Projektübersicht

Hinweis: Alle in diesem Dokument beschriebenen Projekte wurden erfolgreich abgeschlossen und sind bereits auf dem Markt verfügbar.

Index

1. HMI-Lösung

Projektname

Entwicklung eines Linux-basierten Betriebssystems für eine HMI-Lösung in industriellen Walzenmaschinen.

Kurzbeschreibung

Entwicklung und Integration eines vollständigen Embedded-Linux-Systems für ein industrielles HMI-Bedienungsdisplay einer Walzenmaschine. Ziel war die Bereitstellung einer robusten, performanten und zuverlässigen Plattform mit moderner Touch-Bedienung sowie optimierter Systemstartzeit.

Technische Umgebung

- **Programmiersprachen:** C, C++, Python, Bash, QML
- **Betriebssysteme / Bootloader:** Linux Kernel (Mainline), Uboot-denx
- **Buildsystem:** Yocto Project (Zeus), Bitbake, Board Support Package
- **Hardware / SoCs:** ARM-basierte Industrieplattform (NXP i.MX6 und Ti AM335X)
- **Displays & Schnittstellen:** LVDS (5“, 7“ und 10“), Kapazitiver Touch, I²C, GPIO, UART, SPI, Ethernet, CAN, USB, Analog Kamera
- **Entwicklungstools:** GCC, Git, Make, CMake, Docker, Qt5
- **Testing & Debugging:** Oszilloskope, Logikanalysator, UART-Debug-Konsole, Kernel-Logging
- **Versionierung & CI/CD:** Git, Jenkins
- **Projektmanagement:** Jira
- **Dokumentation:** Doxygen, MS Word

Aufgaben & Verantwortlichkeiten

- Aufbau eines vollständigen Board Support Package (BSP) mit Yocto, inklusive Layer-Struktur und mehreren Image-Varianten (Produktiv-, Test- und EMV-Image)
- Anpassung und Pflege des Device Trees für drei Displayvarianten (5“, 7“, 10“)
- Konfiguration, Portierung und Patchen von U-Boot
- Konfiguration, Portierung und Patchen des Linux-Kernels
- Integration und Anpassung von Treibern für:
  LVDS, Touch, I²C, GPIO, UART, SPI, Ethernet, CAN, USB, analoge Kamera, Temperatursensoren, RTC, EEPROM
- Performance-Optimierung der Bootzeit (u. a. „Falcon Mode“)
- Integration von App-Boot und Maintenance-Boot
- Implementierung einer Firmware-Update-Funktion über Diagnostics over Internet Protocol (DoIP)
- Optimierung des UDS-Stacks
- Entwicklung einer Hardware Abstraction Layer Bibliothek (HAL)
- Aufbau automatisierter Tests für Board-Interfaces
- Integration von Boot2Qt, um dem Kunden das Live-Testen und Debuggen seiner Qt-Applikation direkt auf dem HMI zu ermöglichen

2. USV Lösung

Projektname

Entwicklung eines Linux-basierten Betriebssystems für eine USV (Unterbrechungsfreie Stromversorgung) Lösung.

Kurzbeschreibung

Entwicklung und Integration eines vollständigen Embedded-Linux-Systems für ein USV-Lösung. Ziel war die Bereitstellung einer robusten, performanten und zuverlässigen Plattform.

Technische Umgebung

- **Programmiersprachen:** C, Python, Bash
- **Betriebssysteme / Bootloader:** Linux Kernel, U-Boot
- **Buildsystem:** Yocto Project (zeus), Bitbake, Board Support Package
- **Hardware / SoCs:** ARM-basierte Industrieplattform (Ti Am335x)
- **Displays & Schnittstellen:** Ethernet, I²C, GPIO, RS-232, RS-485, USB
- **Entwicklungstools:** GCC, Git, Make, CMake, Docker
- **Testing & Debugging:** Oszilloskope, Logikanalysator, UART-Debug-Konsole, Kernel-Logging
- **Versionierung & CI/CD:** Git, Jenkins
- **Projektmanagement:** Jira
- **Dokumentation:** MS Word

