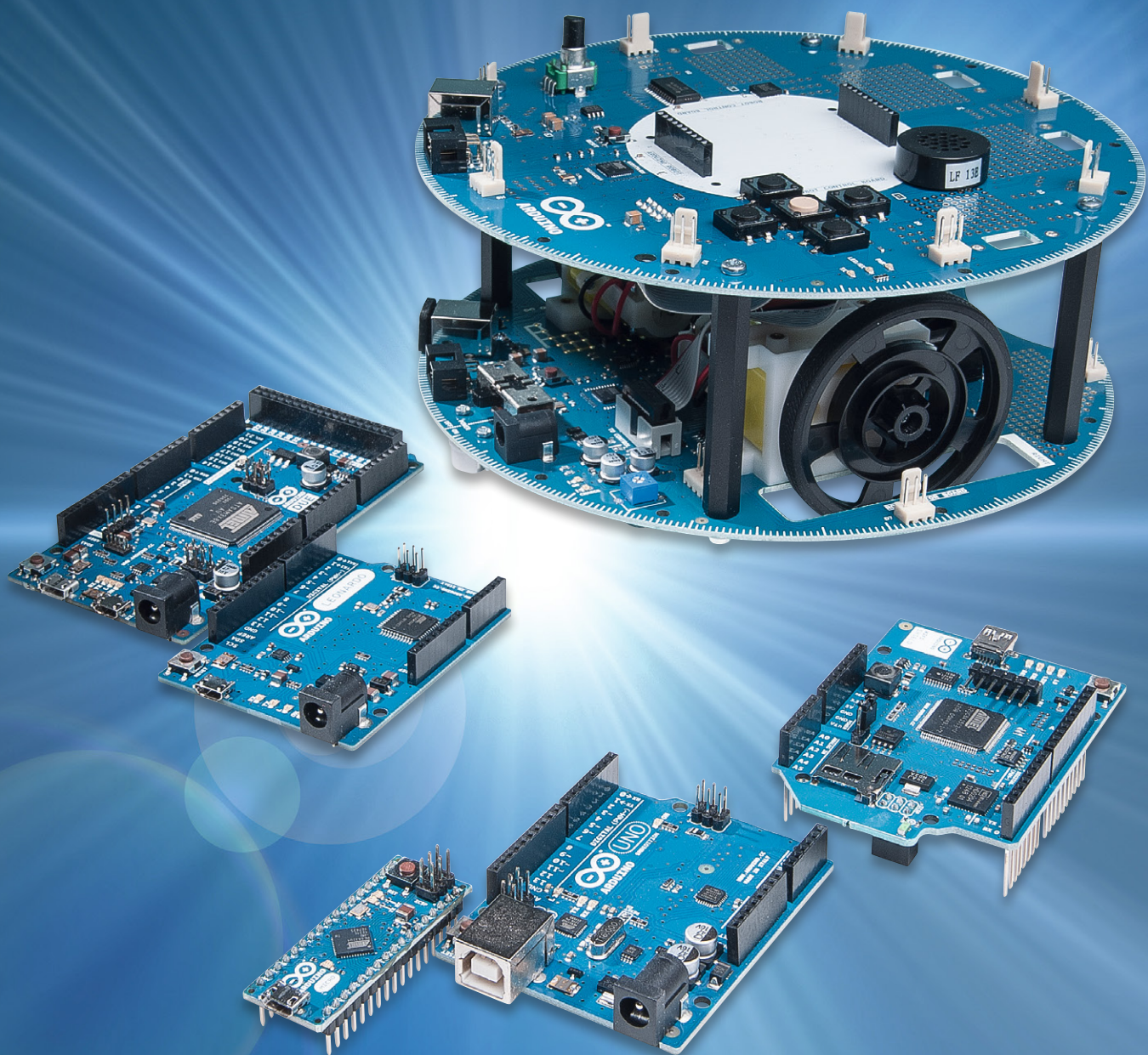




Arduino verstehen und anwenden

Teil 13: Peripheriegeräte: Piezos, Taster, Motoren und Elektromagnete





Bislang wurden überwiegend LEDs als Ausgabemedien für den Arduino eingesetzt. Beginnend mit der bereits auf dem Arduino vorhandenen „LED 13“ über eine einzelne externe LED bis hin zu Lauflichtern mit einer Vielzahl von Leuchtdioden konnten so viele interessante Versuche aber auch nützliche kleine Projekte umgesetzt werden.

