# Mobiles Hardware-Praktikum 2017

Jan Burchard, Tobias Schubert, Bernd Becker

Institut für Informatik, Albert-Ludwigs-Universität Freiburg, Freiburg im Breisgau


# Übungsblatt 1 (24 Punkte)

Abgabe: 10.05.2017 bis 17:00 Uhr

## Willkommen

Herzlich Willkommen zum Hardware-Praktikum! Im laufe des Praktikums werden wir Ihnen schrittweise die Hardware des S-Trike Roboters vorstellen. Die ersten Blätter werden sich auf den Microcontroller (Arduino) konzentrieren. Daran anschließend wird der FPGA eingeführt und programmiert. Abschließend werden dann Projekte gemeinsam auf beiden Plattformen 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. Diese muss nur entpackt werden und benötigt keine dauerhafte Installation. Öffnen Sie dann die IDE.
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 (im Unterordner "drivers"). Nach der Installation sollte der "USB Serial Port" unter "Ports (COM & LPT)" aufgeführt werden.

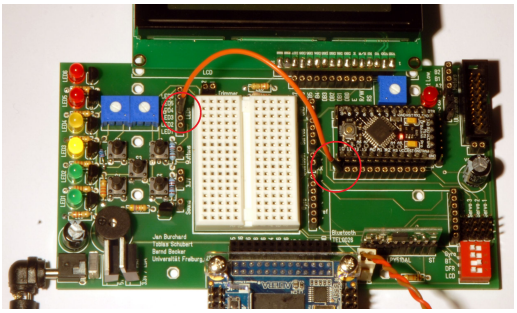
**Linux (Ubuntu + Derivatives):** Der USB-Programmer sollte automatisch erkennt werden (“Ltd FT232 USB-Serial (UART) IC”). Allerdings ist der Zugriff nur für Benutzer in der Gruppe *dialout* möglich. Fügen Sie daher Ihren Benutzer zu dieser Gruppe hinzu und starten Sie neu um den Arduino programmieren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

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

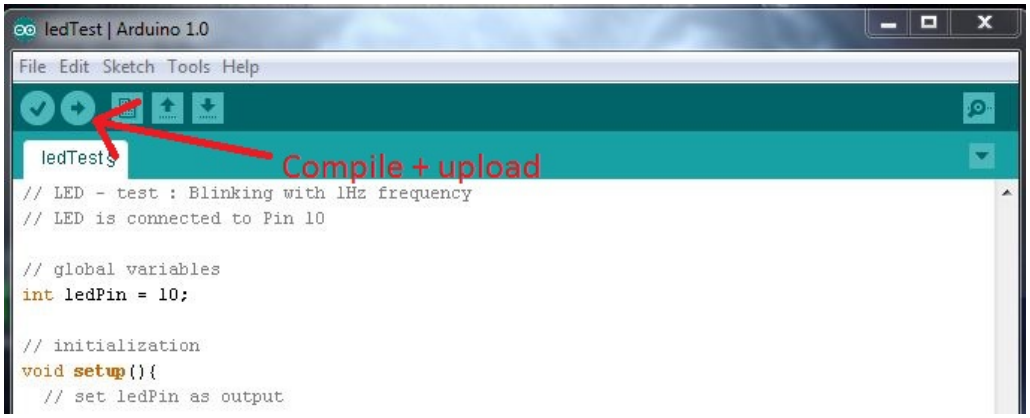
Schalten Sie den Microcontroller mit dem roten DIP-Schalter ein ("uC").

[1] http://arduino.cc

## Aufgabe 1:



Verbinden Sie eine der LEDs auf dem Experimentierboard mit dem Microcontroller (Pin 10). Öffnen Sie dann die Datei "ledTest.ino" mit der Arduino IDE. Kompilieren und übertragen Sie die Datei auf den Microcontroller.




Die verbundene LED sollte nun einmal pro Sekunde blinken.

## Aufgabe 2 (3 Punkte):

Erweitern Sie das Programm aus Aufgabe 1, so dass nun zwei LEDs immer abwechselnd blinken. Verbinden Sie dazu eine zweite LED mit Pin 11 und erweitern Sie den Code entsprechend.

