DIY: Herstellung eines personalisierten Weckers mit Mikrocontroller und mp3-Player

Malte Ollenschläger

***Zusammenfassung*—Dieses Dokument beschreibt die Herstellung eines personalisierten Weckers. Das beinhaltet die Erstellung eines Gehäuses und das Schreiben der Software für den Mikrocontroller. Auch die Bauteilauswahl und die Fertigung von Platinen sind in diesem Projekt inbegriffen. Das Ziel ist einerseits die Fertigung des Weckers, andererseits die Durchführung eines Projektes mit verschiedenen Aufgabenbereichen, um das hierbei Gelernte, in folgenden Projekten zu einer effizienteren Arbeitsweise umzusetzen. Dies betrifft sowohl inhaltliche als auch formelle Arbeitsschritte.**

I. Einleitung

DIE Gesamtfunktion des Weckers setzt sich aus mehreren Teilfunktionen zusammen. Eine der grundlegenden Funktionen ist die *Anzeige der Uhrzeit* auf einem Display. Weiterhin wird das *Datum* durch die Zusammensetzung des aktuelle Monats als Wort und des Tages als Ordinalzahl angezeigt. Ein weiteres Symbol gibt an, ob der *Alarm ein- oder ausgeschaltet* ist. Das *Umschalten* funktioniert durch das Gedrückthalten eines Tasters, über den gleichzeitig bei kurzem Drücken die *Displaybeleuchtung* (UND DER FRONTPLATTE) eingeschaltet werden kann. Die Implementierung der *Weckfunktion* sorgt bei eingeschaltetem Alarm dafür, dass zur konfigurierten Uhrzeit der mp3-Player gestartet und eine sich auf selbigem befindliche mp3-Datei abgespielt wird. Beim nächsten Weckruf wird die darauf folgende Datei abgespielt.

Über ein *Menü*, das über das Bedienen eines zweiten Tasters zu erreichen ist, sind folgende Elemente einstellbar:

- Uhrzeit
- Weckzeit
- Datum
- Lautstärke
- Boombox

Innerhalb des Menüs findet die Navigation über einen Drehencoder statt. Bei der Einstellung des Datums wird neben Tag und Monat auch das Jahr abgefragt. Dies dient der *Erkennung von Schaltjahren*, damit in diesen in der Datumsanzeige auf den 28. Februar statt dem 1. März der 29. Februar folgt. Mit dem Menüpunkt "Lautstärke" lässt sich eben diese des mp3-Players und damit die *Lautstärke des Wecktons* verändern. Wird der Menüpunkt "Boombox" ausgewählt, so kann über ein Cinch-Kabel eine *externe Audio-Quelle* angeschlossen werden und die darauf abgespielte Musik wird über die Lautsprecher des Weckers wiedergegeben. In dieser Zeit ist die Weckfunktion außer Kraft gesetzt.

Die Personalisierung des Weckers wird durch drei Eigenschaften erreicht. Die erste ist die Tatsache, dass die Weckmusik nach dem eigenen Geschmack angepasst werden kann, indem diese auf den mp3-Player kopiert wird. Die zweite personalisierte Eigenschaft ist, dass beispielsweise an Valentinstag und am Geburtstag der Person statt des Datums andere Bilder oder Texte eingeblendet werden. Der offensichtlichste Teil der Personalisierung liegt in der Herstellung des Gehäuses. In die Front des Weckers werden Motive zu denen die Benutzerin einen persönlichen Bezug hat geschnitten. Diese werden zusätzlich zum Display beleuchtet, wenn der entsprechende Taster betätigt wird. Als Vorlage für dieses Projekt diente ein ebenfalls selbst hergestellter Wecker, der auf der Internetseite www.wiesolator.de vorgestellt wurde.

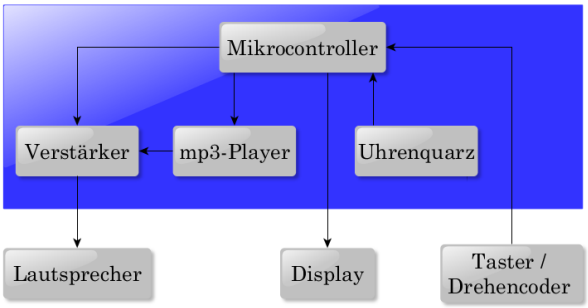

Abbildung 1: Hierarchie der Elementaren Bauteile des Weckers. Die Pfeilspitze zeigt in die Datenflussrichtung. Blau unterlegt sind die für den Benutzer nicht zugänglichen Bauteile im Gehäuse. Die unterste Zeile zeigt die Ein- und Ausgabeebene, worüber der Benutzer mit dem Wecker interagiert.