Aufgaben & Verantwortlichkeiten

- Aufbau eines vollständigen Board Support Package (BSP) mit Yocto, inklusive Layer-Struktur und mehreren Image-Varianten (Produktiv-, Test- und EMV-Image)
- Anpassung und Pflege des Device Trees für unterschiedliche Hardwarevarianten
- Konfiguration, Portierung und Patchen von U-Boot
- Konfiguration, Portierung und Patchen des Linux-Kernels
- Aufbau automatisierter Tests für Board-Interfaces

3. Smart Home Bedienzentrale

Projektname

Entwicklung eines Linux-basierten Betriebssystems für eine Smart Home Bedienzentrale.

Kurzbeschreibung

Entwicklung und Integration eines vollständigen Embedded-Linux-Systems für ein Smart Home Bedienzentrale. Ziel war die Bereitstellung einer robusten, performanten und zuverlässigen Plattform mit moderner Touch-Bedienung.

Technische Umgebung

- **Programmiersprachen:** C, C++, Python, Bash, QML
- **Betriebssysteme / Bootloader:** Linux Kernel (linux-imx), U-Boot (u-boot-imx)
- **Buildsystem:** Yocto Project (Hardknott), Bitbake, Board Support Package
- **Hardware / SoCs:** ARM-basierte Industrieplattform (NXP i.MX8)
- **Displays & Schnittstellen:** 7 Zoll MIPI DSI (Display Serial Interface), Kapazitiver Touch, WLAN, Ethernet, I²C, GPIO, UART
- **Entwicklungstools:** GCC, Git, Make, CMake, Docker, Qt5
- **Testing & Debugging:** Oszilloskope, Logikanalysator, UART-Debug-Konsole, Kernel-Logging
- **Versionierung & CI/CD:** Git, Jenkins
- **Projektmanagement:** Jira
- **Dokumentation:** MS Word

Aufgaben & Verantwortlichkeiten

- Aufbau eines vollständigen Board Support Package (BSP) mit Yocto, inklusive Layer-Struktur und mehreren Image-Varianten (Produktiv-, Test- und EMV-Image)
- Anpassung und Pflege des Device Trees für unterschiedliche Hardwarevarianten
- Konfiguration, Portierung und Patchen von U-Boot
- Bereitstellung der Option, das Board per TFTP-Server zu booten (Netzwerkboot)
- Konfiguration, Portierung und Patchen des Linux-Kernels
- Integration und Anpassung von Treibern für:
  MIPI DSI, Touch, Ethernet, WLAN, Ethernet, Temperatursensoren, RTC, EEPROM, Näherungssensor
- Entwicklung einer Qt-Applikation für interaktive Hardwaretests
- Aufbau automatisierter Tests für Board-Interfaces

4. Intelligente Kamera

Projektname

Entwicklung einer STM32 Firmware zur Überwachung des Hauptsystems bzw. der Haupt CPU.

Kurzbeschreibung

Entwicklung und Integration einer STM32-basierten Bare-Metal-Firmware für ein Kamerasystem. Ziel war die Bereitstellung einer robusten, performanten und zuverlässigen Überwachungslogik, die den Zustand der Haupt-MCU kontinuierlich erfasst und im Fehlerfall automatisch reagiert. Zusätzlich wurden Diagnosefunktionen und Kommunikationsmechanismen zwischen der STM32-Firmware und dem Linux-System der Haupt-CPU implementiert.

