64-211 Übung Eingebettete Systeme SoSe 2025

Aufgabenblatt 00 Termine: KW 15

Gruppe	
Name(n)	Matrikelnummer(n)

Die Übungen zur Vorlesung „Eingebettete Systeme“ sollen den Einstieg in den Umgang und die Programmierung eines Mikrocontroller-Systems anhand der Lösung kleiner praktischer Aufgabenstellungen ermöglichen, um so das im Rahmen der Vorlesung erworbene methodisch-theoretische Wissen auf praktischen Problemstellungen anzuwenden. Hierfür wird ein Mikrocontroller-Board, ein Arduino MEGA2560- oder ein Arduino DUE-Board, und die für die jeweilige Aufgabenstellung erforderliche Zusatzhardware (Taster, Servos, Display, GPS u. ä.) benötigt.

Für die Lösung der Übungen zu ES steht für die häusliche Vorbereitung optional eine Simulationsumgebung bereit, mit der Lösungsansätze validiert werden können. In den Übungen selbst werden die Lösungen dann auf realer Hardware, hier wird ein Arduino DUE-Board nebst Peripherie verwendet, erprobt.
Als Simulationsumgebung wird das *VSM Simulation-Tool* (Virtual System Modelling) *Proteus 8 for Arduino AVR* von Labcenter Electronics eingesetzt.
Proteus 8 for Arduino AVR stellt Simulationsmodelle für die auf AVR-Microcontroller basierenden Arduino-Boards und unterschiedlichster Peripherieelemente (sog. Breakout-Boards) bereit. Für die Lösung der Aufgaben sollte in der Simulation das Arduino MEGA 2560-Board, eingesetzt werden, da dieses mit dem am komfortabelsten ausgestattten Mikrocontroller der ATmega-Reihe, dem *ATmega2560* ausgestattet ist. Darüber hinaus lassen sich Programme für den Arduino Mega 2560 leicht auf andere Architekturen, z.B. den Arduino DUE, der in dem praktischen Teil eingesetzt wird, portieren.

Das **Arduino MEGA 2560**-Board basiert auf einem Prozessor der **8-bit AVR** Architektur, dem ATmega2560, während das **Arduino DUE**-Board auf einem **32-bit ARM Cortex-M3 RISC Prozessor**, dem SAM3X basiert.

Arduino MEGA 2560 und Arduino DUE Entwicklungsboard:
`store.arduino.cc/arduino-mega-2560-rev3`
`store.arduino.cc/products/arduino-due`

Machen Sie sich bitte zunächst mit der **Pinbelegung** des Arduino MEGA2560 Boards vertraut, bevor Sie mit der Lösung der Aufgaben beginnen. Vergleichen Sie die Pinbelegung des Arduino MEGA 2560 und Arduino DUE Entwicklungsboards. Entsprechende Übersichten finden Sie Online unter: Arduino MEGA 2560 Pinbelegung und Arduino DUE Pinbelegung.

Arduino Entwicklungsumgebung `www.arduino.cc/en/Main/Software`

Zur Programmierung wird grundsätzlich die Arduino IDE verwendet. Auch lassen sich Programme für den Arduino ausschließlich unter Verwendung von *Proteus 8* entwickeln, allerdings wird auch hier eine installierte Arduino IDE vorausgesetzt. Die Installation der Arduino-IDE auf ihrem Rechner – falls noch nicht vorhanden – wird Bestandteil der Aufgabe 0.1 dieses Blattes sein.

Online Hilfe www.arduino.cc/reference/en

Die Online-Referenz des Arduino Framework ist nicht immer perfekt, stellt jedoch eine wertvolle Quelle für grundlegende Information zur Syntax und Semantik enthaltener Funktionen dar. Vereinzelt sind Code-Beispiele enthalten, welche die Verwendung der jeweiligen Funktion darstellen.
Empfehlung: Sollte eine der Funktionen des Arduino Framework unverständlich erscheinen, schlagen Sie diese zuerst in der Online-Referenz nach!

