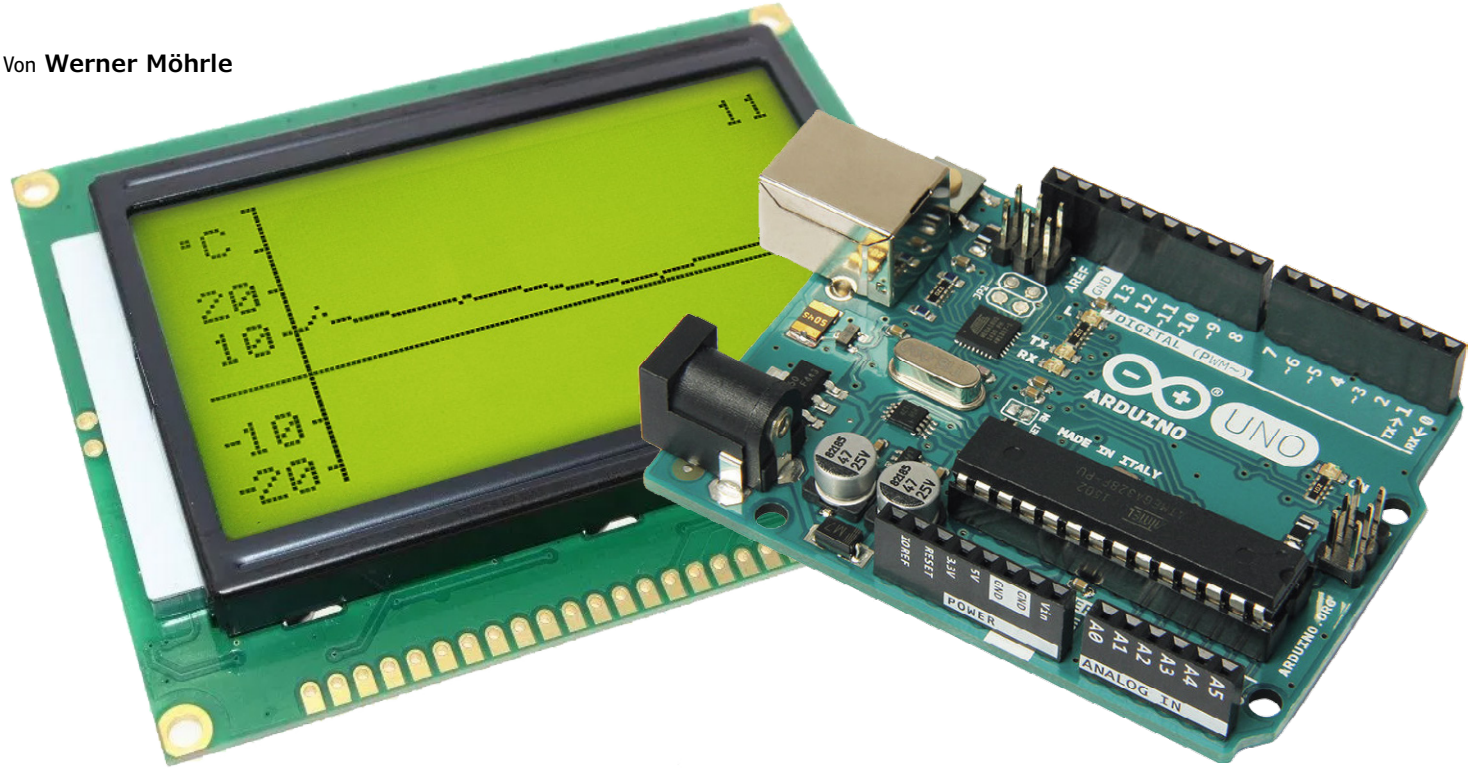
Arduino-Temperaturrekorder

Log around the clock!

Von **Werner Möhrle**



Mit dem Arduino Uno lassen sich schöne kleine Projekte aller Art recht preisgünstig aufbauen. Hier geht es um einen einfachen Temperaturrekorder, der die Messwerte der vergangenen 24 Stunden auf einem Display darstellt. Besondere Aufmerksamkeit verdient die Software, die ohne externe Bibliotheken auskommt.

