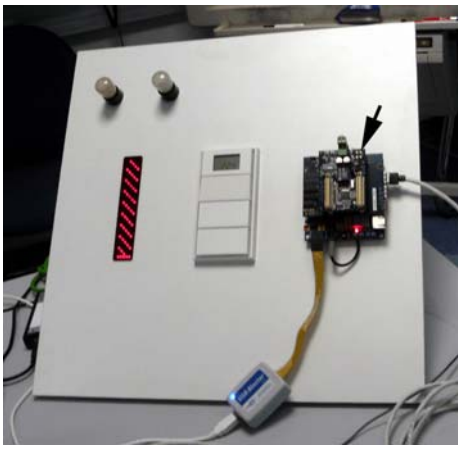
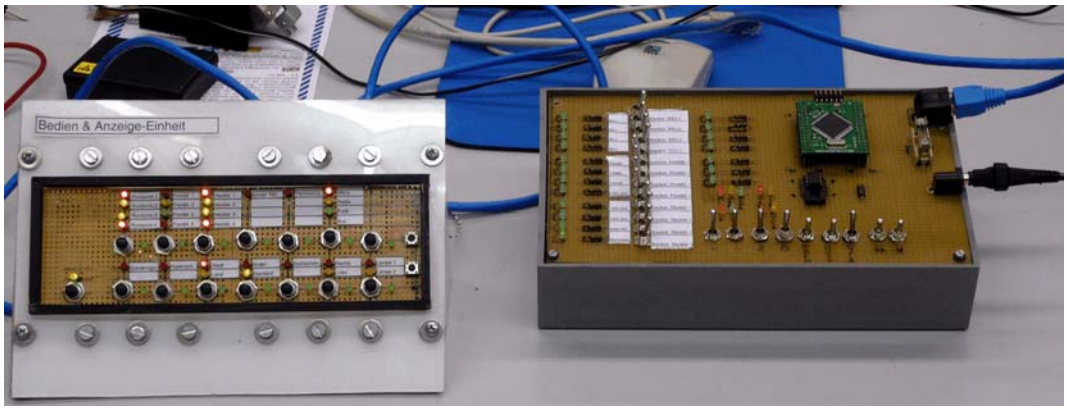
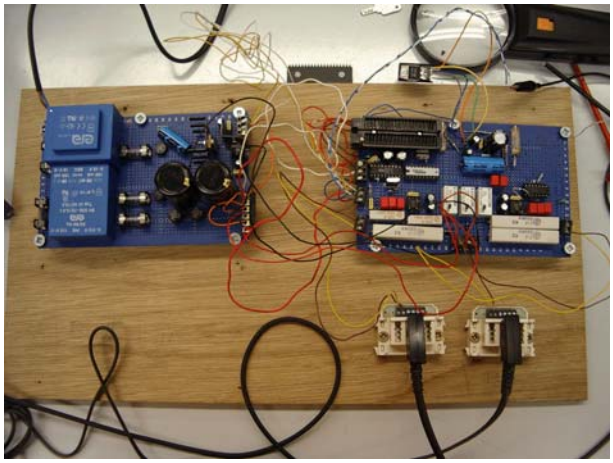
Einführung AU1 / HS 1 (SS09)

Die typische Entwicklungsaufgabe ist eine Allround-Aufgabe:

- Systemlösung finden,
- Komponenten auswählen,
- Hardware entwerfen – und zwar unter Berücksichtigung aller Spitzfindigkeiten (EMV, ESD, Prüfbarkeit, wirtschaftliche Fertigung, Service (fertigungs-, prüf- und servicegerechter Entwurf),
- programmieren,
- ggf. Testumgebung bereitstellen,
- zum Laufen bringen.







Automatisierungstechnik AU1	Hard- und Software-Engineering HS1 (SS09)	Hard- und Software-Engineering HS2
Mikrocontrollertechnik. Elementare Problemlösung mit Mikrocontrollern • Elementare Einführung in die Rechnerarchitektur • Einführung in die maschinennahe Programmierung (am Beispiel Atmel AVR)	Mikrocontrollertechnik. Elementare Problemlösung mit Mikrocontrollern • Elementare Einführung in die Rechnerarchitektur • Einführung in die maschinennahe Programmierung (am Beispiel Atmel AVR) • Universelle und programmierbare Logik • Externe Erweiterung von Mikrocontrollern (Pegelwandlung, ESD, Isolation, Bussysteme, Speichererweiterung usw.)	Mikrocontrolleranwendung. Programmorganisation und Software der Mikrocontroller von Grund auf. • Die eingebaute Peripherie • Grundlagen der Realzeitprogrammierung • Grundlagen der Multiprozessorsysteme (mehrere Mikrocontroller im Verbund) • Grundlagen der Mensch-Maschine-Schnittstellen

Der Lehrstoff:

