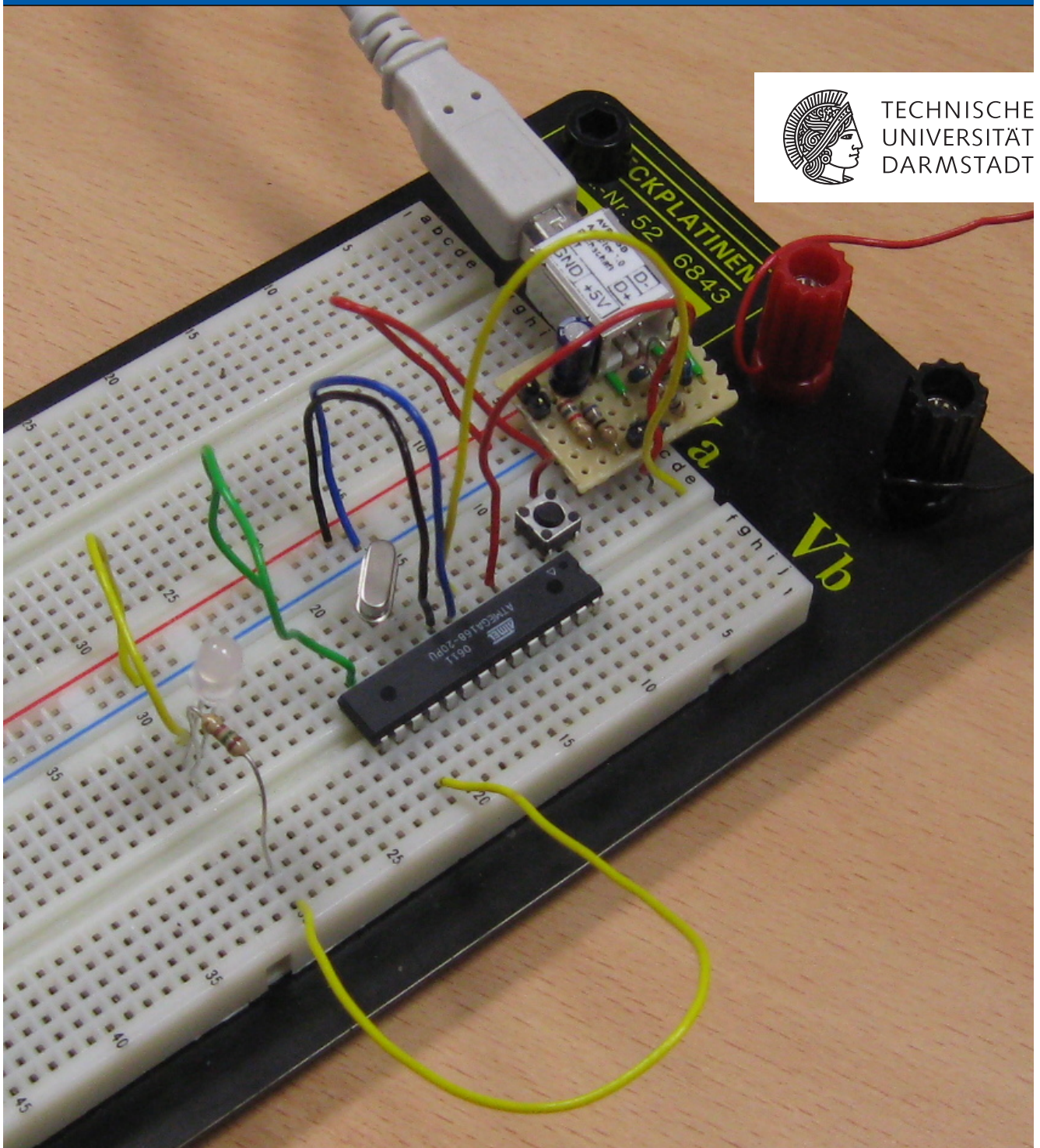


Ophasen Workshop Mikrocontroller

Fachschaft Informationssystemtechnik



TECHNISCHE
UNIVERSITÄT
DARMSTADT



1 Einführung

Es ist erstaunlich was Computer heutzutage alles leisten können und wie einfach man sie mit modernen Hochsprachen programmieren kann. Doch heute werden wir ein wenig hinter die Fassade blicken und die Vorzüge von Java und Co. erst einmal vergessen.

