Projekt "Arduino"

Erstellung eines Thermometers

Stefan Wolf
Angewandte Informatik - Mobile Systeme 2011
stefan.wolf@student.inf.hs-anhalt.de
Matrikel-Nr: 4054391

letzte Bearbeitung: 2. Juli 2013

Inhaltsverzeichnis

1. Einleitung und Motivation

1.1 Entstehung der Projektidee

Als ich zum ersten Mal in die Arduinoveranstaltung meiner Hochschule gekommen bin, hatte ich keine Ahnung, worum es sich dabei überhaupt genau handelt. Daher war es klar, dass ich mich erstmal mit den Basics vertraut machen muss. Auf Grund dessen habe ich mir zuerst ein paar Bücher besorgt und in diesen gestöbert.
Allerdings ging die ganze Problematik nach kurzer Zeit schon relativ leicht von der Hand. Deshalb musste jetzt langsam eine Idee her. Für mich stand fest, dass das ganze nichts mit LED-Würfeln oder Ähnliches zutun haben sollte.
Nach langem Hin und Her hab ich mir gedacht, dass ich mir ja ein Display besorgen könnte und auf diesem irgendetwas anzeige. Das war eigentlich die ganze Entstehung der Idee. Da mir nichts besseres eingefallen ist, wollte ich eine Temperatur auf dem Display anzeigen lassen.

1.2 Vorstellung der Idee

Bei dem Projekt geht es darum eine Temperatur auf dem Display anzeigen zu lassen. Zwischenzeitlich hatte sich die Idee eins, zwei mal verändert. So sollte mal zusätzlich zur aktuellen Temperatur noch der minimale und der maximale gemessene Wert angezeigt werden. Die Idee wurde allerdings nach den ersten Entwürfen der Anzeige wieder verworfen. Auch der Plan die Temperatur ebenfalls in Kelvin und Grad Fahrenheit anzeigen lassen zu können, wurde wieder abgewendet.

Meine Hauptintention lag in der Ideesammelphase darin, das genaue Zusammenwirken der einzelnen Bauteile und deren Ansteuerung kennenzulernen. Auch das Arbeiten mit den Datenblättern musste trainiert werden.
Zu diesem Zeitpunkt dachte ich noch, dass es sich bei diesem Projekt um eine leicht zu lösende Aufgabe handelt. Diese hat sich aber schnell schon als schwieriger herausgestellt als vorher gedacht. Aber dazu an späterer Stelle mehr.

2. Konzept

In der Konzeptionsphase habe ich mich damit befasst, alle benötigten Arbeitsmittel für das Projekt zu besorgen. Im folgenden werde ich meine Teile auflisten und deren Funktion erläutern. Bilder zu den einzelnen Teilen befinden sich im Anhang.

- **Arduino-StarterKit**
  Das Arduino Starterkit haben wir im Rahmen der Veranstaltung bekommen. Aus diesem Starterkit benötigte ich folgende Teile:
  - **Kabel**
    Viele Kabel aus diesem Kit wurden für das Projekt verwendet. Diese braucht man um die einzelnen Bauteile miteinander zu verbinden.
  - **Potentiometer**
    Das Potentiometer aus diesem Kit wurde zur Regelung des Kontrastes genommen.
  - **Temperatursensor LM35**
    Um die Temperatur der Umgebung zu messen und auf dem Display auszugeben, wird dieser Temperatursensor genommen.
  - **Steckplatine**
    Da ich alle Teile aus dem Kit wieder abgeben muss, konnte ich an diese keine Kabel anlöten. Zum Verbinden der einzelnen Teile wurde daher die Steckplatine verwendet.
- **LCD-Display EA DIP204-4**
  Dabei handelt sich um meinen verwendeten Display. Während des Projektes wollte ich die Funktionsweise dieses Teils lernen. Auf Grund dessen sind auch Teile programmiert, die nicht wirklich gebraucht werden.
  Beim Endprodukt ist dieses Gerät dafür zuständig, die aktuelle Temperatur anzuzeigen.
  Das Display besitzt 18 Anschlüße. Es hat 2 Seiten mit je 9-Stück mit einem Rastermaß von 2,0mm. Da die Steckplatine des Arduinos im Rastermaß von 2,54mm bestückt ist, ist es nicht möglich, dieses Display direkt anzuschließen.