Technische Umgebung

- **Programmiersprachen:** C, bash
- **Firmware:** Bare Metal Firmware
- **Hardware:** STM32 (Cortex-M4)
- **Schnittstellen:** UART, FDCAN, I2C, ADC, SPI, Timer, DMA
- **Entwicklungstools:** STM32CubeIDE
- **Testing & Debugging:** Oszilloskope, Logikanalysator, UART Debug Konsole
- **Versionierung:** Git
- **Projektmanagement:** Jira
- **Dokumentation:** MS Word

Aufgaben & Verantwortlichkeiten

- Entwicklung eines Bootloaders, der Firmware-Updates überprüft und entsprechend verarbeitet
- Entwicklung der Applikations-Firmware getrennt vom Bootloader
- Erstellung eines Firmware-Pakets, das Bootloader und Applikation kombiniert
- Entwicklung eines I2C interface zwischen Linux (Haupt CPU) und STM32, sodass der Linux System den STM32 wie ein I2C Device ansprechen kann
- Implementierung eines Update-Mechanismus, der STM32-Firmware direkt aus dem Linux-Userspace aktualisiert
- Implementierung der Funktionalität, mit der STM32 erkennt, welche Quelle (CAN, KL15, …) das System aufgewacht hat, und diese Information an Linux weitergibt
- Erkennung des angeschlossenen Kameratyps durch den STM32 und Weitergabe der Informationen an das Linux-System
- Implementierung eines „Request-Response Watchdog“ zur Überwachung der Haupt-CPU

5. Parksystem

Projektname

Test und Fehlerbehebung einer bestehenden STM32-Firmware.

Kurzbeschreibung

Durchführung von Funktionstests sowie Analyse und Behebung von Fehlern in einer bereits bestehenden STM32 auf Basis von Free RTOS Firmware. Ziel war es, die Stabilität, Zuverlässigkeit und Funktionssicherheit des Systems sicherzustellen.

Technische Umgebung

- **Programmiersprachen:** C
- **Firmware:** Free RTOS
- **Hardware:** STM32 (Cortex-M0)
- **Schnittstellen:** LCD, UART, I2C, ADC, SPI, Timer, DMA, RTC
- **Entwicklungstools:** STM32CubeIDE
- **Testing & Debugging:** Oszilloskope, Logikanalysator, UART-Debug-Konsole
- **Versionierung:** Git
- **Projektmanagement:** Jira
- **Dokumentation:** MS Word

6. I.MX6X und I.MX8X

Projektname

Aktualisierung des Board Support Package für NXP i.MX6- und i.MX8X-SMARC-Module.

Kurzbeschreibung

Aktualisierung und Pflege des kompletten Board Support Package (BSP) für mehrere Varianten der NXP i.MX6- und i.MX8-Module. Ziel war es, Kernel, Bootloader, Yocto-Layer und Applikationen auf den aktuellen Stand zu bringen sowie die Kompatibilität und Stabilität für unterschiedliche Hardwarevarianten sicherzustellen.

Technische Umgebung

- **Programmiersprachen:** C, Python, Bash
- **Betriebssysteme / Bootloader:** Linux Kernel (linux-imx), U-Boot (u-boot-imx)
- **Buildsystem:** Yocto Project (Warrior, Zeus, Hardknott, Kirkstone, Mickledore), BitBake, Board Support Package
- **Hardware / SoCs:** ARM-basierte Industrieplattform (NXP i.MX6 und i.MX8)
- **Schnittstellen / Peripherie:** LVDS (verschiedene Varianten), HDMI, MIPI CSI, MIPI DSI, I²C, SPI, QSPI, eMMC, SD-Card, USB-OTG, CAN, Ethernet, WLAN, UART, I2S, RTC
- **Entwicklungstools:** GCC, Git, Make, CMake, Docker
- **Testing & Debugging:** Oszilloskope, Logikanalysator, UART-Debug-Konsole, Kernel-Logging
- **Versionierung & CI/CD:** Git, Jenkins
- **Projektmanagement:** Jira
- **Dokumentation:** MS Word

Aufgaben & Verantwortlichkeiten

- Integration der Yocto-Layer
- Update und Anpassung von U-Boot
- Update und Anpassung des Linux-Kernels
- Test und Validierung sämtlicher Schnittstellen und Peripherie (Kernel & Userspace)