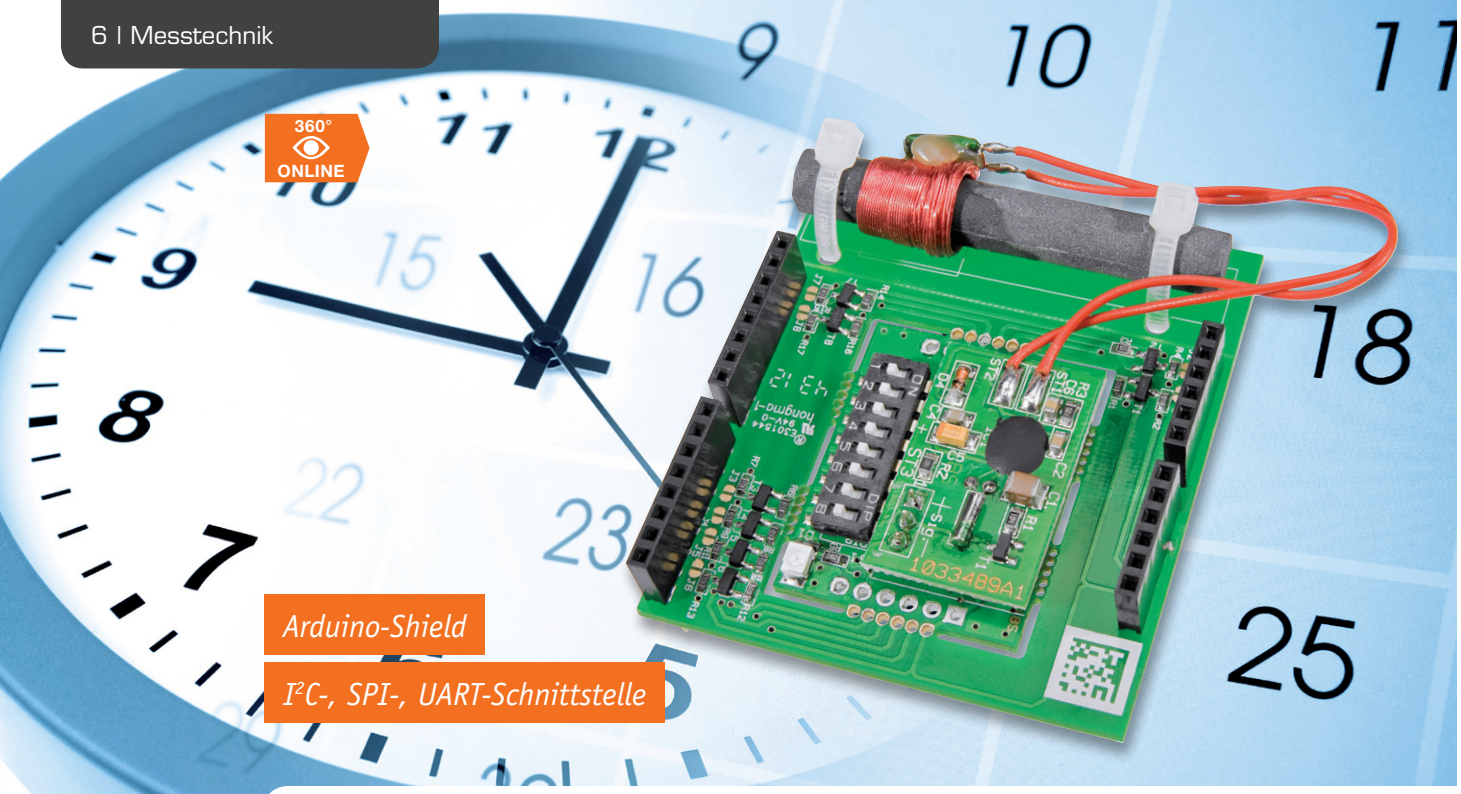


RTC-DCF

Real-Time-Clock mit DCF77-Empfang

Das RTC-DCF ist eine Echtzeituhr, welche mit Hilfe eines DCF77-Empfängers Zeitdaten empfangen und die interne Uhr danach einstellen kann. Aufgrund der drei Kommunikationsschnittstellen I²C, SPI und UART ist das RTC-DCF vielseitig einsetzbar, sowohl in eigenen Schaltungen als auch als Arduino-Shield.