Vor uns haben wir einen typischen Mikrocontroller liegen, dem wir zunächst beibringen wollen eine LED blinken zu lassen (eine Art 'Hello World' der Elektronik). Dazu bauen wir uns, wie im nächsten Abschnitt beschrieben, eine kleine Schaltung auf, die wir per USB mit einem PC verbinden können. Das ganze System beruht auf der quelloffenen Arduino Plattform, einem Projekt speziell für den einfachen Einstieg in die Welt der Mikrocontroller. Zum Programmieren werden wir die in Processing geschriebene Arduino-IDE verwenden. Programmieren werden wir dann aber in C unter Hilfe der Libraries von avr-gcc und Arduino.

Alle Mikrocontroller wurden bereits mit einem Bootloader geflasht. Dieser ist in der Lage Befehle von der Arduino-IDE entgegen zu nehmen und so die kompilierten Anwendungen im eigenen Speicher abzulegen und auszuführen. Über die genaue Funktionsweise sind in dieser Einführung jedoch keine tiefgehenden Kenntnisse notwendig, das alles erledigt Arduino für uns.

1.1 Generelle Hinweise

Bitte geht sorgsam mit der Hardware um, wir möchten den Workshop auch in Zukunft noch anbieten. Wenn Ihr euch unsicher seid, fragt lieber einmal mehr bei den Tutoren nach. Lasst eurer erste Schaltung und größere Umbauten immer von den Tutoren überprüfen. Vor Umbauten immer den USB-Stecker aus dem Rechner ziehen, damit kein Strom mehr durch die Schaltung fließt.

Hingewiesen sei an dieser Stelle noch auf die Nummerierung der Pins. Hardwareseitig werden die Anschlüsse einfach von 1 bis 28, begonnen am oberen linken Pin, durchnummeriert. In Abbildung 1 sind dies die weißen Zahlen innerhalb des Controllers. Arduino verwendet für die Programmierung jedoch eine eigene Nummerierung, die sich an der Funktion des jeweiligen Pins orientiert. Diese Nummerierung ist ebenfalls Abbildung 1, anhand der schwarzen Beschriftung neben dem Controller, zu entnehmen. Beispielsweise wird ein Bauteil, welches am unteren rechten Pin (also Hardware-Pin 15) angeschlossen ist, im Programm als Software-Pin 9 referenziert.

Im Anhang auf Seite 9 werden alle verwendeten Begriffe und Bauteile auch noch kurz erklärt. Bei Unklarheiten einfach mal nachschauen, ob dort weitere Infos stehen.

1.2 Aufbau

Um unseren Mikrocontroller programmieren zu können, müssen wir die Grundschiung aufbauen, die den Controller mit unserem Programmieradapter verbindet. Zunächst stecken wir dazu den ATMEL ATMEGA168 über die Kerbe auf unser Breadboard (siehe Titelbild). Den kleinen USB-Programmieradapter stecken wir ebenfalls auf. Bei beiden Platzierungen ist auf die leitfähigen Verbindungen im inneren des Breadboards zu achten. Des Weiteren verwenden wir einen Quarz, eine Zweifarben-LED und eine Einfarben-LED jeweils mit Vorwiderstand und einige Taster. Alle Bauteile werden gemäß der folgen Liste miteinander verbunden um die Grundschiung zu erhalten. Genaueres ist Abb. 1 auf der folgenden Seite zu entnehmen. Achtet bitte darauf, dass ihr beim Ändern der Schaltung aus Sicherheitsgründen das USB-Kabel vom Rechner abzieht und eure erste Schaltung von einem der Helfer abnehmen lasst, bevor ihr sie zum ersten Mal in Betrieb nehmt.

