Jonas Heilig Microcontroller/Microelektronik 25.04.2023

Einführung in die Mikroelektronik

Das Steckbrett

Wie ist das Steckbrett aufgebaut?

Das Steckbrett innen mehrere leiterbahnen, die alle eine genaue Bezeichnung haben, die man mit Hilfe das aufgezeichnete System ablesen kann. Außerdem haben die meisten Streckbretter an beiden Seiten jeweils zwei leiterbahnen, die man für die Stromzufuhr verwendet. Wie die Leiterbahnen aufgebaut sind siehe Bild PowerPoint.

Die Nachteile / Vorteile des Steckbrettes

Vorteile:

* Steckbretter sind sehr einfach zu bedienen und erfordern keine speziellen Werkzeuge oder Lötkenntnisse.
* Sie sind sehr flexibel und können schnell und einfach umkonfiguriert werden, um neue Schaltungen aufzubauen oder bestehende Schaltungen zu ändern.
* Sie können problemlos viele verschiedene Bauteile und Komponenten aufnehmen, von einfachen Widerständen bis hin zu komplexen ICs (Microchips, Bsp.: Arduino).
* Sie eignen sich gut für den Austausch von Bauteilen und die Fehlerbehebung, da sie eine schnelle und einfache Methode zum Einsetzen oder Entfernen von Bauteilen bieten.

Nachteile:

* Steckbretter sind anfälliger für Fehler und Fehlfunktionen als Lötverbindungen, da die Kontakte auf den Steckbrettern oft instabil sein können, insbesondere bei häufigem Gebrauch oder hohen Belastungen.
* Es kann schwierig sein, eine Schaltung ordentlich und organisiert auf einem Steckbrett aufzubauen, insbesondere bei komplexeren Schaltungen.
* Die Verdrahtung auf einem Steckbrett kann unordentlich und schwer zu lesen sein, insbesondere bei Schaltungen mit vielen Bauteilen und Verbindungen.
* Die Kontakte auf Steckbrettern können im Laufe der Zeit oxidiert oder korrodiert werden, was zu einer schlechteren Leitfähigkeit und Fehlfunktionen führen kann.

Die LED

Die LED hat ein Plus pol die Anode und ein Minus pol die Kathode. Merksatz: „Plus ist die Anode, weil da der Strom ankommt.“ Man erkennt den die Kathode meistens daran das der Pin kürzer ist oder die Seite abgeflacht ist. Jetzt seid ihr dran, versuch doch jetzt einmal die LED zum Leuchten zu bringen. Wie ihr jetzt gesehen habt, ist die LED kurz hell geworden und danach wieder aus. Das Liegt daran, dass sie „Durchgebrannt“ ist. Wie kann man das jetzt beheben? Ganz einfach, man schaltet vor den Plus pol. Aber wie berechnet man einen Widerstand? Dazu benötigt man ein paar Informationen.

Diese währen: {Siehe PowerPoint Folie 10} Was Forward Voltage und DC Forward Current ist erkläre ich später.

Dies Daten muss man dann in die Formel einsetzen. Die Formel bezieht sich auf das Ohm’sches Gesetz, die wichtigste Formel in der Elektronik. Die Formel lautet: **R = (V-V\_LED) / I**.

Wir rechnen jetzt das Beispiel mit dieser Roten LED. Dazu brauchen wir das Datasheet von der LED, das habe ich euch hier mitgebracht. Auf diesem sehen wir dann die Werte von Forward Voltage und DC Forward Current. Dann müssen wir nur noch schauen mit wie viel Volt unsere Quelle hat. Wichtig ist I in A anzugeben und nicht in mA.

In diesem Fall mit 9V:

**( 9V – 2.0V ) / 0.025A = 280.0 Ohm**

Bei 5V:

**( 5V – 2.0V ) / 0.025A = 120.0 Ohm**

Mit diesen Werten kann man jetzt den widerstand bestimmen, in dem an auf die Farbcodes achtet. Die Hintergrund Farbe hat nur den Unterscheid das das Material, aus dem der Widerstand besteht, ein anderer ist. Die Blauen bestehen aus Metallschichten innen und die Braunen aus Kohlenstoffschiechten, wichtig ist das aber für uns nicht.

So jetzt noch ein Tipp von mir, wenn ich etwas ausprobiere, verwende ich immer erstmal einen 1k widerstand, weil bei dem ist man auf der sicheren Seite. Jetzt könnt ihr es mit eurem neuen Wissen erneut versuchen, und vergesst nicht eine neue LED zu verwenden denn die andere ist kaputt.

**Microcontroller**

Was ist ein Microcontroller und was kann er?

Ein Mikrocontroller ist ein kleiner Computer, der in der Lage ist, eine Vielzahl von Aufgaben auszuführen. Er besteht aus einem Prozessor, Speicher, Ein- und Ausgabe-Ports. Mikrocontroller werden oft in Geräten eingesetzt, die eine gewisse Intelligenz erfordern, wie zum Beispiel in Haushaltsgeräten, Spielzeugen, Uhren, Automobilsteuerungen und vielem mehr. Sie können auch in industriellen Anwendungen wie Überwachungs- und Steuerungssystemen eingesetzt werden. Das, was wir verwenden, sind eigentlich keine Microcontroller, sondern Entwicklerbads.