Immer die genaue Zeit

Die Verfügbarkeit genauer Zeitangaben ist bei vielen elektronischen Systemen ein essentielles Feature, ob dies Sicherheitssysteme sind, Datenlogger oder einfache Zeitanzeigesysteme. Meist werden in busbasierten Mikroprozessorsystemen externe Echtzeit-Uhrbausteine (RTC – Real-Time-Clock) wie die DS24xx-/DS13xx-Reihe von Dallas, die RTC724xx-Reihe von Epson oder der PCF8563 von NXP eingesetzt. Diese enthalten in den meisten Fällen auch einen programmierten Kalender, so dass sich auch das komplette Datum auslesen lässt. Das Stellen erfolgt vom Mikrocontroller des jeweiligen Systems durch Beschreiben bestimmter Register, ebenso erfolgt das Auslesen der Zeit über das Ansprechen bestimmter Registeradressen.

Ein gewisses Manko bleibt immer die allein quarzbasierte Zeitsteuerung, trotz hochwertiger Fehleralgorithmen in den RTCs ist hier langfristig eine Drift nicht auszuschließen. Der Idealfall wären hier also entweder der Zugriff auf einen Zeitserver (falls das System Zugang zu einem Netzwerk hat) oder der Bezug genauer Zeitdaten per (periodischem) Funkuhrempfang.

Auf letzterer Möglichkeit basiert unsere RTC-Lösung. Hier haben wir eine stromsparende Kombination eines Low-Power-Mikrocontrollersystems mit integriertem RTC-Baustein und internem EEPROM mit der Synchronisation über einen DCF77-Empfänger gewählt. Der Mikrocontroller ist ein STM8L151 von ST Microelectronics. Dieser verfügt über eine interne RTC mit Kalender mit Alarm-Interrupt und Auto-Wakeup, diese kann über diverse Schnittstellen (SPI, I²C [mit variabler Adresse], UART) programmiert werden und gibt auch hierüber die Zeitdaten aus. Damit haben wir eine autark arbeitende RTC, die dank Funkuhrempfang stets die genaueste Zeit samt Datum, Sommerzeitumschaltung, Kalender etc. ausgeben kann.

Technische Daten

Gerätekurzbezeichnung:	RTC-DCF
Versorgungsspannung:	1,8–3,6 V
Stromaufnahme:	12 mA max.
Schnittstellen:	I²C, SPI oder UART
Umgebungstemperatur:	-10 bis +55 °C
Abmessungen (B x H x T):	
Arduino-Board:	53 x 60 x 6,5 mm (ohne Stiftleisten)
Breakout-Board:	27 x 34 x 6,5 mm (ohne Stiftleisten)
Gewicht:	30 g (inkl. DCF-Modul)
I²C	
Maximale Taktrate:	400 kHz
SPI	
Maximale Taktrate:	500 kHz
Clock-Polarität:	Idle Low
Clock-Phase:	fallende Flanke (2. Flanke)
UART	
Baudraten (in Baud):	4800, 9600, 14.400, 19.200, 28.800, 38.400, 57.600, 115.200

Daneben kann man per Interruptsteuerung einen einstellbaren Alarm auslösen, etwa eine Weck- oder Schaltfunktion. Außerdem ist die Baugruppe als periodische Interrupt-Quelle einsetzbar, dies entweder als Pulsfrequenz oder bei Eintreten einer zeitlichen Bedingung – ein Feature, das in vielen Mikrocontrollersystemen benötigt wird. Auch bei korrektem Empfang eines DCF77-Zeitsignals kann ein Interrupt ausgegeben werden.

Die Platine der DCF77-gesteuerten Echtzeituhr ist so ausgeführt, dass die kleine Baugruppe als Breakout-Board in eine eigene Applikation eingebunden werden kann, aber, im Originalformat belassen, auch als Arduino-Shield einsetzbar ist. Für letztere Anwendung steht eine Library zur einfachen Programmierung via Arduino bereit.

Schaltung

Das Schaltbild des RTC-DCF-Moduls ist in Bild 1 dargestellt. Wesentlicher Bestandteil der kleinen Schaltung ist der bereits beschriebene Mikrocontroller IC1, welcher über die Schnittstellen I²C, SPI und UART angesprochen werden kann, die über die Stiftleisten ST5 und ST6 nach außen geführt sind.