Es sind folgende Schritte notwendig:

- Pin 1 über den Reset-Taster mit GND verbinden.
- Pin 4 mit dem USB-Connector unten verbinden.
- Pin 6 mit dem USB-Connector oben verbinden.
- Pin 7 am ATMEGA168 mit VCC am Adapter verbinden.
- Pin 8 am ATMEGA168 mit GND am Adapter verbinden.
- Pin 13 am ATMEGA168 mit GND verbinden.
- 16 MHz Quarz an Pin 9 und 10

Die für die jeweiligen Aufgaben müssen dann später noch LEDs und Taster auf das Board aufgesteckt werden. Bei LEDs ist darauf zu achten, dass diese nur mit Vorwiderstand betrieben werden. Wird der fließende Strom nicht durch einen solchen Widerstand begrenzt, führt dies zu Zerstörung der Bauteile (was selbstverständlich vermieden werden

sollte). Für solche Erweiterungen der Schaltung können am Controller die Software-Pins 8 - 13 (Hardware-Pins 14 bis 19) verwendet werden. Diese können später beliebig als digitale Ein- und Ausgang definiert werden. Für analoge Ein- und Ausgänge sind die Software-Pins 14 bis 19 (Hardware-Pins 23 bis 28) reserviert.

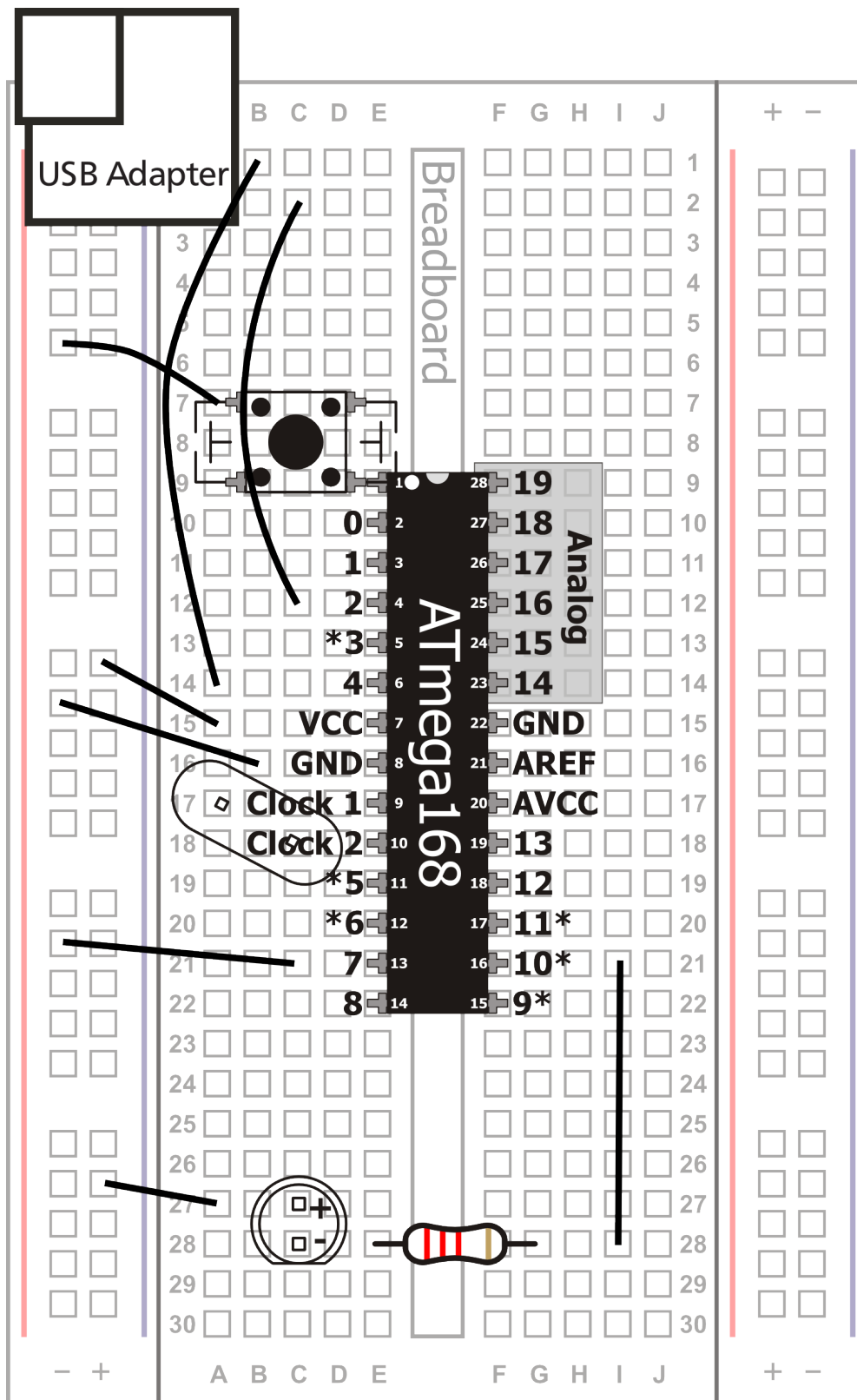


Abbildung 1: Der Schaltungsaufbau. Die äußeren, schwarzen Pin-Beschriftungen gelten für die Programmierung (Software-Pins), die inneren, weißen für den Aufbau (Hardware-Pins)!

1.3 Programmierung

Um mit der Programmierung beginnen zu können, müssen wir zuerst unsere Entwicklungsumgebung starten. Da das Arduino Programm nicht fest auf den Pool-Rechnern installiert werden kann, müssen wir es über die Konsole aus unserem Home-Verzeichnis starten. Vorher laden wir es uns vom Fachschaftsserver herunter. Dazu führen wir folgende Befehle aus:

```
1 cd ~
2 wget http://www.fs-ist.de/public/arduino/arduino.tar.gz
3 tar -xzf arduino.tar.gz
4 cd arduino
5 ./arduino
```

Wer nicht genau weiß was die einzelnen Befehle machen, kann gerne die Tutoren fragen, es ist aber nicht notwendig dies im Detail zu verstehen.

Nach dem sich die GUI geöffnet hat, können wir auch direkt anfangen unser Programm zu schreiben. Die notwendigen Einstellungen wurden bereits vorgenommen.