Der Temperatur-Rekorder misst im Lauf eines Tages die Temperatur und stellt die Messwerte auf einem Grafikdisplay dar. Wenn ein neuer Messwert dazukommt, wird die dargestellte Kurve um ein Pixel nach links verschoben, so dass der älteste Wert verschwindet und der neue erscheint. Das Messintervall lässt sich leicht ändern, um schnellere Temperaturänderungen zum Beispiel bei Regelvorgängen erfassen zu können.
Die Schaltung in **Bild 1** zeigt die (nur) drei Komponenten (und ein paar passive Bauteile) des kleinen Temperaturrekorders.

Programmierbarer Temperatursensor

Der Temperatursensor DS1631 misst Temperaturen zwischen -55 °C und +125 °C, im Bereich 0...70 °C mit einem Fehler von +-0,5 °C. Das Messergebnis wird in einem Register mit einer Auflösung von 9...12 Bit (ganz nach Wunsch des Anwenders) gespeichert und kann dort gelesen werden. Mit einer Versorgungsspannung von 3,0...5,5 V ist der Sensor für den Betrieb an den verschiedensten Mikrocontrollern geeignet. Außergewöhnlich in unserer Zeit ist auch, dass der DS1631 nicht nur als SMD, sondern auch in einem DIP-Gehäuse verfügbar ist. Der DS1631, dessen Adresse durch die Pins A0...A2 programmierbar ist, kommuniziert mit dem Controller über einen I2C-Bus. Die Innenschaltung des Sensors in **Bild 2** zeigt Konfigurationsregister, die vom Controller beschrieben werden und ein Read-only-Temperaturregister, aus dem der Controller die ermittelte Temperatur ausliest. Der Sensor wird vom Arduino-Temperaturrekorder im kontinuierlichen Modus mit einer Messwertauflösung von 12 Bit betrieben.
Mit den beiden Registern TH und TL kann eine Hysterese vorgegeben werden, durch die der DS1631 als Thermostat (mit Schaltausgang TOUT) genutzt werden kann. Diese Funktion wird hier aber nicht genutzt.

Ein passendes Display

Es ist ja heutzutage gang und gäbe, Grafikdisplays mit einem seriellen Bus (I^2C oder SPI) einzusetzen. Da aber schon der Temperatursensor mit einem I^2C-Bus ausgestattet ist und nur drei I/O-Pins des Arduino für sich einnimmt, stehen ausrei-