II. Hardware

In diesem Abschnitt wird die genutzte Hardware erläutert. Hierzu wird in Teil II-A beschrieben, welche Bauteile benötigt werden. Der sich anschließende Teil II-C befasst sich mit der externen Beschaltung der Bauteile. In II-D wird auf die Planung und Fertigung der Platinen eingegangen.

A. Bauteile

Der Schaltplan, der im Vorlageprojekt genutzt wurde, dient als Anhaltspunkt für die Hardware, die benötigt wird. Daraus wurde das Schaubild in Abb. 1 erstellt, um eine Übersicht der wichtigsten Bauteile zu erhalten. Eine Auflistung aller benötigten Bauteile ist in Tabelle I aufgeführt. Die benötigten Widerstände und Kondensatoren sind nicht aufgeführt, da diese den Datenblättern der jeweiligen Bauteile zu entnehmen sind. Zusätzlich werden für das Entprellen der Taster und des Drehencoders 100nF Kondensatoren eingesetzt.

Funktionseinheit	Herstellerbezeichnung
Buchsenleiste	EBLG-20-1-X
Display	EA DOGL128W-6
Displaybeleuchtung	EA LED68x51-W
Drehencoder	STEC11B
Lautsprecher	Visaton FR8-8
Mikrocontroller	Atmega328P
mp3-Player	Intenso Music Walker
Schaltnetzteil 5V/1A	goobay 54805
Spannungsregler 1,5VDC	LM317
Spannungsregler 3,3VDC	LM3940
Spanplatten	
Schrauben	
Muttern	
Taster	FEHLT
Uhrenquarz 32kHz	-
Verstärker	TDA7266

Tabelle I: Benötigte Bauteile. Notwendige Kondensatoren und Widerstände sind nicht angegeben. Diese sind in den jeweiligen Datenblättern zu finden.

Neben dieser Hardware werden Spanplatten zur Fertigung des Gehäuses, sowie einige Schrauben benötigt.

B. Gehäuse

C. Dimensionierung

Grundsätzlich wurden die in den Datenblättern empfohlenen Kondensatoren und Widerstände eingesetzt. Zum Entprellen der Taster sind in den entsprechenden Datenblättern keine Informationen vorhanden. Es werden Keramikkondensatoren der Größe $C = 100nF$ genutzt. Bei dem Spannungsregler für den mp3-Player (LM317) muss eine individuelle Dimensionierung vorgenommen werden, da der Ausgang des gewählten Spannungsreglers variable Spannungen erzeugen kann. Die benötigte Versorgungsspannung beträgt $1,5V$. Nach der im zugehörigen Datenblatt [QUELLE] angegebenen Formel, kann für eine gewünschte Ausgangsspannung die Größe der benötigten Widerstände berechnet werden. Somit ergibt sich die Größe von R_2 nach folgender Gleichung:

$$R_2 = (\frac{V_{out}}{V_{ref}} - 1)/(\frac{1}{R_1} + I_{adj}) \quad (1)$$

Für den Ausgleichsstrom wurden $50\mu A$ und damit die Hälfte des Maximums angenommen. Dieser wird vom Spannungsregler so geregelt, dass $V_{ref} = 1,25V$ gilt. Bei einer Wahl von $R_1 = 10k\Omega$ ergibt sich für $R_2 \approx 1,3k\Omega$.

D. Platinen

Die für dieses Projekt benötigten Platinen wurden mit KiCAD geplant und anschließend im FabLab der Friedrich-Alexander-Universität Erlangen-Nürnberg erstellt. Eine Platine ist für das Display und die zugehörigen Bauelemente (Kondensatoren, Widerstände) ausgelegt. Über einen Steckverbinder kann diese Platine mit einer zweiten verbunden werden, welche die anderen Bauelemente wie zum Beispiel den Mikrocontroller und den Verstärker enthält.