Grundsätzlich besteht ein Programm für Arduino aus mindestens 2 Funktionen:

- `void setup() {}` - Diese Funktion wird als Initialisierung **einmal** zu Beginn des Programms durchlaufen. Hier können bspw. die Ein- und Ausgänge des Controllers definiert werden.
- `void loop() {}` - Diese Funktion wird zur Laufzeit des Programmes **ständig** in einer Endlosschleife aufgerufen. Hier sollte die eigentliche Logik der Anwendung implementiert werden.

Im Anhang dieser Anleitung befindet sich eine kleine Referenz an Funktionen, die für diesen Workshop benötigt werden. Über die Arduino IDE kann ferner auch die Hilfe aufgerufen werden, die eine noch detailliertere Auflistung der in der Library definierten Funktionen bietet. Als Einsteiger kann das folgende Beispielprogramm verwendet werden. Es lässt eine am Pin 10 (Hardware-Pin 16) angeschlossene LED im 100ms Rhythmus blinken.

```
1 /**
2  * hier das einmalige Setup für die Pins
3  */
4 void setup() {
5   pinMode(10, OUTPUT); // setze Pin 10 auf Output
6 }
7
8 /**
9  * Diese Funktion wird in einer Endlosschleife durchlaufen
10 */
11 void loop() {
12   digitalWrite(10, HIGH); // setze Pegel von Pin 10 auf HIGH
13
14   delay(100); // warte 100ms
15
16   digitalWrite(10, LOW); // setze Pegel von Pin 10 auf LOW
17
18   delay(100); // warte 100ms
19 }
```

Nachdem das Programm fertig geschrieben wurde, lässt es sich innerhalb der Arduino-IDE mittels dem Play-Symbol kompilieren. Anschließend kann man die fertig kompilierte Anwendung mit dem Upload-Button auf den Controller übertragen. Dabei ist drauf zu achten, dass dieser vorher mit dem Reset-Taster in den dafür notwendigen Zustand gesetzt wurde. Eventuell auftretende Fehler sind im unteren Ausgabefenster zu sehen.

1.4 Einstiegsaufgabe - Blink

Baut wie in der Einführung beschrieben die Grundschaltung auf eurem Steckbrett auf. Schließt die LED mit Vorwiderstand an einem der digitalen Ein- und Ausgangspins an und bringt sie zum blinken. Versucht dabei jede Zeile des Programmcodes nachzuvollziehen.