Der Uhrenquarz Q1 mit den Lastkapazitäten C8 und C9 fungiert als Taktquelle für die im Mikrocontroller laufende Real-Time-Clock. Die Kondensatoren C2 bis C5 dienen der Stabilisierung und Glättung der Versorgungsspannung des Mikrocontrollers. Über den 8fach-

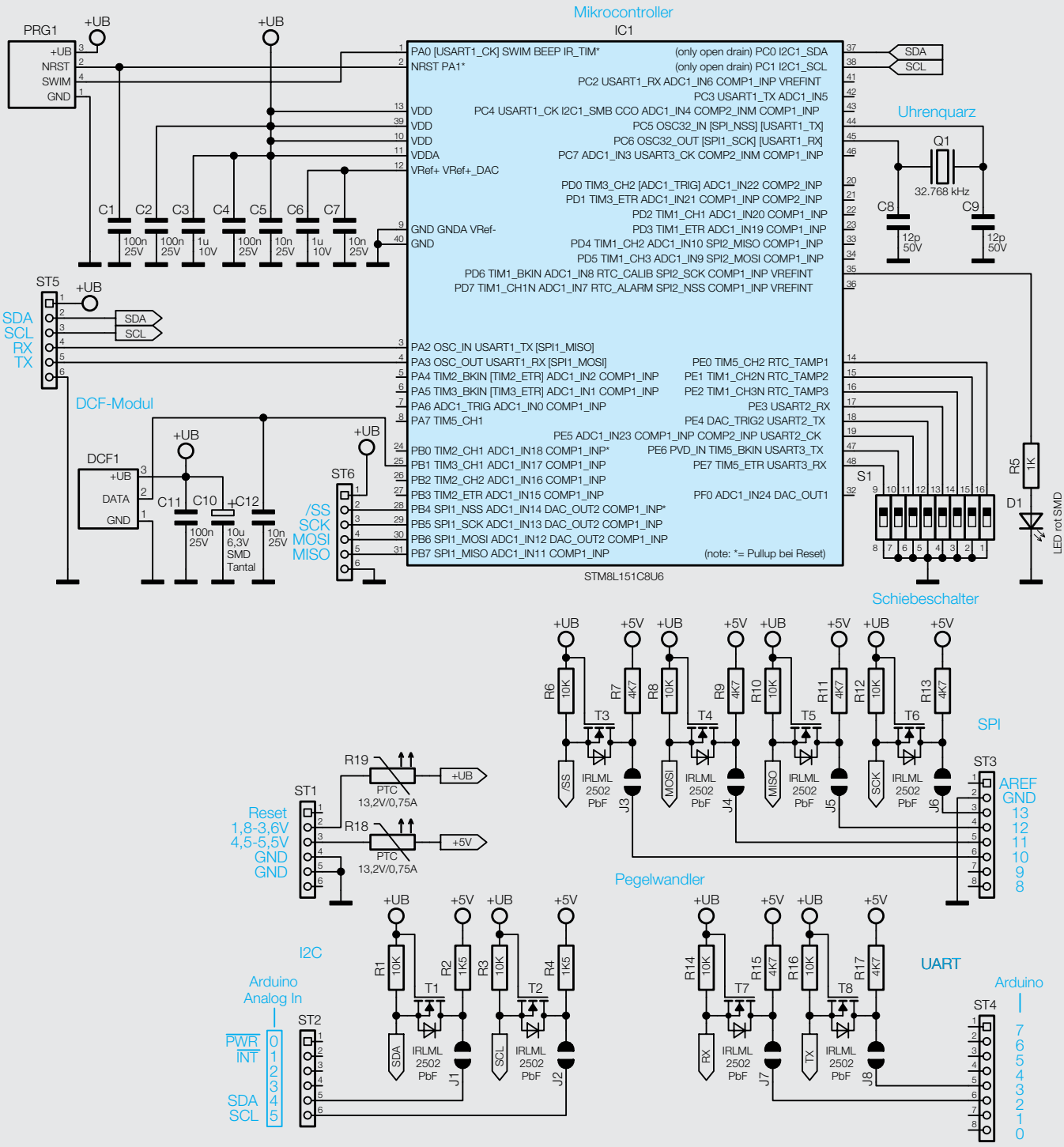


Bild 1: Schaltbild des RTC-DCF-Moduls

Schiebeschalter S1 erfolgt die Auswahl der Kommunikationsschnittstelle und die Einstellung der Baudrate bei Kommunikation über UART bzw. der Adresse bei Kommunikation über I²C.

Über die rote LED D1 mit deren Vorwiderstand R5 lässt sich bei Bedarf das Eintreffen eines Interrupts auch visuell anzeigen. Zum Empfang der aktuellen Uhrzeit ist zusätzlich das DCF-Modul DCF1 in die Schaltung integriert, welches zusätzlich mit den Kondensatoren C10 und C11 gegen Spannungsschwankungen geschützt ist.

