Mobiles Hardware-Praktikum 2023

P. Scholl und M. Pfeifer, Professur für Rechnerarchitektur,
Institut für Informatik, Albert-Ludwigs-Universität Freiburg, Freiburg im Breisgau

Übungsblatt 1 (26 Punkte)

Willkommen

Herzlich Willkommen zum Hardware-Praktikum! Im Laufe des Praktikums werden wir Ihnen schrittweise die Hardware des *S-Trike* Roboters vorstellen. Die ersten Übungsblätter werden sich auf die Microcontroller (Arduino + ESP8266) konzentrieren. Daran anschließend wird das FPGA eingeführt und programmiert. Abschließend wir ein Projekt gemeinsam auf dem ESP und FPGA entwickelt (Hardware/Software Codesign).

Ziel:

Mit Hilfe des Microcontrollers werden verschiedene Komponenten auf dem Roboter angesprochen und ausgelesen.

Prolog:

Laden Sie die Arduino IDE von der Arduino Website[1] herunter und installieren sie diese. Wählen Sie unter *Tools → Board* den Eintrag *Arduino Pro or Pro Mini*, und unter *Processor* den Eintrag *Atmega328 (3.3V, 8Mhz)* aus. Verbinden Sie dann den USB-Programmer mit dem Microcontoller-Board (Stiftleiste an der Stirnseite).

Windows: Um den Treiber für den USB-Programmer zu installieren, öffnen Sie den Gerätemanager und suchen sie den *USB Serial Port* (vermutlich unter *Andere Geräte*). Klicken Sie mit der rechten Maustaste auf den *USB Serial Port* und wählen Sie *Update Driver Software*. Den benötigten Treiber finden Sie im Arduino Ordner (Unterordner *drivers*). Nach der Installation sollte der *USB Serial Port* unter *Ports (COM & LPT)* aufgeführt werden. Falls nicht, versuchen Sie einen Neustart.

Linux (Ubuntu): Der USB-Programmer sollte automatisch erkannt werden (*Ltd FT232 USB-Serial (UART) IC*). Allerdings ist der Zugriff nur für Benutzer in der Gruppe *dialout* möglich. Fügen Sie daher Ihren Benutzer zu dieser Gruppe hinzu und starten Sie neu um den Arduino programmieren zu können.

macOS: Es ist keine Treiberinstallation notwendig. Wählen Sie den Port mit dem Name/Pfad: */dev/cu.usbserial-A<XXXX>*.

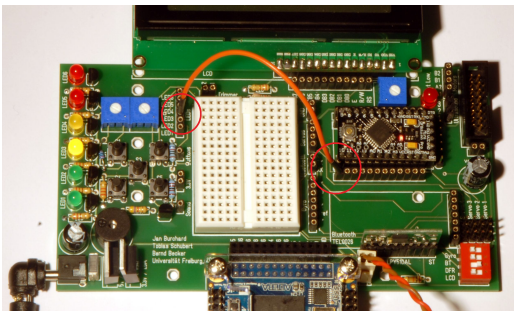
Der Arduino wird mit einer C-ähnlichen Sprache programmiert. Jedes Programm besteht dabei aus einer *setup()* und einer *loop()* Funktion. Der Code in der Setup-Funktion wird einmalig nach dem Einschalten, Resetten oder Programmieren ausgeführt. Danach wird die Loop-Funktion in einer Endlosschleife aufgerufen.

Wichtig: Schalten Sie den Roboter ein (Schalter am hinteren Rad). Schalten Sie ebenfalls den Microcontroller über den roten DIP-Schalter (“uC”) ein.

[1] http://arduino.cc

Aufgabe 1 (5 Punkte):

Verbinden Sie eine der LEDs mit Pin 10 Ihres Arduino-Boards.



Lassen sie die LED mit einer Frequenz von 1Hz blinken (die LED ist dabei für eine Sekunde an und für eine Sekunde aus). Verwenden Sie hierzu:

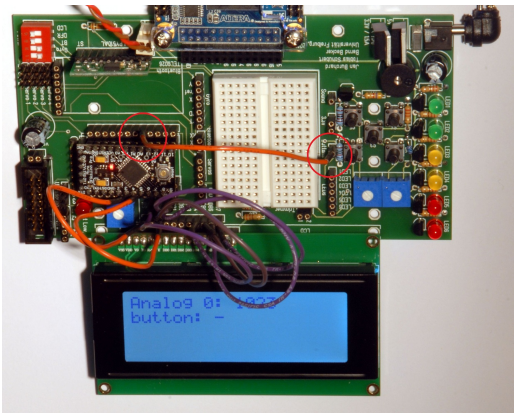
i) die *delay()*-Funktion (1 Punkt)

ii) die *millis()*-Funktion und realisieren Sie damit einen Software-Timer (2 Punkte)

Überlegen Sie sich, was der Vorteil von Methode ii) ist. Schreiben Sie als Antwort dazu einen kurzen Kommentar im Code. (2 Punkte)

Aufgabe 2 (4 Punkte):

Öffnen Sie nun die Datei "displayTest.ino" und verbinden Sie das LCD wie in der Datei beschrieben mit dem Microcontroller. Schalten Sie das LCD ein und übertragen Sie die Datei auf den Microcontroller. Sie sollten nun den Schriftzug "are you ready?" auf dem LCD sehen.