In diesem Beitrag sollen nun weitere Ausgabe-Einheiten an das Arduino-Board angeschlossen werden. Neben Komponenten, die vom Arduino gesteuert werden können, wie etwa Elektromotoren oder -magnete, wird die Verwendung von Eingabegeräten wie Taster und Schalter erläutert. Sogenannte Piezokristalle nehmen hier eine Sonderstellung ein, da sie sowohl als Eingabe- als auch als Ausgabemedien dienen können.

Zusammenfassend werden diese Ein- und Ausgabemedien auch als Peripheriegeräte oder -komponenten bezeichnet. Sie erlauben eine gewisse Einflussnahme des arduino-internen Microcontrollers auf seine Umgebung. Insofern führt dieser Artikel auch in das weite und überaus interessante Gebiet des „Physical Computings“ ein.

Arduino musikalisch

Im Bereich der Mikrocontrolleranwendungen möchte man häufig Daten oder Signale nicht nur visuell darstellen sondern auch akustisch ausgeben. Mit dem Arduino ist das problemlos möglich.

Dazu wird lediglich ein einfacher Schallwandler benötigt. Die einfachste Version eines solchen Wandlers stellt eine piezoelektrische Keramik dar. Legt man eine Wechselfspannung an diese Keramik, so beginnt sie im Takt der Spannung zu schwingen und gibt so einen hörbaren Ton ab.

Bild 1 zeigt zwei verschiedene Versionen von gängigen Schallwandlern, einmal ohne Gehäuse und einmal mit einlötbaren Pins und schwarzem Rundgehäuse.

Da ein piezoelektrischer Schallwandler eine vergleichsweise hohe Impedanz aufweist, kann er direkt mit einem Mikrocontrollerausgang verbunden werden.

Praxisanwendung: Arduino als elektronische Grußkarte

Das folgende Programm spielt eine Melodie auf dem Piezo-Lautsprecher ab. Der piezoelektrische Schallwandler muss hierfür mit GND und Digital-Pin D9 ver-



Bild 1: Piezoelektrische Schallwandler

bunden werden. Nach dem Laden des Programms ertönt die Melodie aus dem Piezo-Lautsprecher.

```
// Melody

int speaker=9;           // select speaker channel
int len = 300, del = 200; // tone length and delay

// tone frequencies
int A = 440, H = 494;
int C = 523, d = 587, e = 659, f = 698;
int g = 784, a = 880, h = 988, c = 1047;

void setup()
{ pinMode(speaker, OUTPUT);
}

void loop()
{ tone(speaker,C,len); delay(del); tone(speaker,d,len); delay(del);
  tone(speaker,e,len); delay(del); tone(speaker,f,len); delay(del);
  tone(speaker,g,len); delay(del); tone(speaker,f,len); delay(del);
  tone(speaker,e,len); delay(del); tone(speaker,d,len); delay(del);
  tone(speaker,d,len); delay(del); tone(speaker,e,len); delay(del);
  tone(speaker,C,len); delay(del); delay(3*del);
}
```

Alarmsignale

Das folgende Programm zeigt, wie man mit dem Arduino ein durchdringendes Alarmsignal erzeugen kann. Die Belegung der Anschlüsse ist unverändert.



```
// ALARM!

int Speaker=9;
int i=0;

void setup()
{ pinMode(Speaker, OUTPUT);
}

void loop()
{ for(i=120;i<1600;i++)
  { tone(Speaker,i,10);
    //delay(100);
  }
}
```

Diese Praxisübung demonstriert, dass die Tonausgabe sehr flexibel programmierbar ist. Es gibt keinerlei Beschränkungen auf einfache Tonfolgen, mit etwas Phantasie und Geduld können auch sehr eindrucksvolle Klänge erzeugt werden.

Umgekehrt erzeugt ein Piezo-Element eine elektrische Spannung, wenn es mechanisch belastet wird. Wie diese Spannung als Steuersignal für den Arduino verwendet werden kann, ist beispielsweise in [1] erläutert.