Zur Nutzung des RTC-DCF als Arduino-Shield ist die Platine entsprechend an die Größe des Arduino-Boards angepasst und wird mit diesem über die Stiftleisten ST1 bis ST4 verbunden. Da der Mikrocontroller IC1 mit einer maximalen Spannung von 3,6 V arbeitet, das Arduino-Board jedoch mit 5 V betrieben wird, können die Datenleitungen der beiden Controller nicht direkt miteinander verbunden werden. Daher ist zwischen jeder Datenleitung ein bidirektionaler Pegelwandler bestehend aus einem MOSFET und zwei Pull-up-Widerständen verbaut.

Tabelle 1

Einstellungsmöglichkeiten des 8fach-Schiebeschalters S1

Schalter								Funktion
8	7	6	5	4	3	2	1	
0	0	0	x	x	x	0	0	UART-Kommunikation mit 4800 Baud
0	0	1	x	x	x	0	0	UART-Kommunikation mit 9600 Baud
0	1	0	x	x	x	0	0	UART-Kommunikation mit 14.400 Baud
0	1	1	x	x	x	0	0	UART-Kommunikation mit 19.200 Baud
1	0	0	x	x	x	0	0	UART-Kommunikation mit 28.800 Baud
1	0	1	x	x	x	0	0	UART-Kommunikation mit 38.400 Baud
1	1	0	x	x	x	0	0	UART-Kommunikation mit 57.600 Baud
1	1	1	x	x	x	0	0	UART-Kommunikation mit 115.200 Baud
x	x	x	x	x	x	0	1	SPI-Kommunikation
a	b	c	d	e	f	1	0	I²C-Kommunikation mit der I²C-Adresse *a-b-c-d-e-f-1*
x	x	x	x	x	x	1	1	keine Funktion (Bootloader)

(0: OFF, 1: ON, x: egal)

Anschlüsse

Die Kommunikation mit dem RTC-DCF kann wahlweise über die Schnittstellen I²C, SPI oder UART erfolgen. Die Einstellung der Kommunikationsschnittstelle erfolgt über die Schalter 1 und 2 des 8fach-Schiebeschalters S1. Die weiteren sechs Schalter dienen zur Einstellung der Baudrate bei Kommunikation über UART bzw. der Adresse bei Kommunikation über I²C. Die verschiedenen Einstellungsmöglichkeiten sind in Tabelle 1 zusammengefasst.

Die acht Signalanschlüsse des RTC-DCF erhalten je nach gewählter Kommunikationsschnittstelle eine unterschiedliche Funktion. Für die Kommunikation über I²C werden die beiden Anschlüsse SDA und SCL, für die Kommunikation über UART die beiden Anschlüsse RX und TX und für die Kommunikation über SPI die Anschlüsse /SS, MOSI, MISO und SCK verwendet. Die jeweils nicht belegten Anschlüsse können als Interruptausgänge für einen Alarm, einen periodischen Interrupt und für die Signalisierung eines empfangenen DCF77-Zeitsignals genutzt werden (siehe Abschnitt „Register“). Die entsprechende Pinbelegung der Interrupts bei den unterschiedlichen Schnittstellen ist in Tabelle 2 dargestellt.

Tabelle 2

Pinbelegung der drei möglichen Interruptquellen

Interrupt	Schnittstelle					
	UART		SPI		I²C	
	RTC-DCF	Arduino	RTC-DCF	Arduino		
Alarm-Interrupt	MOSI	11	TX	0	MOSI	11
Periodischer Interrupt	MISO	12	SDA	18	MISO	12
DCF77-Empfang-Interrupt	/SS	10	SCL	19	/SS	10

Register

Alle schreib- und lesbaren Daten des RTC-DCF sind in insgesamt 16 Registern untergebracht. Tabelle 3 zeigt alle Register mit Adresse, Name und Bedeutung der einzelnen Bits. Da eine ausführliche Beschreibung den Rahmen dieses Artikels sprengen würde, wird an dieser Stelle nur kurz auf die Register eingegangen. Eine ausführlichere Be-

