Vorschläge für kleine Bastel-Projekte im Rahmen des Arduino-Workshops

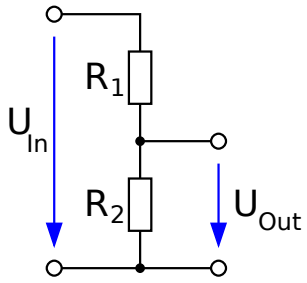
Die folgenden Aufgaben sind Vorschläge für kleine Projekte, mit denen man seinen Arduino, dessen Entwicklungsumgebung und die verschiedenen Ein- und Ausgabe-Funktionen ausprobieren und besser kennenlernen kann. Die Aufgaben haben verschiedene Schwerpunkte (Elektronik, E/A, Programmierung) und sind auch unterschiedlich aufwendig.
Es gibt jeweils nicht nur eine "richtige Lösung" und jedes dieser Mini-Projekte kann auf unterschiedliche Weisen umgesetzt und erweitert und nach deinen Ideen angepasst werden. Die Aufgaben nennen jeweils nur das Ziel und ein paar erste Hinweise, wie man bei der Umsetzung vorgehen könnte.
Viele der Themen sind hier nur angerissen. Zu jedem der Themen findest du im Internet sehr leicht ausführlichere Informationen. Die Projekte lassen sich auch sehr gut zu zweit umsetzen.
Die notwendigen elektronischen Bauteile bekommst du hier auf dem Workshop. Wenn du zu Hause weiterbasteln möchstest, kannst du diese auch am Ende des Workshop zum Selbstkostenpreis erwerben.

1 Widerstandsmeßgerät

Stichworte: Messen mit den analogen Eingängen, Spannungsteiler

Mit dem Arduino lässt sich ein einfaches Widerstandsmessgerät bauen, mit dem man die Ω-Werte von Widerständen ermitteln kann.

Die Schaltung kann man auf einem solchen Spannungsteiler aufbauen:



Ein Spannungsteiler teilt eine Eingangs-Spannung (U_{In}) über zwei Widerstände (R_1,R_2) auf. Es entsteht so eine Ausgangsspannung (U_{Out}).

Das Verhältnis zwischen Ein- und Ausgangs-Spannung ist: $U_{Out} = U_{In}\frac{R_2}{R_1+R_2}$. Bei einer Eingangsspannung $V_{In} = 5V$ und Widerständen $R_1 = 1k\Omega$ und $R_2 = 2,2k\Omega$ also $U_{Out} = 5V\frac{2,2k\Omega}{1k\Omega+2,2k\Omega} = 3,44V$.

Umgekehrt lassen sich durch das Messen von U_{Out} aber auch Rücksschlüsse über den Widerstandswert von R_2 ziehen.....

Eine detaillierte Erklärung von Spannungsteilern (engl. “Voltage Divider”) findest du im Internet, zum Beispiel in der Wikipedia.

Die gemessenen/berechneten Werte kannst du zum Beispiel im “Serial Monitor” (siehe Menue “Tools”) des Arduino ausgeben.

Material: Für diese Aufgabe benötigt du nur ein paar Kabel sowie Widerstände in unterschiedlichen Werten für den Aufbau der Schaltung und zum Messen.

Fragen:

- Wie genau sind die so gemessenen Werte? Vergleiche mit den Messwerten eines Multimeters.
- Was für Auswirkungen hat die Wahl des Widerstandes, den du für R_1 verwendest?
- Ein Multimeter hat meist unterschiedliche Messbereiche. Wie könnte man eine solche Bereichswahl der Schaltung hinzüfügen?

Die Schaltung dieser Aufgabe lässt sich gut als Basis für das Thermometer der nächsten Aufgabe verwenden.

2 Thermometer mit LED-Anzeige

Stichworte: LED-Ansterung, PWM

Das Widerstandsmessgerät der vorigen Aufgabe lässt sich zu einem elektronischen Thermometer erweitern, wenn man als zu messenden Widerstand einen temperaturveränderlichen Widerstand verwendet.

Zur Anzeige kannst du zwei LEDs verwenden. Mit steigender Temperatur zwischen einer grünen und einer roten LED umgeblendet werden. Je wärmer es wird, desto dunkler wird die grüne LED und desto heller die Rote. Das Dimmen der LEDs kannst du mit der PWM-Funktion des Arduino erreichen. Zur Kontrolle kannst du die gemessene Temeperatur natuerlich wieder zusätzlich über den Serial Monitor des Arduino ausgeben.

Material: 2 LEDs (rot/grün, 2,2Volt V_F), jeweils mit passenden Vorwiderständen; 1 Temperatursensor (inkl. Datenblatt).