Taster und Tasterprellen

Tastenabfragen durch einen μC sind ein typisches Beispiel dafür, dass auch scheinbar einfache Schaltungen ihre Tücken haben.

Testen Sie das Programm der folgenden Praxisübung. Schließen Sie den Taster zwischen Digital-Pin 3 und GND an. Führen Sie dann das Programm aus und beobachten Sie dabei LED 13 und gleichzeitig die Ausgabe auf dem seriellen Monitor.

```
// Bouncing

byte n=0;
int SW1=3;
int LED=13;
int Toggle=0;

void setup()
{ Serial.begin(9600);
  pinMode(SW1,INPUT);
  digitalWrite(SW1,HIGH);
  Serial.println("Bouncing");
}

void loop()
{ if(!digitalRead(SW1))
  { n++;
    Serial.print("Switch pressed ");
    Serial.print(n,DEC);
    Serial.println(" times");
    Toggle=!Toggle;
    digitalWrite(LED,Toggle);
    do {} while(!digitalRead(SW1));
  }
}
```

Welchen Wert zeigt die Anzeige im seriellen Monitor, wenn man den Taster beispielsweise 10-mal schnell hintereinander betätigt? Vermutlich liegt der Anzeigewert irgendwo zwischen 12 und 15!

Was ist passiert? Eigentlich sollte doch der Anzeigewert genau mit der Anzahl der Tastendrucke übereinstimmen! Mit

```
{ if(!digitalRead(SW1))
```

wird gewartet, bis der Taster den Pin3 auf LOW-Signal zieht. Dann wird das Programm weiter durchlaufen, bis in der Zeile

```
do {} while(!digitalRead(SW1));
```

darauf gewartet wird, bis der Taster wieder frei gegeben ist. Wie kann es also dabei zu einem mehrfachen Durchlauf der Zählschleife bei einem einzelnen Tastendruck kommen?

Das Problem liegt im Taster selbst. Die elektrischen Kontakte des Tasters liefern beim Betätigen nicht nur einen einmaligen Kontakt, sondern federn mehrmals zurück. Das Resultat ist kein einzelner Pegelwechsel, sondern ein ganzer Pulszug. Bild 2 zeigt das Oszillogramm eines Ausschaltvorgangs. Deutlich sind mehrere „spikes“ zu erkennen, die der Mikrocontroller natürlich korrekterweise als separate Tastendrucke interpretiert. Dieses problematische Verhalten von mechanischen Tastern und Schaltern ist auch als „Bouncing“ (engl. für „Prellen“) bekannt.

Dieses Fehlverhalten kann auf verschiedene Weisen verbessert werden. Neben speziellen schaltungs-technischen Änderungen, wie entsprechend dimensionierten RC-Gliedern, kann insbesondere auch eine sogenannte Software-Entprellung eingesetzt werden.

Im nachfolgenden Sketch wird nach einer Wartezeit von 50 ms der Taster-Status erneut abgefragt. Nur wenn sich zweimal der gleiche Pegel ergibt, wird die Zählvariable erhöht. Spikes innerhalb der 50 ms haben damit keinen Einfluss auf das Zählergebnis.



Bild 2: Kontaktprellen



Wichtiger Hinweis:

Vorsicht! Aufgrund der im Gerät frei geführten Netzspannung dürfen Aufbau und Inbetriebnahme ausschließlich von unterwiesenen Elektrofachkräften durchgeführt werden, die aufgrund ihrer Ausbildung dazu befugt sind. Die einschlägigen Sicherheits- und VDE-Bestimmungen sind unbedingt zu beachten.

```
// Debouncing

byte n=0;      // counter
int SW1=3;
int LED=13;
int Toggle=0;
byte value_1, value_2=0;

void setup()
{ Serial.begin(9600);
  pinMode(SW1,INPUT);
  digitalWrite(SW1,HIGH);
  pinMode(LED,OUTPUT);
  Serial.println("Debounce");
}

void loop()
{ value_1=digitalRead(SW1);
  if(!value_1)
  { delay(50);
    value_2=digitalRead(SW1);
    if(!value_2)
    { n++;
      Serial.print("Switch was pressed ");
      Serial.print(n,DEC);
      Serial.println(" times");
      Toggle=!Toggle;
      digitalWrite(LED,Toggle);
      do{ }while(!digitalRead(SW1));
    }
  }
}
```

Der Anzeigewert und die tatsächliche Anzahl von Tastenaktionen sollten nun deutlich besser übereinstimmen.