Tabelle 3

Registertabelle

Adresse	Name	Bits							
		D7	D6	D5	D4	D3	D2	D1	D0
0h	Sekunde	–	ST[2:0]			SU[3:0]			
1h	Minute	–	MNT[2:0]			MNU[3:0]			
2h	Stunde	–	–	HT[1:0]		HU[3:0]			
3h	Wochentag	–	–	–	–	–	WU[2:0]		
4h	Tag	–	–	DT[1:0]		DU[3:0]			
5h	Monat	–	–	–	MT	MU[3:0]			
6h	Jahr	YT[3:0]				YU3:0]			
7h	Alarm Minute	–	AMNT[2:0]			AMNU[3:0]			
8h	Alarm Stunde	–	–	AHT[1:0]		AHU[3:0]			
9h	Alarm Wochentag	–	AWSU	AWSA	AWFR	AWTH	AWWE	AWTU	AWMO
Ah	periodischer Interrupt	–	–	–	–	–	PIM[2:0]		
Bh	Alarm–Config–Register	–	–	–	–	–	AILED	AIE	–
Ch	Per.–Int.–Config–Register	–	–	–	–	–	PILED	PIE	–
Dh	DCF77–Config–Register	–	–	–	–	–	DCFLED	DCFIE	DCFE
Eh	Status–Register	–	–	–	–	–	AIF	PIF	DCFIF
Fh	–	–	–	–	–	–	–	–	–

schreibung steht online zum Download zur Verfügung.

Uhrzeit/Datum

Die Register 0x0 bis 0x6 enthalten die aktuelle Uhrzeit bzw. das aktuelle Datum. Beim Schreiben dieser Register erfolgt die Übernahme der Daten nach Beendigung der jeweiligen Kommunikation (siehe Abschnitt „Kommunikation"). Beim Lesen dieser Register werden diese beim Start der jeweiligen Kommunikation eingefroren und erst nach Ende der Kommunikation wieder aktualisiert. Dieses Vorgehen verhindert inkonsistente Daten durch einen Sekundenwechsel während der Kommunikation.

Alle Daten der Uhrzeit bzw. des Datums (mit Ausnahme des Wochentags) werden im BCD-Format verwaltet und ausgegeben. Beim Schreiben dieser Register sind die neuen Werte ebenfalls im BCD-Format zu übergeben. Die Codierung des Wochentags erfolgt im Register 0x3 innerhalb der unteren drei Bits, wobei 0x01 dem Montag und 0x07 dem Sonntag entspricht.

Alarm

Mit Hilfe der Register 0x7 bis 0x9 und dem Config-Register 0xB kann der RTC-DCF-Baustein dafür verwendet werden, einen Interrupt zu einer bestimmten Uhrzeit eines Wochentages auszulösen. Die Alarm-Minute (0x7) und die Alarm-Stunde (0x8) werden dabei im BCD-Format angegeben. Innerhalb des Registers Alarm-Wochentag (0x9) ist jedem Wochentag ein Bit zugeordnet, so dass durch Setzen der entsprechenden Bits jede beliebige Wochentagskombination ermöglicht wird. Durch Setzen des AIE-Bits im Alarm-Config-Register (0xB) wird der Alarm-Interrupt aktiviert. Soll die LED auf dem RTC-DCF-Modul zusätzlich bei Erreichen einer Alarmzeit eingeschaltet werden, muss hierfür das AILED-Bit gesetzt werden.

Nach Eintreten eines Alarm-Interrupts wechselt das AIF-Bit innerhalb des Status-Registers (0xE) auf 1. Durch Beschreiben dieses Bits mit einer 1 wird der Alarm-Interrupt zurückgesetzt und das AIF-Bit wechselt auf 0. Ein Beschreiben des Bits mit einer 0 hat keine Auswirkungen.

Periodischer Interrupt

Mit Hilfe des Registers 0xA und dem Config-Register 0xC kann der RTC-DCF als periodische Interrupt-Quelle verwendet werden. Dazu wird der gewünschte Modus in das Register 0xA geschrieben. Mögliche Modi sind dabei ein periodischer Interrupt mit 1 Hz oder 2 Hz (Pulsmodus) oder jeweils zu Beginn einer Sekunde, einer Minute, einer Stunde, eines Tages oder eines Monats (Levelmodus). Durch Setzen des PIE-Bits im Per.-Int.-Config-Register (0xC) wird der periodische Interrupt aktiviert. Soll die LED auf dem RTC-DCF-Modul zusätzlich bei Auftreten eines periodischen Interrupts eingeschaltet werden, muss hierfür das PILED-Bit gesetzt werden.

Nach Eintreten eines periodischen Interrupts wechselt das PIF-Bit innerhalb des Status-Registers (0xE) auf 1. Durch Beschreiben dieses Bits mit einer 1 wird der periodische Interrupt zurückgesetzt und das PIF-Bit wechselt auf 0. Ein Beschreiben des Bits mit einer 0 hat keine Auswirkungen.

