Dokumentation des Konzeptes für das Open[3] Toolbox Projekt beschrieben. Dieses Konzept dient dazu, eine Basis für die Entwicklung der Software zu schaffen. Mit diesem Konzept steht der endgültige Fahrplan fest und es kann hiermit mit der Entwicklung der Software gestartet werden.

2 Architektur des Gesamtsystems

Das System baut auf einer Stern-Architektur auf. Diese ist in *Abbildung 1 Aufbau des Systems* dargestellt.

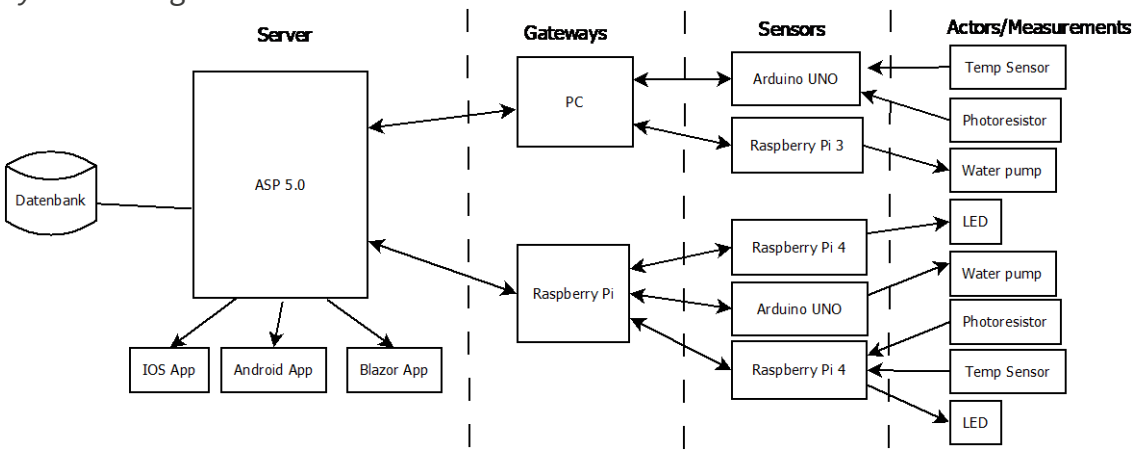

Abbildung 1 Aufbau des Systems

- **Server** ist mit der **Datenbank** verbunden
- **Server** stellt Cross-Platform IOS-, Android- und Blazor-**App** zu Verfügung
- Die **Server** kommunizieren über ein Protokoll (z.B. AMQP) mit den **Gateways**
- Die **Gateways** kommunizieren über ein Protokoll (z.B. LoRa, SigFox, etc.) mit den **Sensoren**
- Jeder Sensor kann zu 0 oder mehreren **Actors** und/oder **Measurements** angeschlossen sein
- **Actors** können von der App aus angesteuert werden
- **Measurements** liefern Daten in einem Intervall

3 Konzept für die Ausgangsbasis der Entwicklung

3.1 Konfiguration

Mittels einer Konfigurationsdatei sollen die Einstellungen für das System parametriert werden können. Diese soll mittels Restriktionen eingeschränkt werden, sodass nur für das System valide Konfigurationsdateien eingelesen werden können.

3.2 Verarbeitung der Konfigurationsdatei

- Server liest Datei ein und erkennt darin Gateways
- Server sendet jedem Gateway den für diesen relevanten Part der Datei
- Gateway liest die Datei ein und erkennt darauf Sensoren
- Gateway sendet er jedem Sensor den für diesen relevanten Part der Datei
- Sensor liest die Datei ein und erkennt darauf Measurements welche er laut Konfigurationen ausliest und Actors, welche er laut Konfigurationen beschreibt

3.3 Grafisches User Interface

In weiterer Folge sollen die Einstellungen über ein Grafisches User Interface parametriert werden können, welches die Konfigurationsdatei erstellt. Jedes Element kann je nach deren Möglichkeiten parametriert werden. In **Fehler! Verweisquelle konnte nicht gefunden werden.** ist ein Beispiel, ohne genauerer Parametrisierung, dargestellt. In ist das gleiche Beispiel mit den Einstellungsmöglichkeiten dargestellt.