Ansteuerung von Relais und Elektromagneten

Ähnlich wie mit Tastern verhält es sich auch mit Relais. Obwohl es sich hier um Bauelemente handelt,

die seit Jahrzehnten erfolgreich in der Elektrotechnik eingesetzt werden, treten in der Kombination mit Mikrocontrollern unerwartete Probleme auf.

Relais sollten aus mehreren Gründen nicht direkt an den Port eines Mikrocontrollers angeschlossen werden. Gängige Relais ziehen Ströme von 100 mA und mehr. Wie bereits aus früheren Artikeln bekannt ist, sollte ein Port nicht mit mehr als 20 mA belastet werden. Außerdem erzeugen die in einem Relais enthaltenen Spulen starke Induktionsspitzen, welche die interne Treiberschaltung eines Controllerausgangs leicht zerstören können.

Beide Probleme können mithilfe eines einfachen Kleinleistungstransistors gelöst werden. Geeignete Typen sind hier etwa der BC547 oder der 2N2222. Zum Abfangen der Induktionsspitzen ist dann noch eine sogenannte Freilaufdiode erforderlich. Universaltypen wie 1N4148 oder 1N4007 sind hier meist vollkommen ausreichend. Bild 3 zeigt die zugehörige Schaltung.

An U_{relais} muss die für das verwendete Relais erforderliche Schaltspannung, meist 5 bis 12 V, angelegt werden.

Mit diesem einfachen Aufbau können nun auch Ströme von mehreren Ampere oder Netzspannungen geschaltet werden. Hierbei ist natürlich immer darauf zu achten, dass die Schaltkontakte des verwendeten Relais für die gewünschten Stromstärken und Spannungen ausgelegt sein müssen. Bild 4 zeigt den zugehörigen Aufbau mit einem klassischen einpoligen Relais.

