Hochschule RheinMain
Fachbereich DCSM - Informatik
Marcus Thoss, M.Sc.
Prof. Dr. Robert Kaiser
Christian Neugebauer

Hardwarenahe Programmierung I

WS 2021/22
LV 1512

Übungsblatt 5

Für dieses Aufgabenblatt (und die folgenden) erhalten Sie ein echtes Mikrocontroller-Board, das ein von Ihnen aus C-Quelltext erzeugtes Programm ausführen soll. Als Hardware wird ein Arduino Leonardo verwendet, der mit dem Prozessor ATmega 32U4 von Microchip (ehemals Atmel), einem Verwandten des zuvor von Ihnen im Simulator genutzten ATmega 16 und einer USB-Schnittstelle für Stromversorgung, Kommunikation und Programm-Download ausgestattet ist. Links mit Dokumentation dieses Boards und des Prozessors finden Sie unter „Weitere Unterlagen“ auf der Webseite der Lehrveranstaltung (Link).

Die Arbeit mit einem echten „Target“ erfordert durch den Download des Programms auf das Board einen zusätzlichen Schritt in Ihrem Entwicklungszyklus:

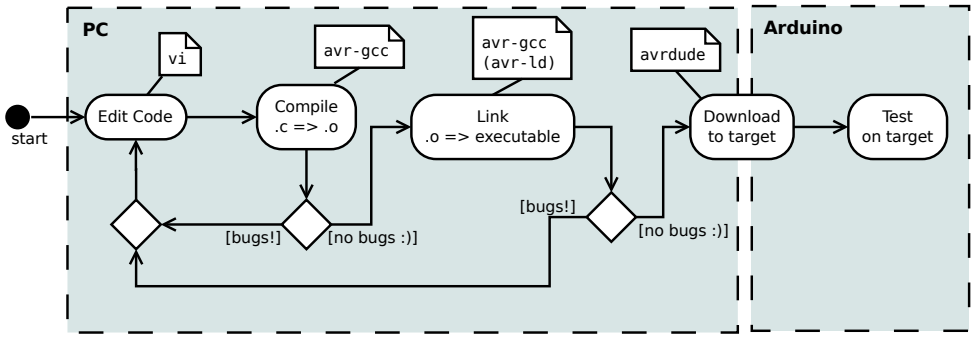

Abbildung 1: Build-Zyklus mit „echter“ Hardware

Wie sie sehen, benötigen Sie für den zusätzlichen Schritt „Download to target“ das Programm `avrdude`. Sollte es auf Ihrem System noch nicht installiert sein, müssen Sie das nun nachholen, z.B. durch Installation des Pakets `avrdude` auf Ubuntu-Systemen.

Aufgabe 5.1 (Das erste Programm auf „echter“ Hardware):

a) Laden Sie von der Materialienseite der Lehrveranstaltung das Beispielprogramm `hwp1_p9.c` in ein neues Übungsverzeichnis herunter.

b) Sehen Sie sich das Programm im Quelltext an, um es in allen Details zu verstehen. Recherchieren Sie unbekannte Informationen (wie `_delay_ms()` oder `PORTC` auf `man`-Seiten oder im Benutzerhandbuch zum ATmega 32U4).

c) Den Hauptteil des Programms müssen Sie selbst noch als Quelltext in der `main()`-Funktion ergänzen: Ihr Programm soll die orangefarbene LED auf dem Board (*unter* dem Display, Aufdruck „L“) im Sekundentakt blinken lassen.

   - Zu Beginn von `main()` muss die LED-Ansteuerung mit einem Aufruf der Funktion `userLedSetup()` vorbereitet werden.
   - Danach soll in einer Endlosschleife die LED abwechselnd an- und ausgeschaltet werden, dazwischen jeweils eine kleine Pause, damit das Blinken im Sekundentakt (1 Hz) erfolgt.
   - Verwenden Sie `while()`, `setUserLedOn()`, `setUserLedOff()` und `_delay_ms()`, um das Programm zu vervollständigen.

d) Erzeugen Sie zuvor durch einen Compile- und einen Linkschritt ein ausführbares Programm (nun wieder mit `avr-gcc`)) daraus. Achten Sie darauf, dass *keine* Warnungen (Option `-Wall` immer verwenden!) oder Fehler beim Compilieren übrig bleiben. Haben Sie die Parameter herausgefunden, schreiben Sie ein passendes Build-Skript oder `Makefile`.

   Hinweis: Der CPU-Typ ist nun `atmega32u4` (nicht mehr `atmega16`); passen Sie die `gcc`-Parameter entsprechend an. Weiterhin *müssen* Sie `-O1` (nicht `-O0`) als Parameter mit angeben, da sonst `_delay_ms()` nicht richtig funktioniert - die Erläuterung hierfür lässt sich allerdings erst später im Detail verstehen.

e) Um das Programm auf die Hardware zu bringen, schließen Sie zunächst das Board via USB-Kabel an den PC an. Die grüne LED auf dem Arduino und die rote LED auf dem Display leuchten, sobald Betriebsspannung anliegt.

f) Laden Sie nun Ihr Programm mittels `avrdude` auf die Hardware. Hierfür müssen Sie kurz vor dem Download den Reset-Button (neben der USB-Buchse) drücken, worauf die orangefarbene LED für ca. 5 s „weich“ blinkt.

   Der Programmer-Typ ist dabei `avr109`, der Connection Port `/dev/ttyACM0` und den Typ der CPU kennen Sie bereits.

   Finden Sie selbst die Flags für diese drei Parameter für `avrdude` heraus! Als vierten Parameter müssen Sie dann noch `-Uflash:w:`*`<Name ihres Programms>`* angeben.

   Sollte der Port `/dev/ttyACM0` nicht akzeptiert werden, kann es sein, dass durch Trennen und erneutes Verbinden des Targets die Nummer des Ports hochgezählt wurde – versuchen es Sie dann mit `/dev/ttyACM1` oder sehen Sie mittels `lsusb` nach, an welchem Port der Arduino momentan angeschlossen ist.

   Für den Notfall liegt auf der Webseite der Lehrveranstaltung ein vorbereitetes Programm `hwp1_p9`, das Sie testweise statt Ihres eigenen Programms downloaden können.

   Ein Debugging des Programms ist in dieser Übung übrigens noch nicht vorgesehen, hierfür benötigen Sie zusätzliche Hardware. Bei Problemen müssen Sie sich zurzeit noch mit Überlegungen zur Fehlerursache, Code-Analyse und -Korrektur und wiederholtem Download behelfen. Wenn Sie die LED zum Leuchten gebracht haben, können Sie auch Blinkmuster als Debugging-Hilfe einsetzen! Ihre Betreuer stehen natürlich immer für Fragen zur Verfügung, wenn Sie gar nicht mehr weiter wissen.

g) Schließlich - wie immer - experimentieren Sie mit den neuen Möglichkeiten und erweitern und ändern Sie das Programm nach eigenen Ideen (wie wäre es mit Blinken der Display-Beleutung oder Morsecode..?)