Steckt nun die LED an einen anderen Ausgangspin, was müsst ihr am Quelltext verändern, damit das Programm weiterhin läuft?

2 Blink EXTREME: Die Zwei-Farben-LED

Wir werden nun eine Steuerungssoftware programmieren, die die eingesetzte LED in zwei Farben leuchten lässt. Diese Zwei-Farben-LED besteht aus zwei internen LEDs. Je nach Polarität der anliegenden Spannung blockiert eine der LEDs, während die andere LED den Strom leitet und leuchtet.

Ist die Zwei-Farben-LED mit dem Vorwiderstand an zwei Ausgabepins (und nicht wie vorher an einem Ausgabepin und der Erde) angeschlossen, kann die Spannung an der Zwei-Farben-LED positiv oder negativ eingestellt werden (Pin1 HIGH und Pin2 LOW oder Pin1 LOW und Pin2 HIGH).

Belast die LED einfach an ihrem Platz, und schließt nun den Draht, der vorher an die Erde angeschlossen war, an einen weiteren Ausgangspin an. Programmiert nun eine Steuerung, die die LED abwechselnd in rot und grün blinken lässt. Variiert dabei die Blinkgeschwindigkeit und versucht die LED immer schneller blinken zu lassen.

3 Der Taster

Wir wollen nun ein weiteres externes Element zu unserer Steuerung hinzufügen. Dieses Element ist ein Taster. Der Taster soll jetzt dafür zuständig sein, dass die LED bei Betätigung angeschaltet wird.

3.1 Verdrahtung des Tasters

Über den Taster wird der Eingabepin 8 des Mikrocontrollers mit Ground verbunden (Unterteilung des Tasters ist über den Strich auf der Rückseite gekennzeichnet). Bei Betätigung des Tasters wird somit der Eingabepin auf Ground gelegt. Damit der Eingabepin auf High liegt, wenn der Taster nicht betätigt ist, ist ein interner Pull-Up-Widerstand notwendig. Der folgende Code soll euch als Hilfestellung dienen.

```
1  /*
2  * Programm Taster
3  */
4
5  int tasterPin = 8;
6
7  void setup() {
8      // hier weitere Pins initialisieren...
9
10     pinMode(tasterPin, INPUT);
11
12     // Damit der Eingang sich immer in einem definierten
13     // Zustand befindet, wird der Pull-Up-Widerstand aktiviert
14     digitalWrite(tasterPin, HIGH);
15 }
16
17 /**
18 * Diese Funktion wird in einer Schleife durchlaufen
19 */
20 void loop() {
21     // hier kann mit digitalRead(8) der Pegel von Pin 8 abgefragt werden
22     // der Pegel kann HIGH oder LOW sein
23     if(...) {
24         // TODO
25     }
26     else {
27         // TODO
28     }
29 }
```

3.2 Aufgabe

Implementiert ein Programm, dass die angeschlossene LED leuchten lässt, wenn der Taster gedrückt wird. Versucht den Zustand der LED zu speichern und diesen bei Betätigung des Tasters zu wechseln, so dass ihr einen Ein- Ausschalter bekommt. Was sind die Probleme die dabei auftreten können?

4 Pulsweitenmodulation (PWM)

In dieser Aufgabe werden wir eine LED mittels PWM ansteuern. PWM steht für Pulsweitenmodulation und beschreibt eine Modulationsart, bei der ein Signal zwischen zwei Werten wechselt und die übertragene Information in der Breite der einzelnen Signale besteht. In diesem Beispiel soll durch diese Methode die Helligkeit der LED geändert werden können. Da der Mikrocontroller mit einer wesentlich höheren Frequenz arbeitet als das menschliche Auge erkennen kann, können wir dieses Prinzip verwenden um unsere LED sehr schnell blinken zu lassen. Für das Auge erscheint die Lichtquelle dann in unterschiedlichen Leuchtstärken. Das Verhältnis zwischen Ein- und Ausschaltdauer bestimmt dabei die Helligkeit.