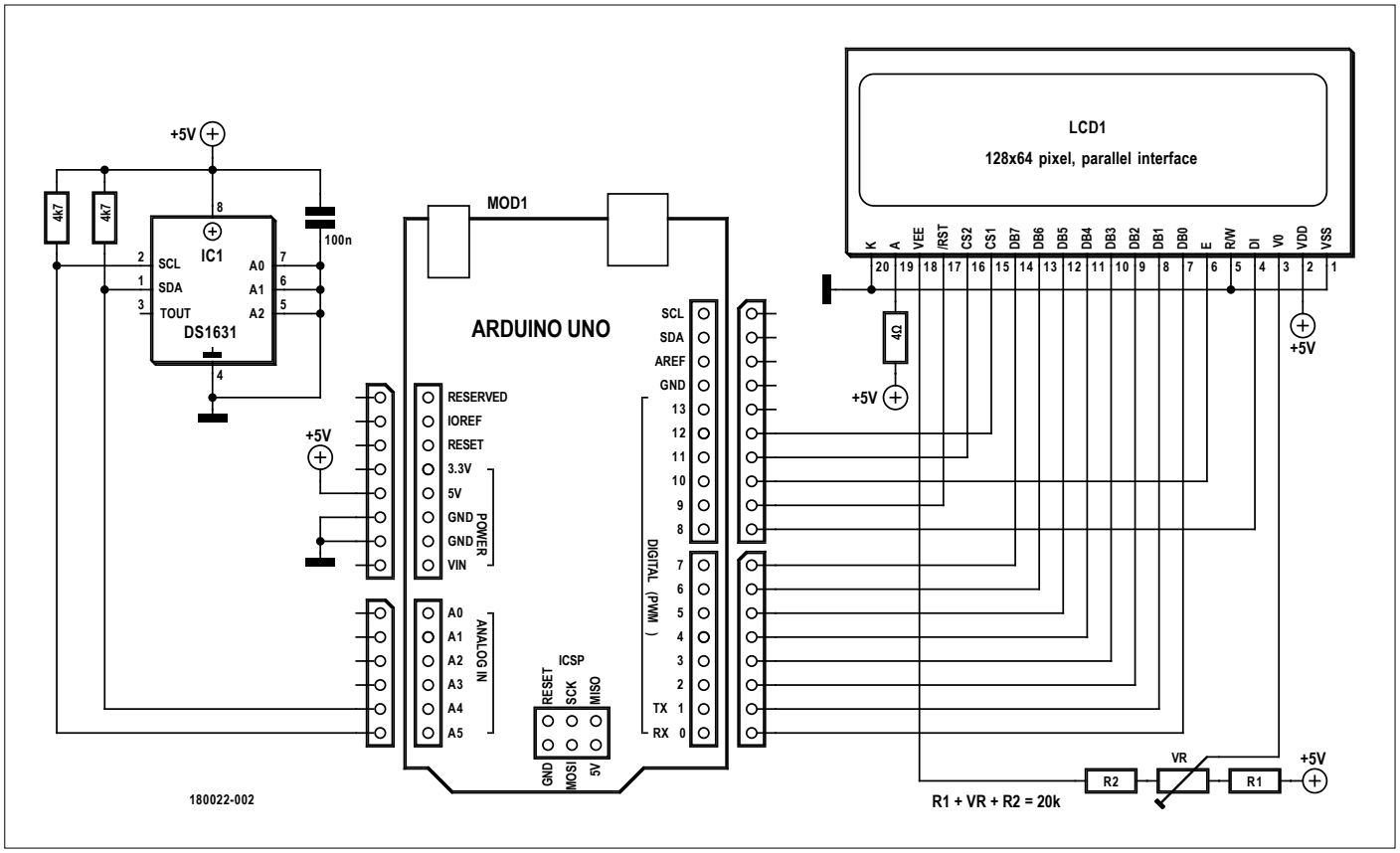


Bild 1. Die dreiteilige Schaltung des Temperaturrekorders.

chend Pins für eine parallele Ansteuerung des Displays zur Verfügung. Vor- oder Nachteile hat das nicht, man spart lediglich ein paar Cent.
Passende Displays mit 128x64 Pixeln gibt es von vielen Herstellern (Visay, Lumex, Winstar...), die für weniger als 10 € bei ebay und Konsorten angeboten werden. Größe und Farbgebung bleiben ganz Ihnen überlassen, Sie müssen lediglich darauf achten, dass die Anschlussbelegung (siehe **Kasten**) passt - es gibt nämlich mehrere Anschlussvarianten.

Im Grafikdisplay (**Bild 3**) wird ein Verlauf von 108 Temperaturmesswerten in einem Koordinatensystem dargestellt. Sollen die 108 Werte 24 Stunden entsprechen, ergibt sich eine zeitliche Auflösung der Darstellung von 800 s, einer knappen Viertelstunde. Die aktuelle Temperatur wird rechts oben im Display eingeblendet.

Das Rechenzentrum

Die dritte Komponente, der Arduino Uno, initiiert die Temperaturmessungen, erfasst die ermittelten Temperaturwerte und rechnet sie in korrekt auf dem Display platzierte Pixel um.

Sketch ohne Bibliotheken

Schon beim ersten Blick auf den gut kommentierten Sketch [1] fällt auf, dass überhaupt keine externen Bibliotheken eingebunden werden. Alle Schreib- und Lesevorgänge sowohl mit dem Sensor als auch mit dem Display werden mit Mitteln vorgenommen, die die Arduino-IDE zur Verfügung stellt.