Für die Displayplatine wurden die Anschlüsse des Displays nicht direkt auf der Platine verlötet. Es wurden Buchsenleisten verlötet in welche das Display samt verlöteter Hintergrundbeleuchtung eingesteckt wird. Dies dient einerseits dazu, dass das Display zum Wechsel oder zur Reparatur ausgebaut werden kann; andererseits dazu, dass der Abstand zwischen Gehäuseinnenseite und Platine vergrößert wird. Dies ist notwendig, damit der Drehencoder, welcher auf der Platine befestigt wird, nicht zu weit nach vorne aus dem Gehäuse herausragt. In die vier Ecken der Platine werden Löcher gebohrt, ebenso im gleichen Abstandsmaß in die Front des Weckers. Anschließend kann die Platine per Schraube und Mutter von innen gegen das Gehäuse gedrückt werden, sodass eine kraftschlüssige Befestigung entsteht und das Display durch die rechteckige Öffnung in der Front des Gehäuses betrachtet werden kann. Für die Erstellung der Platine wurde eine Leiterbahnbreite von $0,5mm$ gewählt, da genug Fläche vorhanden war.

Für die Hauptplatine wurde ebenfalls eine Leiterbahnbreite von $0,5mm$ gewählt. Neben der Displayplatine wurden weitere Bauelemente per Steckverbinder mit der Platine verbunden. Diese sind:

- Lautsprecher
- Taster
- Drehencoder
- LED-Beleuchtung (Bilder)
- mp3-Player

E. Pinbelegung Mikrocontroller

Da es für den Oszillator keine Auswahlmöglichkeiten bezüglich der Pinbelegung gibt, wurden ihm als erstes die entsprechenden Pins zugewiesen. Die Pins für das Displays wurden so gewählt, dass alle nebeneinander liegen, damit das Layout der Platine erleichtert wird und lediglich gerade Leiterbahnen zwischen den Pins des Mikrocontrollers und dem betreffenden Steckverbinder erstellt werden müssen. Gleiches gilt für die Belegung der Pins zur Ansteuerung des mp3-Players.

Die Auswahl für die Belegung der Taster unterliegt lediglich der Einschränkung, dass die Taster (Licht und Drehencoder) einem anderen Interrupt zugeordnet werden als eine Drehung des Drehencoders. Dies führt zu einer vereinfachten Auswertung der Drehrichtung, siehe III-B.

III. Software

Um die gewünschten Inhalte auf dem LCD-Display anzuzeigen, müssen diese Elemente zunächst generiert werden. Dies geschah mittels eines Java-Programms, welches in Abschnitt **??** beschrieben wird. Der zweite Abschnitt befasst sich mit dem umfangreicheren Code, der auf dem Mikrocontroller eingesetzt wird.

A. Bildkonverter

Zunächst werden Dateien mit einzelnen Buchstaben, Symbolen und Bildern in Inkscape erstellt. In Schriftgröße 18 gibt ein zentrierter Großbuchstabe *J* die vertikale Position aller weiteren Buchstaben vor, da dies der höchste Buchstabe ist.

Die erstellte Grafik wird als **.png* exportiert und dem Java-Programm zugeführt. Dieses trägt den Namen *Bildkonverter*. Wird es geöffnet, sind drei Buttons und ein Eingabefeld zu sehen. Über *Datei wählen* wird das zu verarbeitende Bild ausgewählt. Die Umwandlung startet durch Klicken des entsprechenden Buttons. Zunächst wird durch Auswertung des Rotkanals jedes Pixels ein Binärwert erzeugt. Ist dieser größer oder gleich der angegebene Schwelle, so wird das entsprechende Pixel gesetzt. Ist es geringer als der Schwellwert, wird dieser Pixel auf null gesetzt. Anschließend werden jeweils 8 aufeinander folgende Pixel einer Spalte zu einem Hexadezimalwert zusammengefasst, um das Handling zu erleichtern. Wenn ein Bild eine Höhe von 16px hat und eine Breite von 3px, so werden 3 Mal zwei Hexadezimalwerte erzeugt. Diese sechs Werte werden in geschweiften Klammern ausgegeben, damit diese in einem Alphabet-Array im Code für den Mikrocontroller gespeichert werden können. Weiterhin wird eine Vorschau des Binärbildes angezeigt. So müssen die Daten nicht erst auf den Mikrocontroller überspielt und am LCD-Display angezeigt werden, um Fehler zu erkennen. Diese können beispielsweise durch Änderung des Schwellwertes oder durch das Setzen oder Löschen einzelner Pixel in dem Bild mittels Bildbearbeitungssoftware (zum Beispiel Paint) korrigiert werden. Zum Beenden des Konverters dient der Button *Schließen*. Die Ausgabe befindet sich nun in einer Datei mit dem Namen *Konverter-output.txt* in dem Ordner, in dem das Java-Programm gestartet wurde.