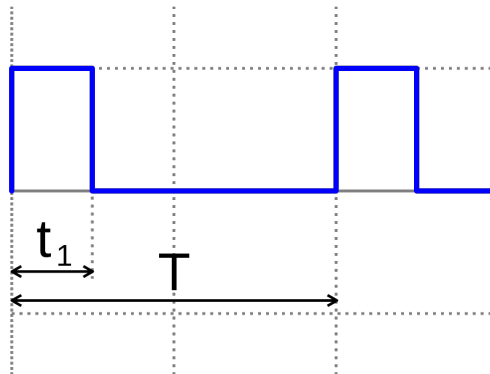


Abbildung 2: Beispiel einer PWM mit Tastverhältnis von 25%

4.1 Softwareimplementierung

In den vorherigen Aufgaben wurde bereits erläutert wie man eine LED an den Mikrocontroller anschließen und diese ein- bzw. aus schalten kann. Nun erweitern wir dieses Programm um eine weitere Funktion in der wir softwaremäßig ein PWM-Signal auf den Ausgang schreiben. Um eine Leuchtstärke von Beispielsweise 25% zu erreichen, wird der Ausgang einen Taktzyklus auf HIGH und 3 Taktzyklen auf LOW gesetzt. Dazu können wir den bekannten DigitalWrite Befehl innerhalb passender Schleifen verwenden. Um eine angemessene Geschwindigkeit zu erreichen, können wir zum Warten den Befehl `delayMicroseconds(micros)` verwenden.

Wir schließen die LED an den Mikrocontroller an und lassen sie in 25% Schritten bis auf volle Leuchtstärke steigen.

4.2 Hardwareimplementierung

In der vorherigen Aufgabe haben wir eine Pulsweitenmodulation (PWM) in Software realisiert. Da PWM häufig in der Praxis eingesetzt wird, ist diese Funktionalität gleich im Microcontroller als Hardwareeinheit zur Verfügung gestellt und erlaubt eine elegantere Implementierung.

Nun nutzen wir die Funktion `AnalogWrite` um einen analogen Wert von 0 (entspricht LOW) bis 255 (entspricht HIGH) mittels PWM auf den Ausgang zu schreiben. Diese Funktion verwendet intern die PWM Hardwareeinheit des Mikrocontrollers, die nur auf einzelnen Pins verfügbar ist. In Abbildung 1 sind diese Pins mit einem * markiert.

5 Kreativität gefragt!

Wir möchten euch nun ein paar Vorschläge für weitere Einsatzmöglichkeiten machen. Die folgende Liste bietet einige Anregungen für weitere Anwendungen.

- Boolesche Algebra 1 - Realisiert mit Hilfe von zwei Tastern eine AND oder eine OR Funktion. Die Funktionen können sowohl in Software, als auch in Hardware (durch entsprechendes Verdrahten der Taster) realisiert werden.
- Boolesche Algebra 2 - Realisiert mit Hilfe von zwei Tastern eine XOR Funktion. Wird ein Taster gedrückt, leuchtet die LED, werden beide, oder kein Schalter gedrückt, ist die LED aus.
- Der Dimmer - programmiert mit Hilfe eines Tasters eine dimmende LED. Wird der Taster gedrückt, soll die LED langsam hochdimmen. Wenn Sie den Maximalwert erreicht, fängt sie wieder bei 0 an.
- Der Dimmer EXTENDED - realisiert einen Dimmer, der sowohl hoch als auch runter dimmen kann. Dazu benötigt ihr zwei Taster.
- Wechselblinker - Lasst langsam Rot und Grün abwechselnd ein- und ausblenden.
- Die Ampel - Die Ampel regelt einen Fußgängerüberweg in einer Einbahnstraße. Drückt ein Fußgänger den Taster, soll die Ampelanlage 1 Sekunde warten und danach auf Rot schalten, damit der Fußgänger die Straße überqueren kann. Nach 6 Sekunden schaltet die Ampel dann wieder auf grün.
- Seid kreativ und überlegt euch eigene Anwendungen für den Mikrocontroller und die Bauteile!