Geben Sie nun den aktuellen Wert des Analog-Pins A0 auf dem Display aus. Die Ausgabe auf dem Display soll dabei in der ersten Zeile stehen und wie folgt aussehen: "Analog 0: xxxx" (wobei xxxx dem aktuellen Wert von A0 entspricht). Versuchen Sie nur die notwendigen Teile des Displays neu zu beschreiben, um ein Flackern des Displays zu vermeiden.

Aufgabe 3 (3 Punkte):

Berechnen Sie aus dem ADC-Wert den tatsächlichen Spannungspegel, der an Pin A0 anliegt und geben Sie diesen mit zwei Nachkommastellen in der ersten Display-Zeile aus. Die Ausgabe soll damit die folgende Form haben "Analog 0: x.xxV"

Aufgabe 4:

Verbinden Sie die das Potentiometer Tr1 ("Trimmer"-Buchse auf dem Experimentierboard) mit dem Analog-Pin A0. Je nachdem, in welcher Stellung sich der Poti befindet, sollte ein anderer Wert an A0 angezeigt werden.

Aufgabe 5 (3 Punkte):

Verbinden Sie nun die Taster (“Buttons”-Buchse auf dem Experimentierboard) mit dem Analog-Pin A1. Je nachdem, welchen Taster Sie drücken, liegt ein andere analog Wert an A1 an. Sie können den ADC Wert zum finden der Grenzen provisorisch in der zweiten Zeile ausgeben. Nachdem Sie die Grenzen gefunden haben, geben Sie den aktuell gedrückten Taster in der zweiten Zeile des Displays wie folgt aus: “button: Sx” (wobei x der gedrückte Taster ist bzw. “–”, wenn kein Taster gedrückt ist).
Beachten Sie dabei, dass der exakte Analog-Wert des Tasters von Umgebungsvariablen wie der Temperatur, und der Akkuspannung abhängig sein kann und programmieren Sie die Erkennung der gedrückten Taste mit entsprechenden Toleranzen.

Aufgabe 6 (2 Punkte):

Wie viele andere Microcontroller auch, verfügt der Arduino über eine UART Schnittstelle. Diese kann dazu genutzt werden, zwischen zwei Systemen Daten zu übermitteln. Dies geschieht über zwei Leitungen (*RX* und *TX*) die jeweils über kreuz angeschlossen werden (*SystemA-RX* an *SystemB-TX* und *SystemA-TX* an *SystemB-RX*). Über diese Schnittstelle wird der Arduino ebenfalls programmiert bzw. mit dem PC verbunden. Bei einer UART Verbindung müssen Sender und Receiver die benutzte Übertragungsgeschwindigkeit kennen. Diese wird in *Baud* angegeben und entspricht der Geschwindigkeit die benötigt wird um ein Bit zu übertragen. Die wohl gängigste Baudrate ist 9600. Damit wird ein Bit in 1/9600s übertragen. Sie können UART auch dazu nutzen, Status oder Debug Nachrichten zu übermitteln. Diese können mit dem seriellen Monitor der Arduino IDE gelesen werden. Der serielle Monitor kann über das Lupensymbol rechts oben geöffnet werden.
Initialisieren Sie die UART Verbindung mit 9600 baud, und geben Sie auf der seriellen Konsole die jeweilig gedrückte Taste aus. Wird keine Taste gedrückt, soll nichts ausgegeben werden.

Aufgabe 7 (5 Punkte):

Verbinden Sie die LEDs der Reihe nach mit den Pins 2 (LED1), 3, 4, 5, 6, 7 (LED6). Verändern Sie Ihren Code dahingehend, dass mit dem Taster “S1” eine der drei Farben und mit dem Trimmer “Tr1” eine Blinkfrequenz linear zwischen 1 Hz und 50 Hz gewählt werden können. Die aktuelle Einstellung (Farbe + Frequenz) soll in der dritten und vierten Zeile des LCDs angezeigt werden.

Hinweis:

- Informieren Sie sich zu dem Vorgang des “Entprellens” und entprellen Sie die Taster dementsprechend, sodass ein Tastendruck jeweils nur zu einer Farbänderung führt.
- Um die Blinkfrequenz linear zu mappen, können Sie die Arduino eigene *map* Funktion benutzen.

Aufgabe 8 (4 Punkte):

Erweitern Sie das Programm aus Aufgabe 7 um eine Lauflicht. Wenn der Taster “S1” gedrückt wird, soll das Blinken stoppen und stattdessen ein Lauflicht angezeigt werden. Das Lauflicht soll zurück “bouncen” also von LED1 zu LED6 und wieder zurück zu LED1 (usw.) laufen. Die Geschwindigkeit des Lauflichts soll wieder an “Tr1” eingestellt werden können. Der Wechsel zur nächsten LED in der Reihe soll dabei der Periodendauer des Blinkens aus Aufgabe 7 entsprechen.

Hinweis:

Nutzen Sie die verbleibende Zeit in dieser Woche und beginnen sie sich die Vorlesungen zu Arduino und VHDL anzuschauen. Außerdem sollten Sie Quartus Prime installieren und das Quartus Tutorial bis zum Ende des dritten Übungsblattes durchgehen.

Abgabe:

Archivieren Sie Ihre Programme in einer ZIP-Datei und laden Sie diese im Übungsportal hoch. Aufgabenteile die durch weitere Aufgaben verändert oder herausgenommen wurden, sollten auskommentiert werden. Ihre Abgabe sollte 2 Programme enthalten: Aufgabe 1 und Aufgabe 2 - 8. Überprüfen Sie die Archiv-Datei auf Vollständigkeit.