- Dem **Server** können mittels dem + Button mehrere **Gateways** hinzugefügt werden
    - Über den Typen wird eingestellt welche Hardwarekomponente für den hinzugefügten **Gateway** verwendet wird

- Dem **Gateway** wird ein Sendeintervall vergeben, welches bestimmt, wie oft der **Gateway** seine gesammelten Daten zu dem **Server** sendet

- Dem **Gateway** können mittels dem + Button mehrere **Sensoren** hinzugefügt werden
  - Über den Typen wird eingestellt welche Hardwarekomponente für den hinzugefügten **Sensor** verwendet wird
  - Dem **Sensor** wird ein Sendeintervall vergeben, welches bestimmt, wie oft der **Sensor** seine gesammelten Daten zu dem **Gateway** sendet
- Dem **Sensor** können mittels dem + Button mehrere **Interfaces** hinzugefügt werden
  - Über den Typen wird eingestellt über welches **Interface** kommuniziert wird
  - Interfacespezifische Einstellungen, wie welche BusID bei dem I2C-Bus verwendet wird, können hier ebenfalls parametriert werden
- Dem **Interface** können mittels dem + Button mehrere **Chips** hinzugefügt werden
  - Über den Typen wird eingestellt um welchen **Chip** es sich handelt
  - Interfacespezifische Einstellungen, wie welche addresse der Chip in dem I2C-Bus hat, können hier ebenfalls parametriert werden
- Dem **Chip** können mittels dem + Button mehrere **Actors** und/oder **Measurements** hinzugefügt werden
  - Über den Typen wird eingestellt, ob es sich um ein **Actor** oder **Measurement** handelt
  - Der **Port** bestimmt welcher **Port** des Chips verwendet wird
  - **Measurements** wird ein intervall vergeben, welches bestimmt, wie oft das **Measurement** einen Wert ausliest