6 Referenz

6.1 Begriffe und Bauteile

- **LED**
Die Light Emitting Diode ist eine lichtemittierende Halbleiterdiode. Eine einfach LED lässt den Strom nur in eine definierte Richtung durch. Die Dual-LED besteht intern aus 2 einfachen LEDs. LEDs dürfen hier grundsätzlich nur mit Vorwiderstand betrieben werden.
- **PWM**
Pulsweitenmodulation ist ein Verfahren um Signale über eine digitale Leitung zu übertragen. (Siehe Aufgabe 4)
- **Ground**
Ground (auch GND, Erde oder Masse) ist der negative Pol der Spannungsquelle und wird mit 0 Volt gleichgesetzt.
- **VCC**
VCC ist die Versorgungsspannung. Wir arbeiten mit einer Spannung von 5 Volt die wir über USB beziehen.
- **Pin**
Pin ist die Bezeichnung für einen Anschluss am Mikrocontroller. Die Nummerierung ist 1 auf Seite 3 zu entnehmen.
- **Quarz**
Der 16 MHz-Quarz schwingt 16 Millionen mal pro Sekunde und dient als Taktgeber für den Controller.
- **Taster**
Der Taster verbindet, wenn gedrückt, alle angeschlossenen Kontakte miteinander. Er funktioniert dann quasi wie eine Drahtverbindung zwischen den angeschlossenen Pins. Wird er so eingebaut wie auf 1 auf Seite 3, verbindet er die obere mit der unteren Kontaktreihe.
- **Widerstand**
Der Widerstand begrenzt den Stromfluss. Hier wird er zusammen mit Dioden eingesetzt, da diese nur einen begrenzten Stromfluss aushalten.

6.2 C-Syntax

- **Variablendeklaration**
`TYP name = Wert;`
- **IF-Bedingung**
`if(Bedingung){ } else { }`
- **While-Schleife**
`while(Bedingung){ }`
- **For-Schleife**
`for(Definition, Bedingung, Anweisung){ }`

6.3 Befehle

- **void setup()**
Diese Funktion wird als Initialisierung einmal zu Beginn des Programms durchlaufen. Hier können die Ein- und Ausgänge des Controllers definiert werden
- **void loop()**
Diese Funktion wird zur Laufzeit des Programmes ständig in einer Endlosschleife aufgerufen. Hier sollte die eigentliche Logik der Anwendung implementiert werden.
- **void pinMode(pin, mode)**
Konfiguriert das Verhalten eines Pins, mode kann entweder INPUT oder OUTPUT sein.
- **void digitalWrite(pin, value)**
Legt den value HIGH oder LOW auf einen pin.

-
- `int digitalRead(pin)`
Liest den Zustand eines Pins aus und gibt diesen zurück.
 - `void analogWrite(pin, value)`
Schreibt einen analogen value zwischen 0 und 255 als PWM-Signal auf einen Ausgang.
 - `void delay(millis)`
Wartet den übergebenen Wert in Millisekunden.
 - `void delayMicroseconds(micros)`
Wartet den übergebenen Wert in Mikrosekunden.

Eine vollständigere Übersicht ist unter der Hilfefunktion zu erreichen.

6.4 Links

<http://www.arduino.cc/> (Arduino Webseite)
<http://www.arduino.cc/en/Reference/HomePage> (Arduino Hilfe)
<http://de.wikipedia.org/wiki/Arduino-Plattform>
<http://www.heise.de/ct/projekte/machmit/processing>
<http://www.obdev.at/products/vusb/usbasploader.html> (USB Bootloader)
<http://www.obdev.at/products/vusb> (Verwendete USB Bibliothek und Schaltpläne)
<http://www.fs-ist.de> (Ein bisschen Eigenwerbung muss sein)

6.5 Dank an

Wir danken dem Fachgebiet Integrierte Elektronische Systeme und besonders Roland Brand für die Ausleihe der Steckbretter und Kabelbrücken, der RBG für die Installation der Compilerumgebung und die notwendige Rechtevergabe auf den Poolrechnern, sowie allen weiteren Personen die diesen Workshop ermöglicht haben.

6.6 Kontakt

Für Fragen oder Verbesserungsvorschlägen könnt ihr uns unter **info@fs-ist.de** erreichen! Wir würden uns über Feedback sehr freuen.