Mögliche Erweiterungen:

- *Erweiterung zum Temperaturregler:*
  Temperatursensoren werden oft als Teil eines Thermostaten eingesetzt um eine Heiz- oder Kühlfunktion zu steuern. Schließe eine Glühlampe an den Arduino als “Heizung” und stelle diese dicht an den Temperatursensor, sodass dieser die Wärme der Glühlampe misst. Du kannst diese Mini-Heizung nun mit dem Arduino steuern und deren Umgebung auf einer vorgegebenen Temperatur halten.

Häufiges Ein- und Ausschalten schadet deiner Heizung. Wie kannst du verhindern, dass diese an der Temperaturgrenze "flackert", und statt dessen in längeren Zyklen an- und ausgeht?

- *Gamma-Korrektur für die LEDs:*
  Die Leuchtstärke von LEDs steigt nicht gleichmäßig mit dem Anteil im PWM-Signal. Insbesondere im unteren PWM-Bereich werden die rote und grüne Überblend-LED schon bei sehr geringen PWM-Änderungen ihre Helligkeit stark ändern. Mann kann dieses Verhalten mit einer sogenannten *Gamma-Korrektur* ausgleichen bei der lineare Helligkeitswerte so umgerechnet werden, dass sie zum Verhalten der LED passen. Füge der grün/rot-Umblendfunktion eine Gammakorrektur hinzu. Im Internet findest du umfangreiche Beschreibungen, wie dies geht.

3 Ansteuerung von 7-Segment-Anzeigen

Stichworte: 7-Segment-Anzeigen, Ansteuerung von Logik-ICs, Potentiometer

Mit 7-Segement-Anzeigen kann man seinem Arduino auf einfache Weise eine unabhängige Anzeigefunktion hinzufügen. Eine 7-Segment-Anzeige (kurz: 7SA) besteht aus 8 LEDs in einem Gehäuse: 7 davon sind so angeordnet, dass man damit eine einzelne Dezimal-Ziffer darstellen kann, ein weitere beleuchtet einen Dezimalpunkt. Man findet (bzw. fand) 7SA an vielerlei Geräten, z.B. zur Anzeige von Einstell- oder Messwerten.

Man kann eine 7SA direkt an die digitalen Ausgänge des Arduino (natürlich wieder mit Vorwiderstand) anschließen. Da dies aber 7 der Digital-Ausgänge belegen würde, verwendet man meist eine separate Ansteuer-Logik um Ausgänge zu sparen oder mehr 7SA anzusteuern als der Arduino Ausgänge hat.
Hierfür kann man zum Beispiel ein Schieberegister verwenden.

Schließe an deinen Arduino an einen der analogen Eingänge einen Potentionmeter (regelbarer Widerstand) und eine 7SA in direkter Beschaltung an 7 digitale Ausgänge an. Die 7SA kann nun Zahlen oder Buchstaben anzeigen. Durch drehen am Potentionmeter soll die angezeigte Zahl bestimmt werden können. Alternativ kannst du an Stelle des Potentiometer auch einen Taster zum durchschalten der Zahlen verwenden.

Du kannst diese Anzeige-Funktion nun stückweise erweitern (die Bauteile sind hier beim Workshop vorhanden):

- *Ansteuern der 7SA durch ein 74595 8-Bit-Schieberegister:*
  Schließe die Segmente der 7SA an die Ausgänge eines 74595 Schieberegisters an. Verbinde die beiden Taktleitungen und die Datenleitung mit dem Arduino.

- *Ansteuern von drei 7SA mit mehreren Schieberegistern:*
  Schließe drei 7SA an drei Schieberegister an und diese an den Arduino. Ein Schieberegister wurde über 3 Ausgänge des Arduino angesteuert. Kann

man 3 Schieberegister auch mit weniger als 9 Ausgängen ansteuern?

Füge dem Potentiometer einen Taster hinzu, der durch Druck die Auswahl der Ziffer erlaubt, die man mit dem Potentiometer verändert.

- *Flipper:*
  Die vorige Ausbau-Stufe erinnert stark an die Highscore-Eingabefunktion eines Flippers. Füge die restlichen Teile des Flippers hinzu und steuere diesen mit deinem Arduino. (Die notwendigen Teile haben wir auf dem Workshop allerdings leider nicht vorrätig.....)

Material: 7-Segment-Anzeigen, Potentiometer, 74HC595 Schieberegister, Kabel, Taster

Die Ansteuerung von Schieberegistern hat starke Ähnlichkeit mit dem "Serial Peripherial Interface" (SPI). Ein Protokoll auf diese Weise durch direkte Manipulation von Ein- und Ausgängen "per Hand" zu sprechen, nennt man auch "Bit banging". Der Arduino besitzt hierfür allerdings auch eine Bibliothek (siehe letzte Aufgabe).