Bei den periodischen Interrupts wird zwischen Puls- und Levelmodus unterschieden (siehe oben). Beim Pulsmodus wird die Frequenz (1 Hz oder 2 Hz) automatisch mit einem Tastgrad von 50 % ausgegeben. Beim Levelmodus wird der Interrupt bei Erfüllung der entsprechenden Bedingung ausgelöst, verbleibt aber in diesem Zustand, bis das PIF-Bit im Statusregister (0xE) zurückgesetzt wird.

DCF77-Empfang

Mit Hilfe des Config-Registers 0xD kann die DCF77-Funktionalität des RTC-DCF eingestellt werden. Dazu kann mit dem DCFE-Bit der DCF77-Empfang aktiviert werden. Wird daraufhin ein DCF77-Zeitsignal komplett empfangen, wird die Uhrzeit und das Datum innerhalb des RTC-DCF automatisch auf die empfangene Zeit gesetzt. Zusätzlich kann durch Setzen des DCFIE-Bits ein Interrupt ausgelöst werden, sobald ein DCF77-Zeitsignal komplett ausgewertet werden konnte. Soll die LED auf dem RTC-DCF-Modul zusätzlich bei Empfang eines DCF77-Zeitsignals eingeschaltet werden, muss hierfür das DCFLED-Bit gesetzt werden.

Nach Eintreten eines DCF77-Empfang-Interrupts wechselt das DCFIF-Bit innerhalb des Status-Registers (0xE) auf 1. Durch Beschreiben dieses Bits mit einer 1 wird der DCF77-Empfang-Interrupt zurückgesetzt und das DCFIF-Bit wechselt auf 0. Ein Beschreiben des Bits mit einer 0 hat keine Auswirkungen.

Kommunikation

Je nach gewählter Kommunikationsschnittstelle (siehe Abschnitt „Anschlüsse") unterscheidet sich das verwendete Datenprotokoll, welches zur Kommunikation mit dem RTC-DCF benötigt wird.

UART

Die Kommunikation über die UART-Schnittstelle wird jeweils mit einem Startbyte (STX: 0x02) begonnen und mit einem Endbyte (ETX: 0x03) abgeschlossen. Müssen Daten mit dem Wert 0x02 (STX), 0x03 (ETX) oder 0x10 (DLE) gesendet werden, sind diese über 2 Bytes zu codieren (byte stuffing). Dazu wird zunächst das Byte 0x10 (DLE) gesendet, welches signalisiert, dass das folgende Byte speziell zu behandeln ist. Anschließend wird das gewünschte Datenbyte mit gesetztem MSB gesendet.

Beispielsweise ergeben sich somit für ein Datenbyte 0x02 die zu sendenden Bytes 0x10 und 0x82. Um zu verhindern, dass bei der Adressangabe bereits für das Minuten- und Stundenregister (0x02 und 0x03) das Byte-Stuffing-Verfahren angewendet werden muss, wird bei jeder Adressangabe zusätzlich das 6. Bit (0x20) gesetzt, so dass sich die Adressen in den Bereich 0x20 bis 0x2F verschieben.

Das Schreiben der Register über UART erfolgt nach dem nachfolgenden Muster. Die Adresse gibt das Register an, an welches das 1. Datenbyte geschrieben werden soll. Das RTC-DCF inkrementiert nach jedem Datenbyte die interne Registeradresse um 1, so dass das folgende Datenbyte entsprechend an die folgende Adresse geschrieben wird. Die Adressen werden dabei als Ring verwaltet, so dass nach Beschreiben des Registers 0xF die interne Registeradresse auf 0x0 zurückgesetzt wird.

Start (1 Byte)	Datenrichtung „Schreiben" (1 Byte)	Registeradresse (1 Byte)	Daten (n Byte)	Ende (1 Byte)
0x02	0x00	0x20 … 0x2F	0x00 … 0xFF	0x03

Das Lesen der Register über UART erfolgt entsprechend dem nachfolgenden Muster. Die Adresse gibt das Register an, ab dem mit dem Lesen begonnen werden soll. Das folgende Byte gibt die Anzahl der zu lesenden Bytes an, wobei ein Lesen über die Grenze 0xF → 0x0 erlaubt ist.

Start (1 Byte)	Datenrichtung „Lesen" (1 Byte)	Registeradresse (1 Byte)	Anzahl Datenbytes (1 Byte)	Ende (1 Byte)
0x02	0x01	0x20 … 0x2F	0x01 … 0x10	0x03

Auf einen Lesebefehl antwortet das RTC-DCF entsprechend dem nachfolgenden Muster, wobei die Anzahl der Datenbytes den vorher angeforderten Daten, beginnend bei der gesendeten Adresse, entspricht. Das RTC-DCF verwendet bei seiner Antwort ebenfalls das Byte-Stuffing-Verfahren bei den Datenbytes 0x02, 0x03 und 0x10 (siehe oben).