- **2x Buchsenleisten**
  Bei diesen Buchsenleisten handelt es sich um 2 10-polige, welche ein Rastermaß von 2mm besitzen. Diese wurden an das Display gesteckt und daran Kabel angelötet. Dieses Teil wäre nicht von Nöten gewesen. Allerdings wollte ich, da ich noch nie vorher etwas gelötet hatte, das Display nicht kaputt machen.

- **Festspannungsregler**
  Dieses Teil regelt eine Spannung auf 3.5V. Der Festspannungsregler wurde dafür verwendet, die 5V des Arduinos für die Hintergrundbeleuchtung des Displays auf 3.5V zu regeln.

3. Umsetzung

Die Umsetzungsphase werde ich im Folgenden in die einzelnen Problem- bzw. Herausforderungsphasen unterteilen. Der Schwierigkeitsgrad der einzelnen Punkte ist dabei allerdings nicht gleichmäßig verteilt. Das bedeutet, dass einige Unterpunkte leichter zu erledigen waren als Andere.
Im Anhang befinden sich wieder Bilder aus den einzelnen Phasen. Diese sind zusätzlich entsprechend nummeriert.

3.1 Löten

Der erste Schritt den es zu erledigen galt, war das Anlöten von Kabeln an die Buchsenleisten. Da ich mich mit dieser Aufgabe noch nie beschäftigt hatte, habe ich einen Kumpel um Hilfe gebeten. Dieser hat mir bei der Realisierung dieses Aufgabenteils geholfen.
In der Lötphase wurde jeweils ein Kabelende eines Kabels mit einem Pin der Buchsenleiste verbunden. Das Andere wurde verzinnt, damit dieses sich leichter mit der Steckplatine verbinden lässt. Damit war das Problem abgeschlossen und ich konnte mich um den nächsten Punkt kümmern.

3.2 Display Anschließen

Beim Anschließen der Pins war es erstmal notwendig, sich über die einzelnen Funktionen schlau zu machen. Dafür musste ein Blick in das Datenblatt des Displays geworfen werden. Im Folgenden werde ich die einzelnen Pins auflisten und erläutern.

- **Pin 1**
  Dabei handelt es sich um den Ground bzw. Masse-Pin.
- **Pin 2**
  Dies ist der Pluspol des Displays. Dieser benötigt eine 5V Stromversorgung.

- **Pin 3**
  Bei diesem Pin handelt es sich um die Kontrastspannung des Displays. Diese habe ich an einen Potentiometer angeschlossen, um damit den Kontrast auf einfache Art und Weise regeln zu können. Allerdings habe ich bemerkt, dass das Regeln über ein Potentiometer wenig Sinn macht, da der Toleranzspannungbereich ziemlich klein ist und sich daher wenig einstellen lässt.

- **Pin 4**
  Mithilfe dieses Pins lässt sich dem Display sagen, ob Befehle oder Daten zum Display geschickt werden. Wenn Befehle auf den Display geschickt werden, ist dieser Pin LOW, bei Daten HIGH.

- **Pin 5**
  Der R/W-Pin teilt dem Display mit, ob im nächsten Schritt auf das Display geschrieben oder vom Display gelesen wird. Dafür muss der Pin mit LOW belegt sein, wenn geschrieben werden soll und mit HIGH beim Lesen.