Um es zu vereinfachen, lesen wir jetzt den Lückentext und füllen es gemeinsam aus.

Alternativen im Betrieb?

Im Betrieb werden Microcontroller verwendet, das sind aber andere. Das Hier ist der Microcontroller auf dem Arduino {Zeigen}, das andere ist ein Speicherchip. Ich habe euch mal einen par Beispiele aus dem Betrieb mitgebracht. {Zeigen was der Microcontroller}

Kommen wir jetzt kurz zu der Versorgung des Microcontrollers. Er hat eine DC-Steckerbuchse {Zeigen} bei der man meistens eine 9V Batterie anstecken kann, das wenn man ein Skript hochlädt es auch verwenden kann, wenn der Arduino nicht am Computer angeschlossen ist. Das ist ein Großer Vorteil, mit dieser Funktion kann man den Arduino ohne extra Stromversorgung Programmieren und testen, Wenn es dann funktioniert, kann man den Prototypen mit einer Batterie versorgen und ist nicht an einen Computer gebunden. Der Arduino hat auch zwei Ausgänge. Einen 5V und einen 3.3V. Die werden benötigt, wenn man größere Bauteile verwendet. Da ist es auch wichtig wieder auf das Datenblatt des Bauteils zu schauen wie viel Strom es aushält. Kommen wir jetzt direkt zu so einem Bauteil. Das LCD mit I2C.

DAS LCD mit I2C

Das LCD heißt ausgeschrieben LiquidCrystalDisplay und ist einfach nur die Technik wie es funktioniert. Aber was ist die I2C-Schnitsttelle? Ausgeschrieben heißt es  Inter-Integrated Circuit und ist Kommunikationsschnittstelle.

Bei einem LCD kann es von 16 Pins auf 2 Daten Pins und 2 Versorgung Pins reduzieren. Das erleichtert einem die Arbeit und Spart viele Pins am Microcontroller. Die Adresse ist bei jedem Typ von Geräten gleich, aber trotzdem sollt es mit dem Datasheet abgleichen. Im Normalfall ist die I2C Adresse beim 16x2 LCD **0x27**.

Kommen wir nun zum LCD. So sieht es aus {Zeigen}, sucht es mal aus eurem Baukasten raus und verbindet es mit 4 kabeln. Die Anschlüsse am I2C des LCD müssen an verschieden Ports am Arduino. Hier eine Liste.

Wenn ihr jetzt alles verbunden habt, müssen wir noch den Arduino Programmieren.

Ich zeige euch jetzt den code in verschieden teilen, ich müsst ihn nur untereinander abschreiben. Text mit Pfeilen davor müsst ihr nicht mitschreiben. Während ihr abschreibt, erkläre ich euch den Code.

Wichtig zu sagen: Mit delay das Skript zu pausieren ist eigentlich nicht richtig, weil bei delay der ganze Prozessor angehalten wird. Optimal wäre es mit einem Timer, das überschreitet aber den Rahmen der GFS.

Wenn das Skript funktioniert, erweitert es mit diesen Aufgaben. Wenn ihr Fragen oder Probleme habt meldet euch dann komme ich. Kleiner Tipp bei dem Teil mit dem eine dritte nachriecht hinzufügen, ihr müsst nichts neuschreiben, es reicht, wenn ihr den richtigen teil an die richtige Stelle hin kopiert.

Der Servo-Motor

Der Unterschied ist das der Servomotor ist Modelbaumotor. Das heißt man kann ihn nur auf einen bestimmen Winkel einstellen und nicht durchlaufen lassen.

Warum sind bei den Knöpfen Widerstände? Die sind dafür da Fehler zu vermeiden. Wenn man die Widerstände nicht verwendet, kann es passieren das der Arduino mein das der Knopf gedrückt wurde.

Um den Code zu verstehen ist es wichtig einige Zeilen zu verstehen. „**servo.write();“** -> gibt den Winkel an. „**if (digitalRead(TASTER1\_PIN) == HIGH) {}**“ -> Wenn Taster1 in Strom ankommt, also wenn er gedrückt ist, führt er alles, was in der geschweiften klammer steht. Jetzt sehen wir auch für was „**Serial.begin(9600);**“ ist. Nämlich kann man mit „**Serial.println("");**“ in die Serial Konsole schreiben. Serial kann man aber auch dem Arduino daten übertragen und dadurch auch sagen, was er machen soll. z.B. Schreibe die Zahl 1-play, 2-stop, 3-Weiter, 4-zurück.

Ich zeige euch jetzt den gesamten Code und ihr sollt in wieder abschreiben, den nächsten Code müsst ihr selbst schreiben.

{Code erklären} {Bei Teil 4 bei … = Rückwärts}

Jetzt probiert es aus, wenn ihr alles richtig gemacht habt, sollte es funktionieren.