Genereller Hinweis: Speichern Sie die Lösung zu jeder Aufgabenstellung gesondert ab. Dieses erleichtert einerseits die Zuordnung von Aufgabe zu Lösung und andererseits hilft es Ihnen bei der Bearbeitung von Aufgaben, die teilweise über Aufgabenblätter hinweg aufeinander aufbauen. Berücksichtigen Sie bei der Entwicklung der Lösungen auch eine spätere Wiederverwendung des Codes. **Kommentare im Code sind ausdrücklich erwünscht!**

Die grundlegende Programmstruktur eines Arduino Sketches zeigt folgendes Beispiel:

```
// Deklaration/Definition von Variablen mit globaler Sichtbarkeit

uint8_t pin_led = 13;           //Nr. des ext. Pins
bool ledState = LOW;            //Zustandsvariable

void toggle_ledstate() {
  ledState = !ledState;
}

void setup()
{
  // Anweisungen der einmaligen, initialen Konfiguration des Mikrocontrollers

  // Beispiel: Konfiguration des digitalen I/O Anschlusspins
  // mit der Nummer 13 (hier: als Ausgangspin)
  pinMode(pin_led, OUTPUT);
  digitalWrite(pin_led, ledState);
}

void loop()
{
  // Anweisungen, die der Mikrocontroller innerhalb einer Iteration der
  // nicht terminierenden Hauptschleife ausführen soll

  // Beispiel: Wechsel des Pegels am Ausgangspin 13 (400ms-on; 200ms-off)
  toggle_ledstate();
  digitalWrite(pin_led, ledState);
  delay(400);
```

```
  toggle_ledstate();
  digitalWrite(pin_led, ledState);
  delay(200);
}
```

Listing 1: Programmstruktur eines Arduino Sketches

Folgende Funktionen werden in diesem Sketch verwendet:

```
* pinMode(⟨pin⟩, ⟨mode⟩)                    → pinMode
* digitalWrite(⟨pin⟩, ⟨value⟩)              → digitalWrite
* delay(⟨ms⟩)                              → delay
```

Aufgabe 0.1 Installation Arduino-IDE und Proteus8

Ziel dieser Aufgabe ist die Bereitstellung der Voraussetzungen zur Bearbeitung der weiteren Aufgaben der Übung. Hierfür benötigen sie die Arduino-IDE und den Proteus 8 Simulator. Die Arduino Software wird für alle gängigen Plattformen bereitgestellt, während Proteus 8 nur unter einem Windows-Betriebssystem (Win7 aufwärts) lauffähig ist.

- Windows-System:
  Überspringen Sie den folgenden Abschnitt zur Einrichtung einer Windows-Guest-Machine in einer VM auf einem Linux-System und fahren Sie mit dem Abschnitt Arduino-IDE fort.
- Linux-System:
  Sollte auf Ihrem Rechner kein Windows-Betriebssystem installiert sein, bleibt Ihnen beispielsweise die Möglichkeit, eine virtuelle Maschine auf Ihrem Host-System einzurichten:
  - Als freie Software ist beispielsweise VirtualBox von ORACLE verfügbar. Laden Sie sich VirtualBox für Ihren Host herunter und richten Sie es ein. Unter Ubuntu ist VirtualBox bereits in den offiziellen Ubuntu-Paketquellen enthalten und kann direkt installiert werden.
  - Vergessen Sie bei VirtualBox nicht, das *VirtualBox Extension Pack* zu installieren; es wird u. a. für den Zugriff auf die USB-Schnittstellen benötigt.
    Laden Sie hierfür ebenfalls von der VirtaulBox-Website das *Oracle VM VirtualBox Extension Pack* herunter und installieren Sie es.
    Starten Sie VirtualBox und wählen Sie unter *Werkzeuge–>Erweiterungen–>Installieren* das zur installierten VirtualBox-Version heruntergeladene Extension Pack (*Oracle_VM_VirtualBox_Extension_Pack-x.x.xx-x.vbox-extpack*) zur Installation aus.
  - Installieren sie Windows als Guest-Maschine. Microsoft bietet eine kostenlose, allerdings auf bis zu 90 Tage begrenzte, Win11-Entwicklungsumgebung als Disk-Image für verschiedene Virtuelle Maschinen zum Download an. Um ein Disk-Image zu installieren, gehen Sie folgendermaßen vor:
    - entpacken Sie das heruntergeladene Disk-Image
    - starten Sie VirtualBox und klicken auf *Werkzeuge–>Willkommen–>Importieren*
    - wählen Sie für `Quelle`: *Lokales Dateisystem*

* und geben Sie unter Datei: *Pfad\WinDev$_{ddmm}$Eval.ova* ($_{ddmm}$ – Tag und Monat der Bereitstellung) den Pfad zur entpackten heruntergeladenen WindowsVM.ova Datei (Open Virtual Applications; ein tar-Archiv eines OVF-Directories (Open Virtualization Format)) an.