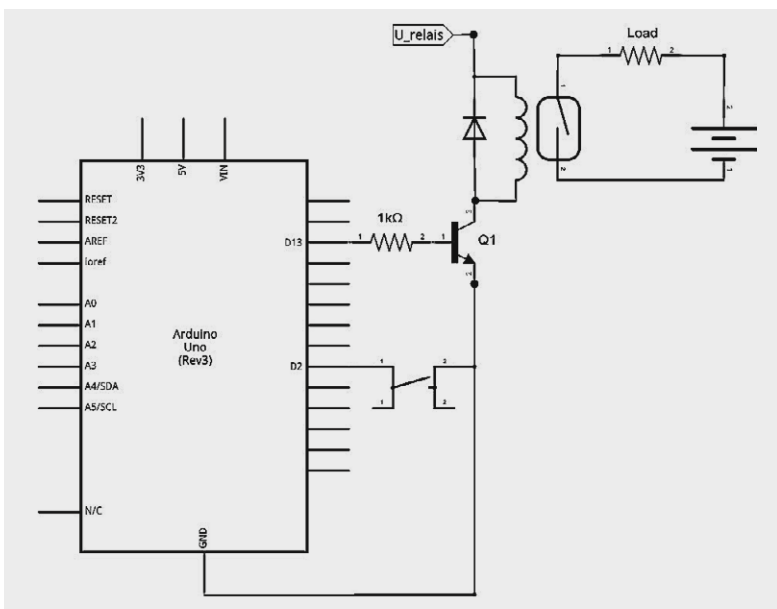


Bild 3: Relais am Arduino

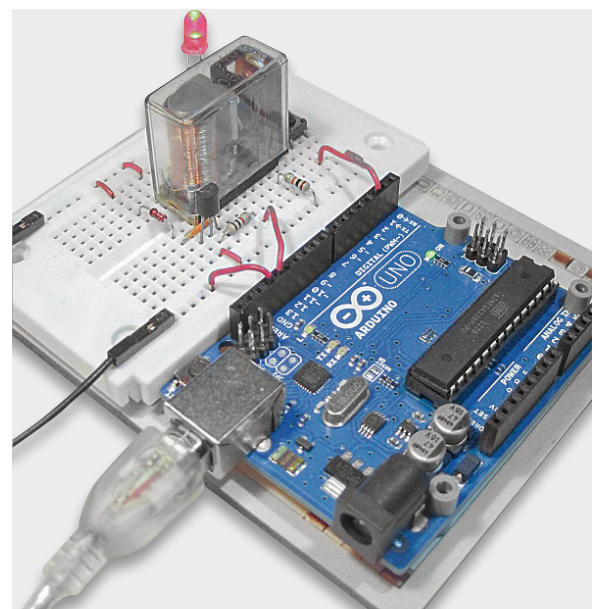


Bild 4: Relais am Arduino

Der Sketch dazu sieht so aus:

```
// switch relay via button including switch-off delay

int button = 2, relay = 13;
int stateRelay = LOW, stateButton, previous = LOW;
long time = 0, debounce = 500;
int stayON = 5000; //stay on for 5000 ms

void setup()
{ pinMode(button, INPUT_PULLUP); pinMode(relay, OUTPUT);
}

void loop()
{ stateButton = digitalRead(button);

  if(stateButton == LOW && previous == HIGH && millis()
  - time > debounce)
  { if(stateRelay == HIGH)
    { delay(stayON); stateRelay = LOW;
    }
    else stateRelay = HIGH;

    time = millis();
  }
  digitalWrite(relay, stateRelay);
  previous = stateButton;
}
```

Man erkennt, dass auch hier wieder eine softwaretechnische Entprellung vorgenommen wurde.

Neben der einfachen Relaissteuerung enthält der Sketch auch eine Ausschaltverzögerung. Diese lässt etwa das Licht in einem Treppenhaus nach den Tastendruck noch einige Zeit nachleuchten.

Ist die Ausschaltverzögerung nicht erwünscht, so kann die Programm-Zeile

```
delay(stayON);
```

einfach gelöscht werden.

Neben Relais können so auch Elektromagneten angesteuert werden. Auf diese Weise kann der Arduino beispielsweise auch die Entriegelung einer Türschließenanlage betätigen.

Drehzahlsteuerung von Elektromotoren

Analog zu den Elektromagneten können auch Gleichstrommotoren über den Arduino gesteuert werden. Motoren geringer Leistung können wieder über Kleinsignaltransistoren angesteuert werden. Für stärkere Motoren sind entsprechend ausgelegte Transistoren oder FETs erforderlich.

Ein großer Vorteil bei der Ansteuerung von Gleichstrommotoren mit dem Arduino besteht darin, dass die Motoren nicht nur ein- und ausgeschaltet werden können, sondern sogar eine Drehzahlsteuerung möglich ist. Wird der Motor über einen PWM-Ausgang mit dem Arduino verbunden, kann die Motorleistung in weiten Grenzen eingestellt werden.

