AUTOSAR

SoSe 2015

27. Mai 2015

Beitragende:
Daniel Tatzel (DT)
Florian Laufenböck (FL)
Markus Wildgruber (MW)
Philipp Eidenschink (PE)
Tim Schmiedl (TimS)
Tobias Schwindl (TobiS)

VersionsNr	Datum	Auslöser	Beschreibung
1.0	21.04.2015	DT	Erster Entwurf

1 Projekt Beschreibung

1.1 Vernetzte Ballschussanlage

- 1-2 Bricks
- Ausgabe(durch Display,LEDs etc.)
- Stop-Trigger
- Variable Aufteilung unter den Bricks: Stopp-Taste, Auslösung Taste(auch über Ultraschall), Ausgabe

1.2 Benötigte VFB-Komponenten und Schnittstellen (DT)

- Komponenten
  - Application Software Component
  - Sensor-Actuator Software Component
  - ECU Abstraction Software Component
- Schnittstellen
  - Client/Server
  - Events
  - Sender/Receiver (auch mit synchronisierung)

1.3 Namenskonventionen und Standardrückgabtyp (Alle)

Für RTE-Funktionen:	RTE_<Funktionsname>_<Portname>_<Direction>
Für den Rest:	<Komponente>_<Funktionsname>
Standardrückgabtyp:	uint32_t

1.4 Komponenten Funktionsapi

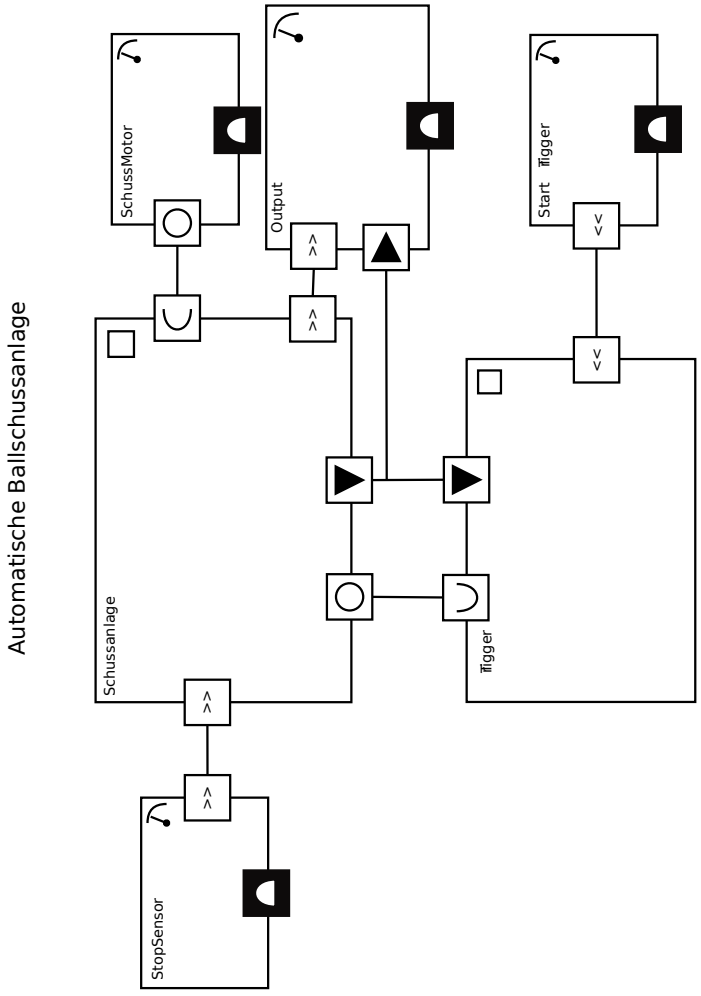


Abbildung 1.1: Komponentendiagramm der Ballschussanlage (DT)

2 Komponenten-Beschreibung

Schussanlage (FL)

- Besteht aus einer Task mit zwei Runnables
- erste Runnable prüft periodische die Abbruchbedinung(hier: Taster)
- zweite Runnable managt den Schussmotor
- Kein Autostart des Tasks, wird über den Trigger gestartet
- Ports siehe Komponentendiagramm

Benötigt: Task und Event

Trigger (PE)

- Ein Task
- Wird zu beginn gestartet (Autostart)
- Wartet auf Event vom Input

Benötigt: Task und Event

Output (MW)

- Autostart
- Wird durch Event von Schussanlage getriggert
- Prüft nach Event die empfangene Nachricht
- Zeigt Nachricht in Abhängigkeit der empfangen Nachricht an

Benötigt: Task und Event

SchussMotor (TimS)

- Kein Autostart
- Servertask wird durch Schussanlage (client) gestartet
- Steuert Motor zum schießen an

StopSensor (TobiS)