Damit der gesamte Virtuelle Desktop dargestellt wird, können Sie die Anzeige in den *Skalierten Modus* versetzen (Menü: *Anzeige–>Skalierter Modus* oder *Host+C*, wobei die auf der rechten Seite platzierte *Strg*-Taste als *Host*-Taste konfiguriert ist.
Hinweis: Sollte nach Start der Win11-Guest Machine dessen Fenster schwarz dargestellt bleiben und die Maschine scheinbar hängen, könnte es an der 3D-Beschleunigung der Anzeige liegen, die bei VirtualBox 7 standardmäßig aktiviert ist. Fahren sie die Guest Machine herunter und deaktivieren Sie die 3D-Beschleunigung unter *WinDev$_{xxxx}$Eval–>Ändern–>Anzeige–>Erweitert*.

- Arduino-IDE:
  Folgen Sie der Anleitung auf der Arduino-Website und installieren sie die aktuelle Version. Hier steht die letzte Version der Arduino IDE 1.8.X, die Version 1.8.19, oder die aktuelle Version der Arduino IDE 2 zur Verfügung. Beide Varianten sind für die ES-Aufgaben gleichermaßen geeignet. Der Vorteil der IDE 2 liegt in der Möglichkeit des Hardware-Debuggings, was allerdings Boardseitig ein ST-Link Interface voraussetzt, welches der verwendete Arduino DUE nicht zur Verfügung stellt. Zudem wird derzeit die IDE 2 auch durch die aktuelle Proteus8 Version 8.17 (s.u.) noch nicht vollständig unterstützt. Deshalb sollte für Proteus8 die Arduino IDE 1.8.19 installiert werden.

  Stellen Sie sicher, dass sowohl das Arduino MEGA2560-Board als auch das Arduino DUE-Board unterstützt wird (Tools–>Board–>Board Manger: Arduino AVR Boards und Arduino ARM (32-bits) Boards).

- Simulator Proteus8:
  Laden Sie sich die Proteus-Installationsdatei proteus8.12.SP0.exe herunter und folgen Sie der Anleitung. Bei der Abfrage der Adresse des Lizenz-Servers geben Sie bitte ein: *licensing.labcenter.com:8884/pcls/UniHamburg/* Sollten Sie die Adresse per copy & paste übertragen, achten Sie bitte darauf, dass Sie keine führenden oder abschließende Leerzeichen versehentlich mit einfügen. Überprüfen Sie – insbesondere bei Nutzung von Windows als Gastmaschine in einer VM – bitte auch, dass die aktuelle Zeit zur eingestellten Zeitzone passt.
  Beim Start des Simulators wird zum Auschecken einer Lizenz ein Benutzername und Passwort abgefragt. Diese Daten werden den eingeschriebenen Teilnehmern der Vorlesung/Übungen zu Beginn der Veranstaltung per Mail mitgeteilt.
  Proteus8 erwartet eine Arduino-Installation unter *C:\Program Files (x86)\Arduino* (Arduino IDE 1). Gegebenenfalls müssen Sie unter *P*roteus8 bei geöffnetem *S*ource-Code Tab (Firmwareprojekt, siehe Aufg. 0.2) unter *S*ystem Settings–>Compilers Configuration den Pfad für *A*rduino AVR entsprechend anpassen (Standardpfad: *C:\Program Files (x86)\Arduino*).

Aufgabe 0.2 Einarbeitung in Arduino-IDE und Proteus-VSM

Machen Sie sich mit der Arduino-IDE und dem Proteus8-Simulator vertraut. Nutzen Sie hierzu die Einführungen auf der Arduino-Website (`www.arduino.cc/en/Guide/ArduinoUno`) und auf den Labcenter Seiten (`www.labcenter.com/tutorials/`, alternativ auch Proteus Tutorials, wobei für die Übungen der Abschnitt *VSMTutorial (Graphs)* weitestgehend irrelevant ist und der Abschnitt über *PCB-Layout (Printed Circuit Board)* gar nicht benötigt wird).
Für erste Tests steht für Sie ein Arduino-Sketch (siehe auch Listing: 1) und für die Simulation jeweils ein Proteus Projekt mit integriertem Code, ein sog. Firmwareprojekt (siehe Abb. 1) und ein Projekt mit extern generierter Firmware (eben extern, mittels Arduino-IDE generierte Firmware, siehe Abb. 2) zum Download bereit.