Start (1 Byte)	Daten (n Byte)	Ende (1 Byte)
0x02	0x00 ... 0xFF	0x03

SPI

Die Kommunikation über die SPI-Schnittstelle wird durch eine fallende Flanke an /SS gestartet und entsprechend durch eine steigende Flanke an /SS beendet. Bei fallender Flanke werden alle Register des RTC-DCF eingefroren, so dass beim Auslesen stets konsistente Daten zur Verfügung stehen. Ebenso werden beim Schreiben der Register die neuen Werte erst nach der steigenden Flanke an /SS übernommen. Auf das Fortlaufen der internen Uhr hat der /SS-Pin indes keinen Einfluss.

Zur Kommunikation mit dem RTC-DCF wird ein Clock-Signal benötigt, welches im Leerlauf auf Low-Potential liegt (Polarität Idle Low) und bei dem die Datenleitungen auf die fallende bzw. 2. Flanke (Phase) abgetastet werden.

In Bild 2 sind die vier verschiedenen Datentransfer-Formate über SPI angegeben.

Mit Hilfe des Single-Write-Formats kann ein einzelnes Register des RTC-DCF beschrieben werden. Dazu wird im ersten vom Master gesendeten Byte das High-Nibble auf 8h gesetzt. Das Low-Nibble enthält die zu schreibende Registeradresse (siehe Tabelle 3). Das folgende Byte enthält die neuen Daten für die vorher angegebene Registeradresse. Wird nun ein weiteres Byte vom Master gesendet, wird dieses wieder als Angabe des Formats und der Registeradresse interpretiert, das darauffolgende Byte entsprechend wieder als neue Daten für die zuvor übermittelte Registeradresse. Das Single-Write-Format eignet sich besonders für das Schreiben von Registern, die nicht aufeinanderfolgend angeordnet sind. Alle geschriebenen Daten werden nach Beendigung der Kommunikation durch eine steigende Flanke an /SS übernommen.

Im Vergleich dazu ist es mit dem Burst-Write-Format möglich, aufeinanderfolgende Register direkt zu beschreiben, ohne dass nach jedem Datenbyte die Adresse neu übermittelt werden muss. Dazu wird im ersten vom Master gesendeten Byte das High-Nibble auf 0x0 gesetzt. Das Low-Nibble enthält wiederum die zu schreibende Registeradresse (siehe Tabelle 3). Alle folgenden Bytes enthalten die neuen Daten ab der vorher angegebenen Registeradresse, wobei das RTC-DCF nach jedem Datenbyte die Adresse automatisch um 1 erhöht. Dabei folgt nach der Adresse 0xF die Adresse 0x0, so dass ein ringförmiges Schreiben möglich ist. Alle geschriebenen Daten werden nach Beendigung der Kommunikation durch eine steigende Flanke an NSS übernommen.

Mit Hilfe des Single-Read-Formats kann ein einzelnes Register des RTC-DCF gelesen werden. Dazu wird im ersten vom Master gesendeten Byte das High-Nibble auf 0xC gesetzt. Das Low-Nibble enthält die zu lesende Registeradresse (siehe Tabelle 3). Mit den folgenden 8 Takten an SCK sendet das RTC-DCF nun die Daten des vorher angegebenen Registers an den Master. Wird nun ein weiteres Byte vom Master gesendet, wird dieses wieder als Angabe des Formats und der Registeradresse interpretiert. Mit den darauffolgenden 8 Takten an SCK sendet das RTC-DCF wiederum die Daten der zuvor gesendeten Registeradresse an den Master. Das Single-Write-Format eignet sich besonders für das Lesen von Registern, die nicht aufeinanderfolgend angeordnet sind.

Im Vergleich dazu ist es mit dem Burst-Read-Format möglich, aufeinanderfolgende Register direkt zu lesen, ohne dass nach jedem Datenbyte die Adresse neu übermittelt werden muss. Dazu wird im ersten vom Master gesendeten Byte das High-Nibble auf 0x4 gesetzt. Das Low-Nibble enthält wiederum die zu lesende Registeradresse (siehe Tabelle 3). Daraufhin sendet das RTC-DCF entsprechend den Takten an SCK die Daten ab der vorher angegebenen Registeradresse, wobei sich nach jedem Datenbyte die Adresse automatisch um 1 erhöht. Dabei folgt nach der Adresse 0xF die Adresse 0x0, so dass ein ringförmiges Lesen möglich ist.