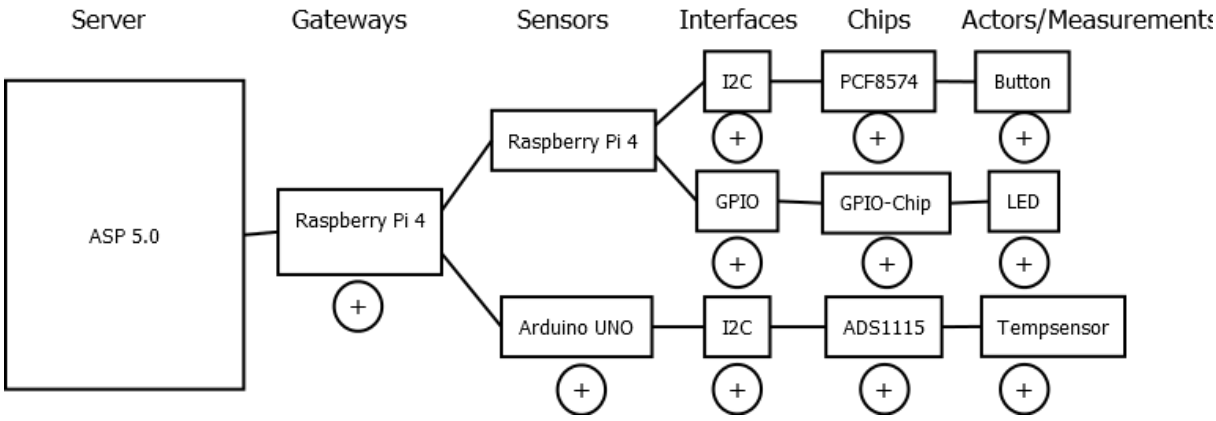

Abbildung 2Grafisches User Interface vereinfachte Darstellung

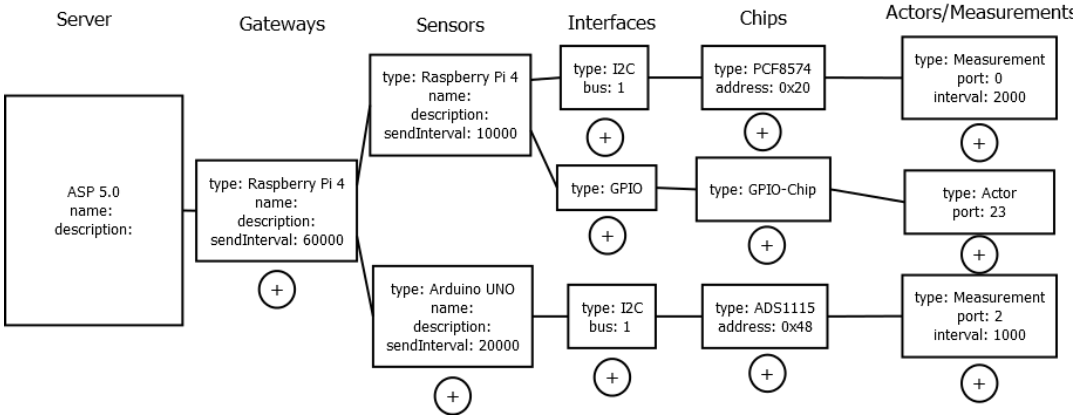

Abbildung 3 Parametriertes, grafisches User Interface

3.4 Hardwarekomponenten

Einplatinencomputer

Für den Einsatzzweck des **Gateways** und der **Sensoren** wird ein Einplatinencomputer vorgesehen.

Unter den betrachteten Einplatinencomputer stach der Raspberry Pi 4 durch das Preisleistungsverhältnis, die Anschlüsse, die Rechenleistung und die Möglichkeiten besonders hervor. Um die Berechnungen des Einsatzzweckes durchzuführen reichen 2GB RAM aus, weshalb sich für diese Variante entschieden wurde. Durch die Verbreitung des Raspberry Pis in der Community wird die Fehlerbehebung und die Dokumentationsfindung erleichtert, wodurch ein effizienteres Vorankommen in der Entwicklung ermöglicht wird.

Einplatinencomputer	Prozessor	Speicher	Konnektivität	Pins	Preis
Raspberry Pi 4	BCM2711, Quad-core 1.5GHz	1/2/4/8 GB RAM LPDDR4 Micro SD-Card	2xUSB 2.0, 2xUSB 3.0, Ethernet (Speed-Gigabit) Wi-Fi (Protocol-802.11ac) Bluetooth (V5)	40 Pins, UART, SPI, I2C 3.3 Volt	49€(1GB RAM) 42€ (2GB RAM) 59€ (4GB RAM) 77€ (8GB RAM)
Raspberry Pi Zero W	BCM2835, Single core 1 GHz	512 MB RAM, Micro SD-Card	USB 2.0, Wi-Fi (802.11n) Bluetooth (V4.1)	40 Pins, UART, SPI, I2C, 3.3 Volt	20€
Banana Pi M2	Allwinner A31s, Quad-core 1.2GHz	1024MB RAM DDR3, Micro SD-Card	4x USB 2.0, Ethernet(Speed-Gigabit) Wi-Fi (Protocol-802.11 b/g/n)	40 Pins, UART, SPI, I2C ,	54€
Orange Pi 3	Allwinner H6, Quad-core 1.8GHz	2048MB RAM LPDDR3, Micro SD-Card	USB 2.0, 4xUSB 3.0, OTG, Ethernet(Speed-Gigabit) Wi-Fi (Protocol-802.11 a/b/g/n/ac), Bluetooth (V5)	26Pins, UART, SPI, I2C, 3.3Volt	52€

Microcontroller

Für den Einsatzzweck der **Sensoren** wird neben dem Raspberry Pi zusätzlich ein Microcontroller vorgesehen.

Unter den betrachteten Microcontroller wurde der Arduino UNO R3 gewählt. Durch die Verbreitung des Arduino UNOs in der Community wird die Fehlerbehebung und die Dokumentationsfindung erleichtert, wodurch ein effizienteres Vorankommen in der Entwicklung ermöglicht wird.