- **Pin 6**
  Bei diesem Pin handelt es sich um den Enable-Pin. Die Wirkungsweise dieses Pins herauszufinden, hat mir die meiste Zeit des Projektes gekostet. Das lag einfach daran, dass der Enable-Pin nur ganz kurz im Datenblatt erwähnt wurde und sozusagen vom Leser erwartet wurde, dass er das dafür notwendige Know-how besitzt. Dieser Pin muss nachdem alle Pins richtig gesetzt wurden einen Flankenwechsel von High auf Low haben. Nachdem dies passiert ist, führt das Display den gesendeten Befehl aus.
  Allerdings muss jetzt noch darauf geachtet werden, dass man, bevor das nächste zum Display geschickt wird, wartet, bis der Befehl fertig ausgeführt wurde.
  Dafür gibt es zwei Möglichkeiten. Zum einen kann man die im Datenblatt stehenden Zeiten abwarten. Dies hat den Vorteil, dass man den Prozess die nötige Zeit schlafen legen könnte und die CPU diesen vorerst nicht ausführen muss. Aber es gibt den Nachteil, dass man unter Umständen zu lange wartet, da im Datenblatt die Maximaldauer der einzelnen Befehle steht. Zum anderen könnte man ein Busy-Flag durch Polling abfragen. Der Vorteil hierbei besteht darin, dass man immer nur so lange wartet, wie nötig ist und keine unnütze Zeit verstreichen lässt. Allerdings läuft hierbei der Prozess immer auf 100% und die CPU kann sich ggf. in der Zeit nicht voll und ganz um andere wichtige Prozesse kümmern.
  Bei meinem Projekt hab ich mich für die erste Variante entschieden, da ich keine zeitkritische Anwendung damit bedienen möchte.

- **Pin 7 - Pin 14**
  Dabei handelt es sich um die 8 Datenpins von DB0 - DB7. Mit deren Hilfe, werden die Daten und Befehle spezifiziert.

- **Pin 15**
  Dieser Pin hat keine Funktion.

- **Pin 16**
  Dieser Pin ist der Reset-Pin. Mithilfe eines Flankenwechsels von High auf Low kann man damit das Display komplett zurücksetzen.
- **Pin 17**
  Hierbei handelt es sich um den Pluspol der LED-Hintergrundbeleuchtung. Dieser ist mit einem Festspannungsregler verbunden der die Spannung auf 3,5V herunter regelt.
- **Pin 18**
  Dieser Pin ist der Minuspol der LED-Hintergrundbeleuchtung.

Mithilfe dieser Informationen konnte man dann die Pins mit dem Arduino verbinden. Einen ausführlichen Schaltplan habe ich im Anhang hinterlegt. Als das Display fertig angeschlossen war, habe ich mit der Ansteuerung begonnen.

3.3 Display Programmieren

Nachdem das Anschließen des Displays abgeschlossen war, habe ich damit angefangen mir zu überlegen, wie ich das Display jetzt testen kann. Da ich keine Ahnung hatte, wie das Ganze genau funktioniert, habe ich damit begonnen eine kleine Displayklasse zu schreiben. Diese sollte anfangs nur ganz spartanische Methoden besitzen. So ist folgende kleine Headerdatei entstanden:

```
#define RW 12
#define RS 13

#define DB0 10
#define DB1 9
...
#define DB7 3

class Display
{
        private:
                /*
                 * Dieses Attribut beinhaltet die 8 Datenbits.
                 */
                int DB_ARRAY[8];
                /*
                 * Diese Methode bestimmt ob das Byte aus dem
                 * value-Parameter 1 oder 0 ist und gibt dies
                 * entsprechend zurueck.
                 */
                int getBit(int value, int bitNumber);
```

```
public:
        /*
         * Konstruktor hat DB_ARRAY initialisiert
         */
        Display();
        /*
         * Sollte Datenbyte an Display senden
         */
        void writeData(int value);
        /*
         * Sollte Befehl an Display senden
         */
        void executeCommand(int value);
}
```

Wie man sieht hatte der Enable-Pin zu diesem Zeitpunkt noch gar keinen Platz im Programm gefunden, da ich damit nicht umzugehen wusste. (Sieht man daran, dass die passende Präprozessorvariable fehlt!)
Nachdem diese Version nicht funktioniert hatte, hat es der Pin aber trotzdem ins Programm geschafft und nach ewigem Ausprobieren und Abändern das Quelltextes, hat es dann auch irgendwann funktioniert.