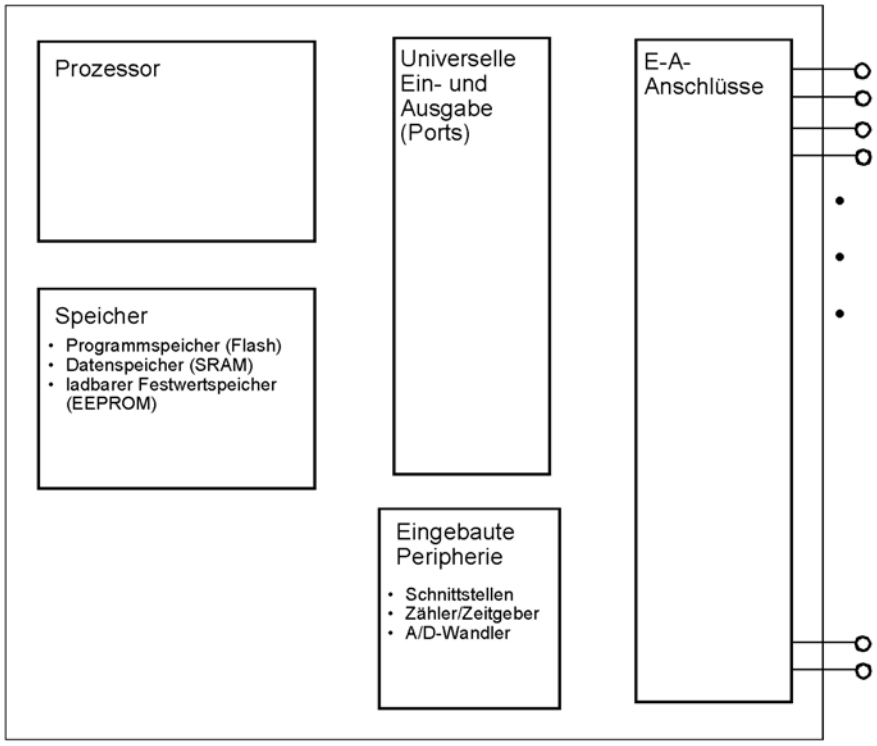
1. Aufgaben und Lösungsansätze in der Automatisierungstechnik
2. Problemlösung mit Mikrocontrollern
3. Mikrocontroller – Aufbau und Wirkungsweise
4. Einführung in die Rechner- und Prozessorarchitektur
5. Programmiermodelle und Programmierphilosophien
6. Problemlösung durch Anwendungsprogrammierung
7. Grundlagen der Maschinenprogrammierung
8. Einführung in die Realzeitprogrammierung

Der Fokus: Mikrocontroller im praktischen Einsatz

Typische Szenarien der Anwendung:

- die Hardware ist gegeben,
- die Entwicklungsumgebung ist gegeben,
- die Ressourcen sind knapp,
- die Problemlösung steht unter Zeitdruck,
- man kann nicht alles haben,
- es läuft keineswegs alles ideal – man muß sich halt zu helfen wissen ...

Anmerkung:
Im Gegensatz dazu betrifft das Fach Hard- und Software-Engineering die Entwickung des gesamten Komplexes aus Hard- und Software von Grund auf.



Der Mikrocontroller
- ist zum einen ein kleiner Prozeßrechner,
- ist zum anderen eine Alternative zur anwendungsspezifischen Hardware (Kostensenkung – er wird nicht deswegen eingesetzt, weil es etwas zu rechnen gibt, sondern nur um eine bestimmte Funktion billiger zu erfüllen als dies mit einer zweckgebundenen Schaltung möglich wäre).

Vermischte Stichworte

Der Mikrocontroller und die Außenwelt:

- E-A-Ports
- Eingebaute Peripherie
- Externe Erweiterung

Problemlösung durch Anwendungsprogrammierung:
Wir haben nur die blanke Hardware und einen Compiler und/oder Assembler.

Anwendungsprogrammierung in C:

- Besonderheiten (z. B. gegenüber der PC-Programmierung): C ist hier maschinen- und compilerspezifisch,
- Ansätze und Auswege.

Der C-Compiler ist ein Mittel zur Arbeitserleichterung, gewährleistet aber keine Maschinenunabhängigkeit. Bereits beim Wechsel vom Compiler des Anbieters A zum Compiler des Anbieters B geht es nicht ohne Eingriffe von Hand ab.

Die gängige Praxis:

- C-Programmierung, wenn es um algorithmische Probleme geht,
- Assemblerprogrammierung, wenn die Hardware bis aufs letzte auszunutzen ist (z. B. exakte Kontrolle über das Zeitverhalten),
- die meisten Entwickler bleiben bei einer Entwicklungsumgebung (und damit beim gleichen Compiler). Nachteile (jedes System hat welche), die sich von Zeit zu Zeit unangenehm bemerkbar machen, werden in Kauf genommen.
- die gesamte Anwendung wird an sich in C programmiert; Assemblerprogrammstücke werden bedarfsweise eingefügt (richtig: in bestimmten Funktionen konzentriert; falsch: gleichsam mit dem Salzstreuer über das gesamte Programm verteilt).

Einführung in die Rechner- und Prozessorarchitektur:
AU1 betrifft nicht die Rechnerarchitektur im allgemeinen Sinne. Vielmehr beschränken wir uns

- auf Auslegungen, die sich in der Praxis durchgesetzt haben,
- auf Leistungsklassen, wie sie für Embedded Systems typisch sind (ein Befehl zu einer Zeit, keine Parallelverarbeitung usw.).

Weshalb maschinennahe Programmierung?

- sie ist in der Praxis nach wie vor erforderlich (Nutzung maschinenspezifischer Besonderheiten, maximale Ausnutzung der Hardware (höchstes Leistungsvermögen oder geringster Aufwand), Umgehung von Unzulänglichkeiten (Workarounds)),
- sie vermittelt grundlegendes Erfahrungswissen zum Verstehen, Beurteilen und Auswählen von Prozessorarchitekturen.

Lernziele:

- Grundlagen des Aufbaus und der Wirkungsweise von Prozessoren und Mikrocontrollern (Einführung in die Rechnerarchitektur),
- Grundlagen der Anwendungsprogrammierung,
- Grundlagen der Maschinenprogrammierung,
- Grundlagen der Programmentwicklung (Ausdenken – Programmieren – zum Laufen bringen).

Übungen/Praktika:

- die E-A-Ports der Mikrocontroller ausnutzen,
- Anwendungsprogrammierung in C,
- Anwendungsprogrammierung in Assembler (Atmel AVR).