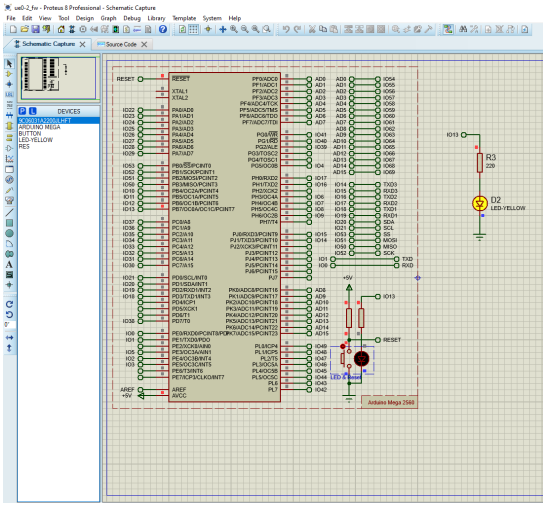

Abbildung 1: Proteus Projekt: ue0-2_fw.pdsprj
Firmware: Proteus
Verdrahtung: Namenszuordnung

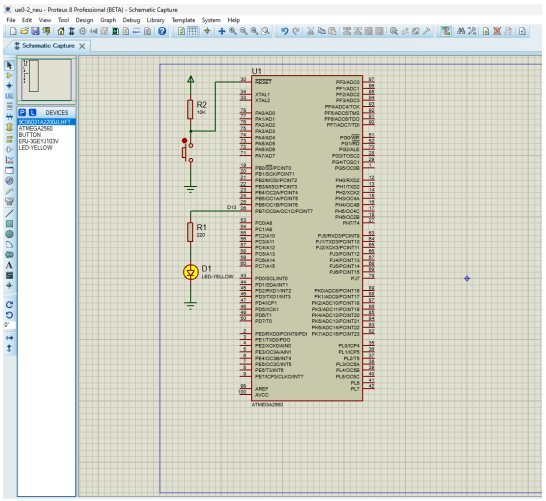

Abbildung 2: Proteus Projekt: ue0-2.pdsprj
Firmware: Arduino-IDE
Verdrahtung: explizite Verbindungen

Die Entwicklung eines Sketches sollten sie grundsätzlich mit der Arduino-IDE beginnen. Auf diese Weise erhalten sie einen syntaktisch korrekten Sketch, der, sollten keine semantischen Fehler vorliegen, auf dem gewählten Arduino-Board lauffähig sein sollte und auch die gewünschte Wirkung erzielt.

Dieser Sketch kann

- entweder als Source-Code in ein Proteus Firmware-Projekt (siehe Abb. 1), z. B. `ue0-2_fw.pdsprj`, einfließen (empfohlene Vorgehensweise),
- oder aber die mit der Arduino-IDE erzeugte Firmware — der Code, der im Falle eines realen Hardwareaufbaus per Upload in den Prozessor geladen wird, jetzt aber explizit in eine Datei geschrieben werden muss (`Sketch->Export Compiled Binary`) — für Projekte genutzt werden, die direkt diese erzeugte Firmware verwenden (siehe Abb. 2). Hierzu führen Sie einen Doppel-Klick auf den Prozessor im Scematic des Projektes aus, und geben in dem sich öffnenden Kontext-Menü unter `PROGRAM FILE` die im Hex-Format erzeugte Binärdatei an.

Der Arbeitsablauf wäre jetzt ähnlich wie bei einem echten Hardwareaufbau: Der Prozessor wird mit (virtuellen) Patchkabeln mit den Komponenten verbunden, die Software wird mit Hilfe der Arduino-IDE entwickelt und die erzeugte Firmware in den Prozessor geladen. Die in den Simulator integrierten Möglichkeiten zum Debuggen des Codes stehen wie bei der realen Hardware jetzt nicht aber nicht mehr zur Verfügung!

Validieren Sie den Blink-Sketch mittels eines Proteus Firmware-Projektes.
Als erste praktische Aufgabe des Aufgabenblattes 01 werden Sie den Sketch im realen Aufbau testen.