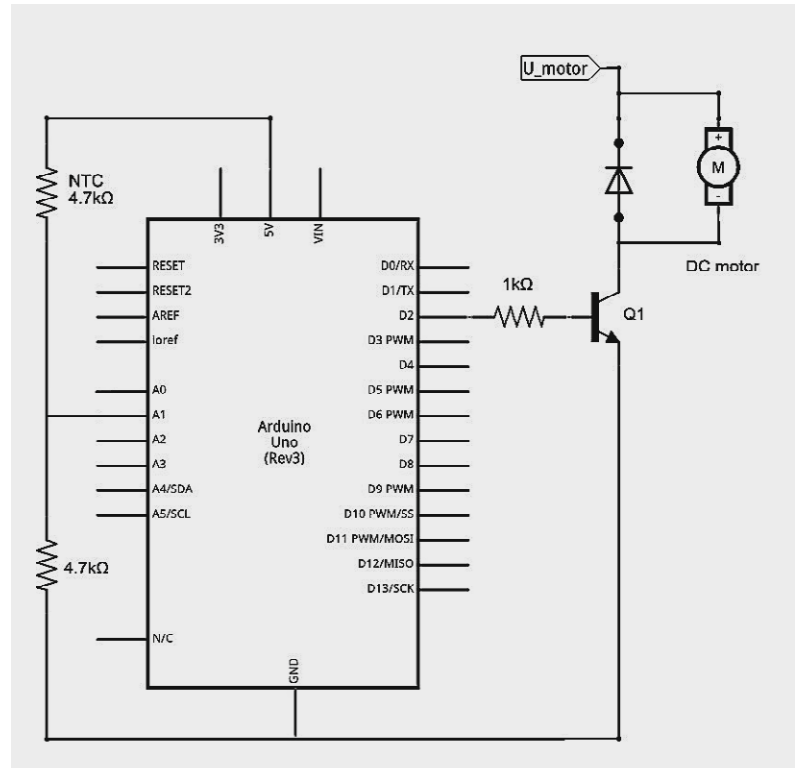


Bild 5: Ventilatorsteuerung via Arduino

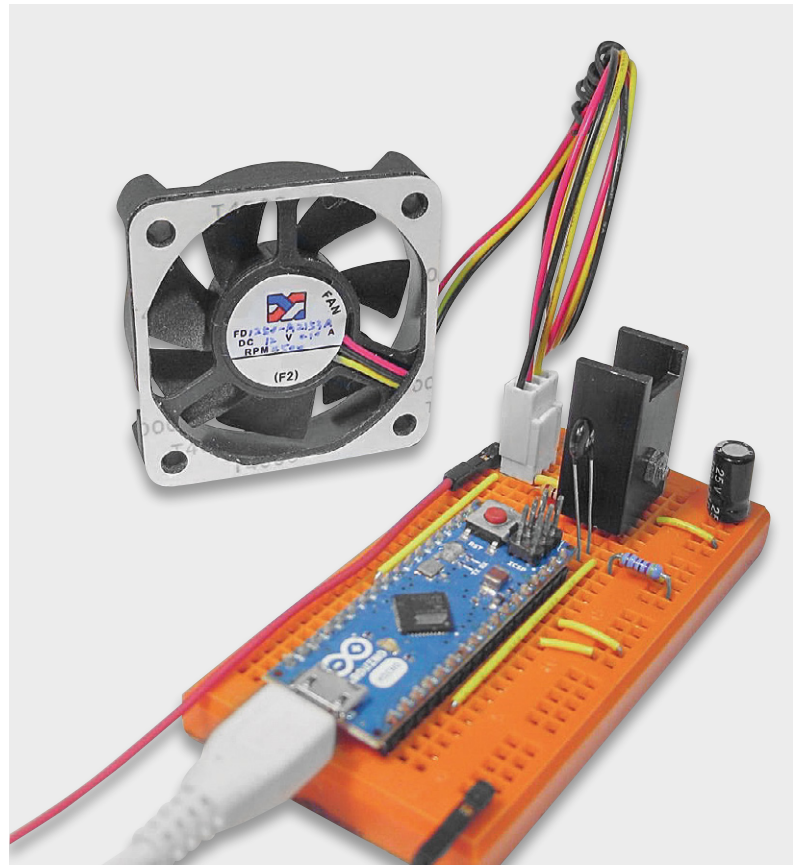


Bild 6: Aufbaubild zur Ventilatorsteuerung

Die Pulsweitenmodulation hat zudem den Vorteil, dass im Steuertransistor im Vergleich zu einer linearen Spannungssteuerung sehr wenig Leistung verloren geht. Zudem erreicht ein PWM-gesteuerter Motor auch bei geringen Drehzahlen bereits ein großes Drehmoment. So können Roboterfahrzeuge, Rollladensteuerungen oder Wasserpumpen sehr feinfühlig angesteuert werden.



Die Ansteuerung des Motors erfolgt wieder über einen Transistor, je nach Stromaufnahme des Motors kann hier auch ein Leistungstransistor wie etwa der BD135 erforderlich sein.

Als Temperatursensor wurde hier ein einfacher NTC eingesetzt. Details zu diesem Sensortyp finden sich im Artikel 6: „Sensortechnik und Messwerterfassung“ zu dieser Serie (ELVjournal 5/2014).

Der zugehörige Sketch liest den Spannungswert am Analogeingang A1 aus und rechnet diesen in einen Temperaturwert um. Über einen Vergleich mit dem Schwellwert „tempLimit“ wird dann der Ventilator ein- bzw. ausgeschaltet. Bei Bedarf kann hier natürlich auch noch eine gewisse Hysterese vorgesehen werden, um übermäßig häufiges Schalten in der Nähe der Umschalttemperatur zu verhindern.