B. Atmel-Code

Der Code für den Mikrocontroller setzt sich aus verschiedenen Dateien zusammen. Die Hauptaufgabe übernimmt eine Endlosschleife in der Main-Funktion der Datei *EADOGL128.c*. In *EADOGL128_commands.c* wurden die Befehle zur Ansteuerung des LCD-Displays implementiert. Um den mp3-Player und den Audio-Verstärker anzusprechen, werden die in *mp3.c* erstellten Funktionen genutzt. Nicht selbst geschriebener Code wurde in Form von Standardbibliotheken wie *avr/io.h* genutzt sowie in der Datei *sbit.h* [QUELLE], welche es ermöglicht, Pins wie Variablen ansprechen zu können.

EADOGL128.c: Es gibt mehrere Interrupt-Service-Routinen, die Einfluss auf die ausgeführten Aktionen in der Main-Funktion nehmen. Im Normalzustand befindet sich der Mikrocontroller im sleep-Modus. Um in diesen einzutreten sind einige Vorbereitungen notwendig, die in der Funktion *goodNight* implementiert sind. Das Verlassen des sleep-Modus geschieht durch eines von mehreren möglichen Interrupts. Hierzu zählen die Bedienung des Drehencoders und des Tasters sowie ein Timer-Overflow aufgrund der Zeitmessung mittels Uhrenquarz. Dieser Overflow findet ein Mal pro Sekunde statt. Wenn sich auf dem LCD-Display durch das Zeit-Update nichts ändern muss (Sekunde<60), wird der sleep-Modus erneut aktiviert.

Im Falle einer Benutzereingabe oder eines notwendigen Updates des LCD-Displays, wird die Funktion *goodNight* verlassen und das betreffende Ereignis wird abgearbeitet. Danach tritt der Mikrocontroller erneut in den sleep-Modus ein.

Interrupt-Service Routinen:
Die ISR PCINT0_vect setzt bei gedrückten Tasten das entsprechende Flag auf *high*. Somit wird zum Beispiel das Einschalten des Lichts in der main-Funktion eingeleitet. Gäbe es nur einen globalen Interrupt für alle Benutzereingaben, so müsste der letzte bekannte Status des Drehencoders gespeichert werden, damit überprüft werden kann, ob eine Änderung stattgefunden hat. Durch die Zuweisung der Signale, die bei einer Drehung verändert werden an eine andere ISR (PCINT2_vect), als jener die bei Tastendruck aufgerufen wird (PCINT0_vect), kann dieser Schritt eingespart werden. Die ISR (PCINT2_vect) vergleicht den Status der beiden betreffenden Signale. Die Drehrichtung lässt sich daraus ermitteln, ob beide Signale gleich (beide *high* oder beide *low*) oder unterschiedlich sind. Je nach Ergebnis dieses Vergleichs wird bei einer detektierten Rechtsdrehung eine Zählvariable inkrementiert sowie bei Linksdrehung dekrementiert. Diese muss bei Abruf auf *null* zurückgesetzt.

Bei eingeschalteter Beleuchtung bewirken alle Benutzereingaben eine Verlängerung der Leuchtdauer so, als ob der Taster für die Beleuchtung zu diesem Zeitpunkt betätigt wurde.

Die dritte ISR wird aufgerufen, wenn ein Overflow in dem vom Uhrenquarz getakteten Counter stattfindet. Da ein 8-bit Timer mit einem Prescaler von 128 genutzt wird, findet dieser Overflow ein Mal pro Sekunde statt. Hierbei wird die Variable *sekunde* um eins inkrementiert. Wenn diese Variable den Wert 59 übersteigt, wird ein Flag gesetzt, welches das Update der Minute und gegebenenfalls der Stunde sowie des Datums in der main-Funktion einleitet. Im gleichen Zug wird damit der Inhalt des LCD-Displays entsprechend verändert.

Eine vierte ISR wird genutzt um die Dauer eines Tastendrucks zu ermitteln. Bei langem Drücken des Drehencoders wird so das Menü aufgerufen. So ist es mögich, dass der Taster für das Licht bei kurzem Drücken zum Einschalten der Beleuchtung und bei langem Drücken zum Ein-/Ausschalten des Alarms führt.

uC ohne sleep 6,53mA mit 1,03mA mit Spannungsregler3,3 und sleep 9,3 mA Folglich hat Spannungsregler 9,3-1,03= 8,2mA zusätzlich 1,5 Spannungsregler 9,51 mA –> macht 0,2mA das alles mit Display 9,83 –> Display 0,32mA

IV. Quellen

http://www.avrfreaks.net/sites/default/files/SBIT.H