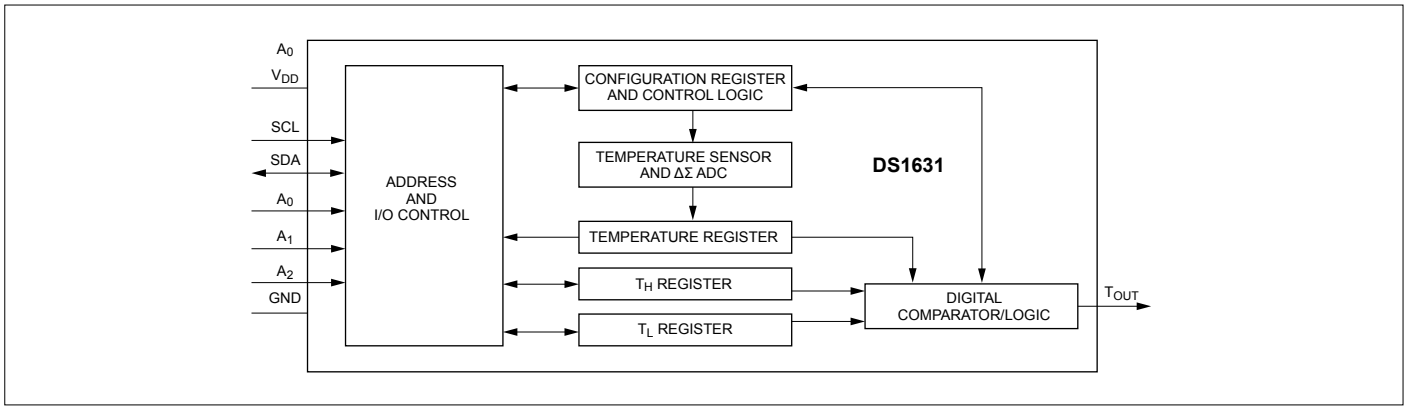


Bild 2. Interne Funktionen des Temperatursensors (Quelle: Maxim Integrated).

Anschlussbelegung des Displays		
Pin	**Symbol**	**Funktion**
1	VSS	Versorgungsspannung 0 V
2	VDD	Versorgungsspannung +5 V
3	V0	Betriebsspannung LCD-Treiber
4	D/I	H: Dateneingang, L: Befehlscode-Eingang
5	R/W	H: Daten lesen, L: Daten schreiben
6	E	Enable (negative Flanke)
7...14	DB0...7	Datenbus
15	CS1	Chip select für IC1 (Segmenttreiber 1...64)
16	CS2	Chip select für IC1 (Segmenttreiber 65...128)
17	RES	Reset (aktiv L)
18	VEE	Ausgang negative Spannung
19	A	LED-Hintergrundbeleuchtung Anode
20	K	LED-Hintergrundbeleuchtung Kathode

Die wichtigsten Funktionen finden sich am Ende des Sketches. Die Initialisierung von Controller, Temperatursensor und Display wird in der Funktion `setup` vorgenommen. In der Hauptroutine `loop` werden nach Aufruf durch den Taktgeber (`meas_flag==1`) die Daten des Temperatursensors ausgelesen (`read_temp`) und daraus der Temperaturmesswert berechnet (`cal_temp`). Bei einer Über/Unterschreitung der darstellbaren Grenzwerte von -23°C und +40°C wird die Variable auf null gesetzt. Daraus werden der Wert der Ordinatenachse berechnet (`cal_measval`), die entsprechenden Daten zum Laden des Displays mit dem neuen Messwert ermittelt (`set_meas(measval)`) und schließlich das Display damit aktualisiert. Als Vorbereitung für die Ausgabe der nächsten Messung werden dann noch die Pixeldaten der Temperaturwerte um ein Pixel nach links verschoben, so dass der älteste Messwert verschwindet.