Möchte man die Motorleistung in Abhängigkeit von der Temperatur regeln, so kann man den Steuertransistor auch an einen PWM-Ausgang anschließen und die Pulsweite der Spannung entsprechend der gemessenen Umgebungstemperatur erhöhen oder reduzieren.

```
// fan control
// using P5V to 4k7 NTC temperature sensor @ ADC1, 4k7 to GND

int fanPin = 2;
int offset = 28; // temperatur calibration - offset
float cal = 0.1135; // temperatur calibration - slope
int ADC_value;
float temp, tempLimit = 25;

void setup()
{ pinMode(fanPin, OUTPUT);
  pinMode(13, OUTPUT);
  Serial.begin(9600);
}

void loop()
{ ADC_value=analogRead(1);
  temp=ADC_value*cal-offset;
  Serial.print("Temp = "); Serial.print(temp,1); Serial.println(" ");
  delay(1000);

  if (temp > tempLimit)
  { digitalWrite(13, HIGH);
    digitalWrite(fanPin, HIGH);
  }
  else
  { digitalWrite(13, LOW);
    digitalWrite(fanPin, LOW);
  }
}
```

Ausblick

Nachdem in diesem Beitrag der Anschluss von verschiedenen Peripheriegeräten erläutert und diskutiert wurde, stehen im nächsten Artikel Sieben-Segment-Displays im Vordergrund. Diese weit verbreiteten Anzeigetypen gestatten die einfache und gut lesbare Anzeige von Zahlen und Ziffern.

Aber auch auf die Steuerung weiterer peripherer Komponenten wird ein späterer Artikel nochmals genauer eingehen. Dort wird es dann auch darum gehen, wie ein Elektromotor in beiden Drehrichtungen gesteuert werden kann und wie sogenannte Schrittmotoren zur präzisen Positionierung von Druckerköpfen oder Roboterarmen eingesetzt werden. **ELV**



Das Bauteilepaket zum Onlinekurs

Praxiskurs
AVR-Mikrocontroller
in C programmieren

Mikrocontroller-Onlinekurs

- ▶ Multimediales Lernen mit Video- & Audio-Sequenzen
- ▶ Umfangreiche Kursunterlagen online verfügbar
- ▶ Mit optimal auf den Kursinhalt abgestimmtem Hardware-Paket
- ▶ Unterstützung durch Online-Forum

12 Lerneinheiten mit individueller Erfolgskontrolle

- Einstieg in die Hardware
- Physical Computing
- USB-Schnittstelle als Kommunikationsmedium
- Programmieren und Erprobungsumgebung
- Programmierung mit C und Processing
- Programmierpraxis
- Schrittmotorpraxis
- Messen, Steuern und Regeln
- Smartcars
- Motor und Encoder
- Schallhandlet, Taster und Dröhler
- Hande Applikation zum Mikrocontroller-System

Ohne Lötarbeiten experimentieren! Bauteile einstecken und starten!

ELV Kompetent in Elektronik

FRANZIS

Empfohlenes Material	Best.-Nr.	Preis
Arduino UNO	CD-10 29 70	€ 27,95
Mikrocontroller-Onlinekurs	CD-10 20 44	€ 99,-

Alle Arduino-Produkte wie Mikrocontroller-Platinen, Shields, Fachbücher und Zubehör finden Sie unter: www.arduino.elv.de



Weitere Infos:

[1] Lernpaket „AVR-Mikrocontroller in C programmieren“, Franzis-Verlag, 2012

Mikrocontroller-Onlinekurs, Franzis-Verlag, exklusiv für ELV, 2011, Best.-Nr.: CD-10 20 44

G. Spanner: Arduino – Schaltungsprojekte für Profis, Elektor-Verlag, 2012
Best.-Nr.: CD-10 94 45

Grundlagen zur elektronischen Schaltungstechnik finden sich in der E-Book-Reihe „Elektronik!“
(<http://www.amazon.de/dp/B000XNCB02>)

G. Spanner: Coole Projekte mit dem Arduino Micro, Franzis-Verlag, 2014