Nachdem ich es hinbekommen hatte erfolgreich mit dem Display zu kommunizieren, musste ich es schaffen, float-Werte an den Display zu senden. Daher mussten dafür auch noch Methoden geschrieben werden. Dieser Schritt war allerdings im Vergleich zum Vorherigen relativ leicht und schnell erledigt.
Als Letztes musste ich mir jetzt noch überlegen, wie ich die Werte auf dem Display darstelle. Ich habe mich daraufhin ziemlich schnell dafür entschieden, die Temperatur über alle 4 Zeilen anzeigen zu lassen und mich mit Photoshop um ein paar Entwürfe gekümmert. Die Schwierigkeit dabei bestand darin, dass ich für diese Aufgabe nur 8 eigene Zeichen definieren durfte. Nachdem ich 2 verschiedene Entwürfe erstellt hatte, habe ich einen Kumpel nach seiner Meinung über diese gefragt. Nach kurzer Zeit hatte er mir ein Bild mit einer Schriftart gesendet, die er im Internet gefunden hatte. Diese ist dann schließlich auch die finale Schriftart geworden.
Als auch dieser Schritt erfolgreich abgearbeitet war, kam ich zum vorerst letzten Abschnitt meines Projektes. Ich musste jetzt schließlich noch den Temperatursensor anbauen und programmieren.

3.4 Temperatursensor Anschließen und Programmieren

Da ich meine Grundkentnisse, während ich die anderen Abschnitte abgearbeitet hatte, ausbaute, war dieser Schritt mit Abstand am einfachsten zu realisieren. Den Sensor musste man einfach nur über 3 Pins anschließen. Zwei Anschlüsse davon sind der Plus-

und der Minuspol. Der dritte Pin ist die Ausgangsspannung aus dem Gerät, die abhängig von der Temperatur ist. Dieser wird einfach nur an einen der analogen Eingang des Arduinos angeschlossen.
Um die aktuelle Temperatur auszulesen, muss man jetzt einfach nur den Pin auslesen und ein bisschen rechnen. Im folgenden zeige ich das Ganze im Quellcode:

```
...
/*
 * adc - Analog/Digital-Converter
 * gibt Wert von 0 bis 1023 (2^10) zurueck!
 */
int adcValue = analogRead(temperatureSensorPin);
/* milliVolt - Voltzahl in mV, die am Sensor anliegt */
float millivolts = (adcValue / 1024.0) * milliVolt;

/* Sensor-Output ist 10mV pro Grad Celsius */
float celsius = millivolts / 10;
...
```

Allerdings musste ich feststellen, dass die Temperatur manchmal ziemlich große Ausreißer hat. Auf Grund dessen habe ich die Temperatur mittels Mittelwertbestimmung erhalten. Ich habe immer die letzten 14 gemessenen Werte gespeichert und den Mittelwert aus diesen Zahlen und der neu gemessenen Zahl bestimmt, um diesem Effekt entgegen zu wirken.

3.5 Coderefactoring

Im letzten Schritt musste der Code noch hübsch gemacht werden. Da man jetzt die komplette Funktionsweise kennt, konnte man sich nun darüber Gedanken machen, wie man den Quellcode am besten aufbaut.
Ich habe mich dafür entschieden, das ganze mit 3 Klassen und 2 Strukturen zu realisieren. Den kompletten Quellcode findet man ebenfalls komplett kommentiert im Anhang.

4. Ergebnisse bzw. Fazit

Während des Projektes hab ich eine Menge positive wie negative Erfahrungen dazu gewonnen und ich denke, dass sich das Ergebnis nicht zu verstecken braucht. Ich habe gelernt, wie ich mit elektronischen Bauteilen umzugehen habe, wie man mit diesen kommunizieren kann und wie man ein solches Projekt aufbaut. Leider musste ich feststellen, dass die Informationen die in den einzelnen Datenblättern stehen nicht immer die Wahrheit erzählen oder gar nützlich sind. Beispielsweise war die Definition von eigenen Zeichen für das Display vollkommen falsch beschrieben.
Außerdem konnte ich feststellen, dass mir das Programmieren mehr Spaß bereitet, als das Löten und Zusammenstecken. Nichtsdestotrotz hat es einen riesen Spaß gemacht und ich würde kommenden Arduino-Projekten nicht von vornherein einen Stein in den Weg legen.
Das bedeutet, dass ich auch in Zukunft dafür offen wäre, eine Problemstellung dieses Typs zu realisieren. Interessant fände ich es, eine kleine Art Betriebssystem mit einfachem Dispatcher und einfachem Scheduler für das Arduino zu bauen. Sodass man unter Umständen mehr als ein Projekt mit dem Arduino verwalten könnte. Vielleicht gibt es ja irgendwann die dafür notwendige Zeit.