Taktgeber in dem Sketch ist die Interrupt-Service-Routine ganz am Sketchende. Hier wird das Messintervall auf 800 s eingestellt, sodass die insgesamt 108 darstellbaren Messintervalle in 108*800 s=24 h einmal durchgeschoben werden. Zum Test des Sketches kann man das Intervall zum Beispiel auf 2 s verkürzen.

Die Steuerung des Sensors findet sich ebenfalls ziemlich weit unten im Sketch. Dabei wird die 2-Wire-Hardware-Schnittstelle des Controllers (A4 für SDA und A5 für SCL) genutzt. In der Funktion `write_to_config` wird das Konfigurationsregister beschrieben. Mit `TWDR=0xAC` wird auf das Konfigurationsregister zugegriffen und mit `TWDR=0x0C` der kontinuierliche 12-Bit-Modus gewählt. Dann startet die Funktion `start_convert` den Messbetrieb (`TWDR=0x51`).

Die Funktion `read_temp` sorgt dafür, dass die zwei Bytes des Registers mit dem aktuellen Temperaturwert nacheinander gelesen (`TWCR=0x84`) und in den Variablen `temph` und `templ` gespeichert werden. Anschließend überführt die Funktion `cal_temp` die beiden binären Werte in die vorzeichenbehaftete Float-Variable `temp`.

Die 64 horizontalen Reihen der 128x64-Dot-Matrix des Displays werden auf eine Temperaturskala von -23...+40°C aufgeteilt, die 128 vertikalen Spalten in 19 Dots für die Darstellung der

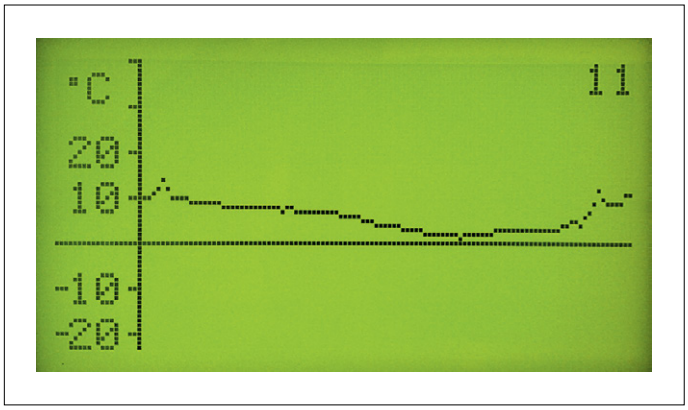


Bild 3. Die Temperaturanzeige mit den 108 Messwerten der letzten 24 Stunden.

Temperaturskala und in 109 Dots für den Temperaturverlauf. Das Display selbst wird über einen Segmenttreiber für die linke Hälfte (CS1) und einen für die rechte (CS2) angesteuert, aufgeteilt in jeweils acht übereinander liegende Pages mit 64x8 Bits.

Kurz über das Programm geflogen

Im Array `num[11][5]` wird festgelegt, wie die Zahlen der im Display rechts oben gezeigten aktuellen Temperatur dargestellt werden, also welche Pixel schwarz sein sollen. Im Array `dmask[8][128]` wird der gesamte Display-Inhalt gespeichert, um die Messwerte nach jeder Messung um einen Punkt nach links verschoben neu ausgeben zu können.

Es folgen Funktionen zur Ansteuerung des Displays: `set_ytics` und `set_ylabels` zeichnen die Achsen mit Beschriftung der Temperaturskala, `shift_left` verschiebt den bisherigen Temperaturverlauf um ein Pixel nach links, `set_meas` ermittelt, mit welchem Datum und an welcher Stelle der neue Messwert in `dmask` geladen wird und `load_gdisp` lädt das Display mit den in diesem Array gespeicherten Daten. Schließlich trägt `load_num` noch den letzten Messwert als Zahl rechts oben ein, sofern in diesem Bereich keine Messwerte liegen (ansonsten weiter nach unten verschoben).

Die Software ist mit deutscher und englischer Kommentierung von der Elektor-Projektseite [1] abrufbar. ⏮

180022-01

Weblink

[1] Elektor-Projektseite:
www.elektormagazine.de/180022-01