4 Befehls-Verarbeitung über serielle Schnittstelle

Stichworte: Serielle Kommunikation, Befehls-Verarbeitung

Viele Geräte (z.B. Modems, Netzwerk-Switches, manche Messgeräte) stellen über eine serielle Verbindung eine einfache Befehls-Schnittstelle bereit, über die man das Gerät konfigurieren, Werte auslesen oder Aktionen anstoßen kann.
Eine solche Schnittstelle kann mit der seriellen Bibliothek des Arduino implementiert werden.

Schreibe einen Arduino-Sketch der Text-Befehle (d.h. Befehlswort und dann ENTER) über die serielle Schnittstelle einliest und eine LED entsprechend ein- und ausschaltet (LEDON, LEDOFF).

Du hast hier zwei unterschiedliche möglichkeiten. Entweder, du schreibst die notwendigen Programmfunktionen selbst, oder du greifst auf die avr-libc Bibliothek zurück (http://www.nongnu.org/avr-libc/), die viele nützliche Funktionen bietet. Sowohl die allgemeinen Funktionen einer C-Standardbibliothek als auch AVR-spezifische Funktionen, zum Beispiel zur Ansteuerung der integrierten Schnittstellen des atmega328p-Mikrocontrollers (http://www.nongnu.org/avr-libc/user-manual/modules.html).
Du kannst zur Kommunikation mit dem Arduino wieder den Serial Monitor verwenden.

Wichtiger Hinweis: Schliesse die serielle Schnittstelle des Arduino nie direkt an die serielle Schnittstelle eines PC an. Der Arduino arbeitet mit 5 Volt, der PC mit 12 Volt. Die Übertragung wird nicht funktionieren und du läufst in Gefahr, die Eingänge deines Arduino zu beschädigen! Beim Anschluss an eine

serielle Schnittstelle muss ein sog. Level-Shifter dazwischengeschaltet werden, der zwischen den unterschiedlichen Spannungen vermittelt.

Diese erste Stufe kann nun wieder schrittweise erweitert werden:

- *Einlesen von Befehlsparametern:*
  Lies zusätzlich zu einem Befehl einen einzelnen Parameter ein und erlaube so das setzen einer 7SA oder des PWM-Wertes einer LED (z.B. "SET7SA 3", oder "PWM 50").
- *Steuerung der Befehls-Schnittstelle mit einem zweiten Arduino:*
  Tue dich mit einem anderen Workshop-Teilnehmer zusammen und schreibt gemeinsam einen Client für die Befehlsschnittstelle. Der Client erhält einen Potentiometer oder Taster, liest diesen aus und sendet einen entsprechenden Befehl an den anderen Arduino. So lässt sich z.B. mit dem Potentiometer am einen Arduino die 7SA-Anzeige am anderen Arduino steuern.

5 IC-Ansteuerung mit SPI

Viel Spezial-ICs wie z.B. Sensoren, Speicher, I/O-Bausteine können über die seriellen Protokolle "Serial Peripherial Interface" (SPI) oder "Inter-IC Protocol" (I2C) angesprochen werden.
Das SPI-Protocol ähnelt stark der Ansteuerung eines Schieberegisters, wie es in der vorletzten Aufgabe verwendet wurde. Man muss das Protokoll aber nicht immer "per Hand" implementieren, da der Mikrocontroller des Arduino hierfür eine Hardware-Unterstützung bietet und die Arduino-Programmierumgebung das SPI-Protokoll als Teil der Standard-Bibliothek.
SPI ist ein asymmetrisches Protokoll: Ein "Master" steuert einen "Slave". Die Arduino-Bibliothek ermöglicht das Ansteuern von ICs als SPI-Master. Eine Slave-Funktion ist standardmäßig leider nicht vorhanden, es gibt aber Implementationen von Dritten im Internet zu finden.
Eine Beschreibung des SPI-Protokolls findet sich z.B. in der Wikipedia oder der AVR Application Note 151 (google nach "atmel avr151").

Viele ICs können mit der SPI-Bibliothek des Arduino angesprochen werden. Für den Workshop haben wir folgende ICs zur Verfügung:

- Digital-Analog-Converter (Umwandlung digitaler Daten in entsprechende Spannnungswerte)
- I/O-Expander (um dem Arduino weitere digitale I/O-Leitungen zur Verfügung zu stellen)
- Digitale Potentiometer (Digital steuerbarer Widerstand)
- EEPROM (SPI-adressierbarer, nicht-flüchtiger Speicher)

Jeder dieser Bausteine kann in seiner vollen Funktionalität per SPI angesteuert werden und für sehr unterschiedliche Arduino-Projekte nützlich sein.