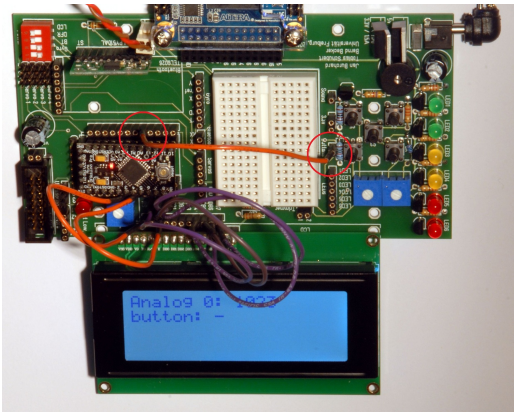
## Aufgabe 3 (4 Punkte):

Verbinden Sie eine rote, gelbe und grüne LED mit den Pins 10, 11 und 12. Implementieren Sie dann in einem neuen Arduino Sketch eine einfache Ampel mit der folgenden Signalabfolge:

$$\text{rot} \xrightarrow{5\,Sekunden} \text{rot+gelb} \xrightarrow{1\,Sekunde} \text{grün} \xrightarrow{3\,Sekunden} \text{gelb} \xrightarrow{1\,Sekunde} \text{von vorne}$$

## Aufgabe 4:

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 "system ready" auf dem LCD sehen.




## Aufgabe 5 (4 Punkte):

Geben Sie den aktuellen Wert des Analog-Pins A0 auf dem Display aus. Dieser wird im loop-Abschnitt bereits in einer Variable gespeichert. Die Ausgabe auf dem Display soll dabei in der ersten Zeile stehen und wie folgt aussehen: "Analog 0: xxx" (wobei xxx dem aktuellen Wert entspricht).

## Aufgabe 6 (4 Punkte):

Verbinden Sie die Taster ("Buttons"-Buchse auf dem Experimentierboard) mit dem Analog-Pin A0. Je nachdem welchen Taster Sie drücken, sollte ein anderer Wert an A0 angezeigt werden. Geben Sie den aktuell gedrückten Taster in der zweiten Zeile des Displays wie folgt aus: "button: xx" (wobei xx der gedrückte Taster bzw. "–" ist).
Beachten Sie, dass der exakte Analog-Wert abhängig von Umgebungsvariablen wie der Temperatur, der Akkuspannung, dem Wochentag und ähnlichem ist und programmieren Sie die Erkennung der gedrückten Taste mit entsprechenden Toleranzen.

## Aufgabe 7 (3 Punkte):

Übertragen Sie ihr Ampel-Programm in die aktuelle Datei. Geben Sie auf dem LCD in der dritten Zeile jeweils den aktuellen Zustand der Ampel an (also zum Beispiel “green”). Ihr Programm sollte sowohl die LEDs als auch das LCD ansteuern.

## Aufgabe 8 (2 Punkte):

Ändern Sie die Signalabfolge so ab, dass die Ampel den Zustand “rot” nur verlässt wenn die Taste “S1” gedrückt wird.

## Aufgabe 9 (4 Punkte):

Erweitern Sie die Ampel mit Taster um eine Tonausgabe. Bei “rot” soll ein 100 Hz Ton ausgegeben werden, bei “grün” ein 400 Hz Ton. Verbinden Sie dazu den Piepser (Buchse Sound) mit Pin 13 des Arduino.
Verwenden Sie die Arduino-Funktion “tone” um Töne auszugeben.[2] Der Ton muss nicht durchgehend ausgegeben werden, ein *tuten* ist ausreichend.

## Abgabe:

Archivieren Sie Ihre Programme in einer ZIP-Datei und laden Sie diese im Übungsportal hoch. Ihre Abgabe sollte 3 Programme enthalten: Blinkende Leds, Ampel und Ampel mit Taster und Tonausgabe. Bewertet werden Ihre Implementationen von Aufgabe 2, 3, 5, 6, 7, 8 und 9. Überprüfen Sie die Archiv-Datei auf Vollständigkeit.

[2] http://www.arduino.cc/en/Tutorial/Tone