- Autostart
- Prüft Taster
- Setzt Event für Schussanlage

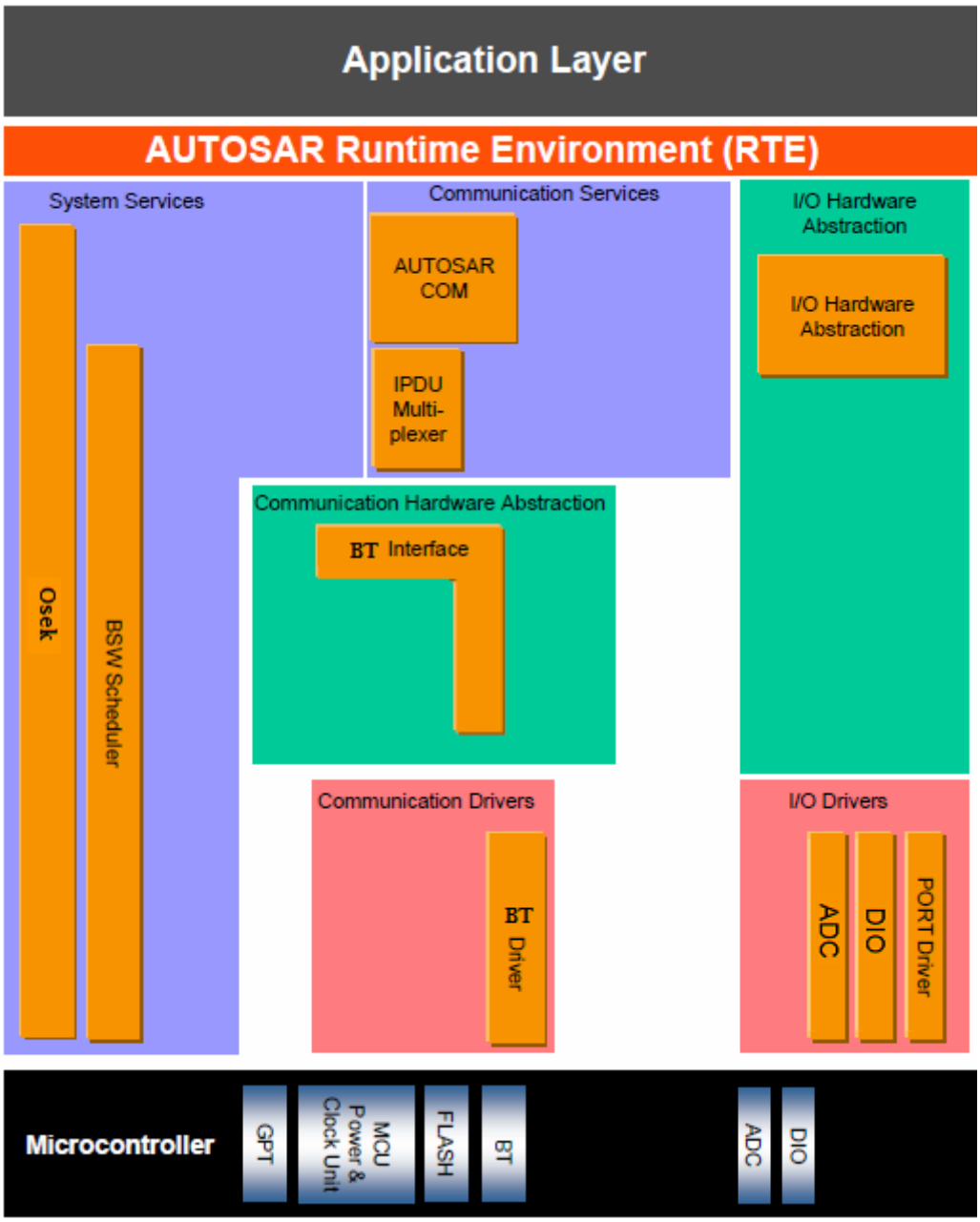
Benötigt: Task, Timer und Event

StartTrigger (TobiS)

- Task zum Erkennen von Zielen
- Autostart
- Sendet Event an Trigger
- Erkennung durch periodische Abfrage

Benötigt: Task und Timer

2.1 Architekturschicht und Funktionsapi



2.1.1 Funktionapi

1.) System Services
keine Funktionen

2.) Communication Services

- Abstraktionsebene um Nachrichten zu verschicken
- StdReturnType TransmitMessage(char∗ message)
- StdReturnType ReceiveMessage(char∗ message)

3.) I/O Hardware Abstraction

- StdReturnType ReadDigitalInput(PortName)
- StdReturnType ReadAnalogInput(PortName)
- StdReturnType DriveMotor(Port_Name, Direction, speed, angle)

4.) Communication Hardware Abstraction

- Für unser Projekt eigentlich unnötig, da wir nur eine Kommunikationsebene haben(theoretisch mehr durch I2C, aber hier uninteressant)
- StdReturnType SendMessageBT(char* message)
- StdReturnType GetMessageBT(char* message)

5.) Communication Drivers

- es wird nur ein Treiber für das Hardware BT gebraucht:
- StdReturnType BT_Write(char* message)
- StdReturnType BT_Read(char* message)

6.) I/O Drivers

- benötigt für den zusätzlichen I2C expander
- StdReturnType ReadI2C(PortName)
- StdReturnType WriteI2C(PortName)