Das Coden

Jetzt kommen wir dazu, dass ihr etwas selbst codet, und ich gebe euch nur ein paar Zeilen, die ihr dann richtig verwänden, müsst. Aber davor dürft ihr nochmal eine LED ohne Arduino an und ausschalten lassen. Hier nochmal der alteschaltplan, ihr müsst nur ein Bauteil ergänzen. Bei Fragen meldet euch bitte. {Wenn alle fertig dann Muster Lösung zeigen}

Aber jetzt geht es ans Coden.

Baut erstmal die Schaltung nach. Als Widerstand könnt ihr einen 1k Widerstand verwenden.

Meine Tipps und Hinweise für euch sind:

* Speichert die Pin-Nummer als Variabel, so könnt ihr sie oben ändern und sie wird überall im Skript angepasst.
* Ihr müsst den Pin auf Output stellen: pinMode(ledPin, OUTPUT);
* Verstell es mit digitalWrite(ledPin, HIGH); HIGH or LOW ist an oder aus.
* Ihr dürft delay verenden

Wenn alle es fertig haben, machen wir das die LED nur blinkt, wenn der Knopf gedrückt ist

Das ist der neue Aufbau, die LED könnt ihr lassen ihr müsst nur den Knopf hinzufügen

Zum Code ich gebe euch die neuer Loop Funktion zuteil vor, sonst würde es den Rahme dieser GFS sprengen. {Loop Funktion zeigen} Trotzdem müsst ihr die variable für den Knopf-Pin selbererstellen und auch den pinmode festlegen mit Input.

Das, was ihr bei der LED jetzt gelernt habt, könnt ihr jetzt auf ein anderes Beispiel anwenden

Die Relais

Relais sind kleine Bauteile, die den Stromfluss ändern können. Mann hat in der Mitte den Input in die Relais und links und rechts davon jeweils einen Ausgang und je nachdem wie es geschalten ist, ändert es den Output. Die Relais Platinen, die wir verwenden haben zudem noch drei Steckplätze:

* +; Die Versorgung der Platine.
* -; Die Versorgung der Platine.
* S; Zum Schalten.

Die Relais Schalten sich wir LED, wenn S auf HIGH ist der Output ein anderer, als wenn es nicht eingesteckt ist oder auf LOW. Wie finden wir am besten heraus wie es Wenn es Geschalten oder normal ist? Wer Stecken es ab damit wir wissen es ist auf normal/nichtgeschalten. Dann messen wir es mit dem Messgerät. In die Mitte kommt der Rote „Stift“ und den Schwarzen „Stift“ machen wir auf egal welche Seite. Wenn es piepst, würde da Strom fliesen. {Zeigen, und auch das es geschalten ist die andere Seite ist. Beim Zeigen nicht mit Arduino, sondern überbrücken}

Jetzt dürft ihr die Relais einbauen anstatt der LED. Achtet darauf das der Arduino in der Zeit kein Strom hat. Wenn ihr das geschafft habt, entfernt ihr im Skript den Blink teil, lasst nur die Zeile mit dem digitalWrite auf HIGH damit bei Knopfdruck die Relais geschalten werden. Fügt nun vorne an die Relais eine Rote und eine Grüne LED hinzu. Wenn das Signal bei S LOW ist, soll die Rote LED leuchten und wenn es auf HIGH ist die Grüne. Vergesst die Widerstände nicht. Die Katoden von beiden LEDs könnt ihr auf die – reihe des Steckbrettes anschließen soweit ihr sie verwendet. Genauso mit den beiden + Plätzen könnt ihr an der Seite anschließen. Ich kann nur empfehlen die Steckbrett leiterbahnen an den Seiten zu verwenden.

Mein Projekt

Was macht mein kleiner MP3 Player?

Wenn er Strom bekommt, wird automatisch der erste Song von der SD-Card abgespielt. Mit den Tasten kann man dann verstellen welcher Song abgespielt werden soll. Es gibt nur vor und zurück, eine direkte Song Auswahl gibt es nicht. User dem kann man die Lautstärke regeln und Pause machen. Es gibt leider kein Bildschirm, der den Titel anzeigt, da man mit dem DF-Player Mini die Title in Zahlen angeben muss(0001,0002, …).

Für meinen Kleine MP3-Player habe ich verwendet:

* 1x Arduino UNO
* 1x DF Player Mini
* 1x 220Ohm Widerstand
* 2x 3Watt 8Ohm Lautsprecher
* 5x Knöpfe
* 5x 10k Ohm Widerstand
* 2x Steckbretter
* Jumperkabel

Was macht der DF-Player?

Der DF-Player kann von einer SD-Karte Audio Dateien abspielen. Außerdem kann die SD-Karte lesen und daher auch Audiodateien mit bestimmten Namen abspielen und der DF-Player kann den Output der Lautstärke regeln.

Habt ihr noch Fragen?

Vielen Dank für eure Aufmerksamkeit!