Microcontroller	Prozessor	Speicher	Pins	Preis
Arduino UNO	ATmega328	32KB FLASH, 2KB RAM	14 DigitalI/O(6 PWM) , I2C, SPI, UART, 6 Analog, 3.3/5 Volt	22€
SparkFun RedBoard	ATmega328P	32KB FLASH, 2KB RAM	14 DigitalI/O(6 PWM) , I2C, SPI, UART, 6 Analog, 3.3/5 Volt	23€
Seeeduino v4.2	ATmega328	32KB FLASH, 2KB RAM	14 DigitalI/O(6 PWM), I2C, SPI, UART , 6 Analog, 3.3/5 Volt	12€
NodeMCU v2 - ESP8266	Tensilica LX106	96KB FLASH, 64KB RAM	16 DigitalI/O, PWM, I2C, SPI, UART, 3.3V	9€

Measurements/Actors

Für die **Measurements** und **Actors** wurde das Sensorkit X40 von Joi-it ausgewählt. Dieses bietet viele Möglichkeiten verschiedene Measurements/Actors über verschiedene Interfaces anzusteuern. Des weiteren ist ein Analog-Digital Converter enthalten, welcher es ermöglicht Analoge Sensoren auf dem Raspberry Pi auszulesen.

3.5 Technologieauswahl

Während der Entwicklung wurden Raspberry Pis und Arduinos als Hardware für Sensoren verwendet. Diese Geräte sind unter Entwicklern sehr beliebt. Außerdem sind sie einfach zu bedienen, was den Entwicklungsprozess beschleunigt.
Als Measurements und Actors wurde während der Entwicklung das SensorKit X40 der Firma JoyIT verwendet. Dieses Paket beinhaltet eine Vielzahl an Sensoren und Aktoren, die in Kombination mit einem Raspberry Pi oder Arduino leicht anzuwenden sind.

3.6 Cross-PlatformSDKs

Es wird bei der Entwicklung darauf geachtet, Cross Platform SDKs zu verwenden.
Der wesentliche Vorteil hierbei ist, dass man nicht an eine bestimmte Plattform gebunden ist.
Dies bietet ein großes Maß an Flexibilität im Betrieb, da der Server z.B. auf sowohl auf einem Windows- als auch auf einem Linux-Server installiert werden kann, ohne dafür die Software anpassen zu müssen.

3.7 Ablaufdefinition der Funktionalität der Applikation

Bevor Sensoren aktiviert werden können, muss eine Konfiguration des gesamten Projekts und aller sich darin befindenden Komponenten erstellt werden.

Über einen visuellen Editor können dem Projekt, Gateways samt Sensoren und dessen Bestandteile zusammengefügt werden.
Im Laufe dieser ersten Konfiguration wird jedem Gerät eine SetupID zugewiesen, welche im späteren Verlauf eine Rolle spielen wird.
Parallel zur Erstellung der Konfiguration des gesamten Projektes, muss ebenfalls eine "Basis-Software" auf den Geräten installiert werden. Während dieser Installation muss auf jedem Gerät die SetupID angegeben werden, die für das jeweilige Gerät in der Konfiguration vorgesehen ist. Dies ermöglicht die Identifikation des Gerätes bei der ersten Verbindung mit dem Server.
Ist die Konfiguration und die Installation der Basis-Software auf den Geräten abgeschlossen, können sich die Geräte mit dem Server verbinden und bekommen, abhängig von ihrer SetupID, die für sie vorgesehene Konfiguration und ihre endgültige, vom Server/Datenbank erzeugte ID, die im weiteren Betrieb verwendet wird.

3.8 Definition der Anforderungen und Funktionalitäten der Plug&Play Konfigurationssoftware

Ein Ziel ist es, die Konfigurationssoftware möglichst einfach zu gestalten. Mit der Plug&Play Funktionalität der Konfigurationssoftware ist es möglich, bereits von anderen Personen vordefinierte Sensoren, samt Measurements und Actors, in die eigene Konfiguration zu importieren.