Während der Kommunikation mit dem RTC-DCF über SPI muss zwischen jedem übertragenen Byte eine Pause von min. 50 µs eingehalten werden. Des Weiteren benötigt der RTC-DCF nach einer steigenden Flanke an /SS min. 300 µs zur Übernahme der neuen Daten. In dieser Zeit darf keine Kommunikation zum Gerät stattfinden.

I²C

Der Ablauf der Kommunikation über I²C ist in Bild 3 schematisch dargestellt. Dabei sendet der Master nach der Startbedingung zunächst die I²C-Adresse des angeschlossenen RTC-DCF. Die oberen 6 Bit ergeben sich durch die Einstellung des Schiebeschalters S1 (siehe Abschnitt „Anschlüsse“). Das niederwertigste Bit der I²C-Adresse ist fest auf 1 gesetzt. Zusätzlich wird mit der I²C-Adresse die Richtung des nachfolgenden Datenstroms eingestellt. Eine 0 entspricht dabei einem Schreibvorgang, eine 1 entsprechend einem Lesevorgang. Beim Schreiben folgt nach der I²C-Adresse die Adresse des zu schreibenden Registers. Alle folgenden Bytes enthalten die neuen Daten ab der vorher angegebenen Registeradresse, wobei das RTC-DCF nach jedem Datenbyte die Adresse automatisch um 1 erhöht. Dabei folgt nach der Adresse 0xF die Adresse 0x0, so dass ein ringförmiges Schreiben möglich ist.

Alle geschriebenen Daten werden nach Beendigung der Kommunikation durch eine Stoppbedingung übernommen.

Wird die Richtung des Datenstroms auf Lesen gesetzt, beginnt das RTC-DCF mit dem folgenden Byte mit dem Senden der Registerdaten, wobei der Lesevorgang mit dem Register beginnt, auf dem der interne Adresszeiger derzeit positioniert ist. Nach jedem gesendeten Byte erhöht das RTC-DCF die Registeradresse automatisch um 1, so dass das jeweils nächste gesendete Byte die Daten des nächsten Registers enthält. Dabei folgt nach der Adresse 0xF die Adresse 0x0, so dass ein ringförmiges Lesen möglich ist.

Damit der Lesevorgang bei einer bestimmten Registeradresse begonnen wird, ist vorher ein Schreibvorgang mit der entsprechenden Registeradresse durchzuführen, wobei die Kommunikation nach Senden der Adresse durch eine wiederholte Startbedingung unterbrochen und in den Lesevorgang übergegangen wird (siehe Bild 3).

Arduino

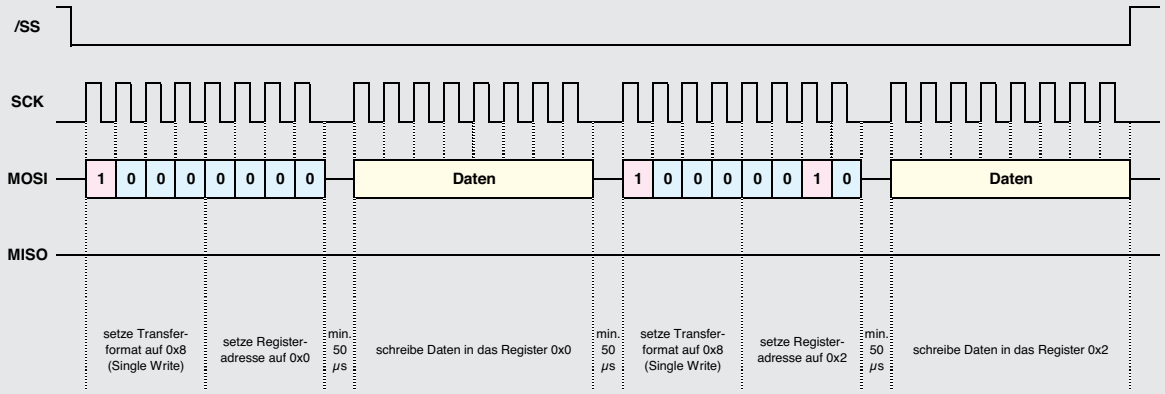
Für die Verwendung des RTC-DCF mit dem Arduino-Board steht eine Library mit allen Funktionen des Moduls online zum Download zur Verfügung.

Bevor das RTC-DCF mit dem Arduino-Board verwendet werden kann, sind die Lötjumper J1 bis J8 entsprechend dem gewählten Kommunikationsweg mit Lötzinn zu schließen.

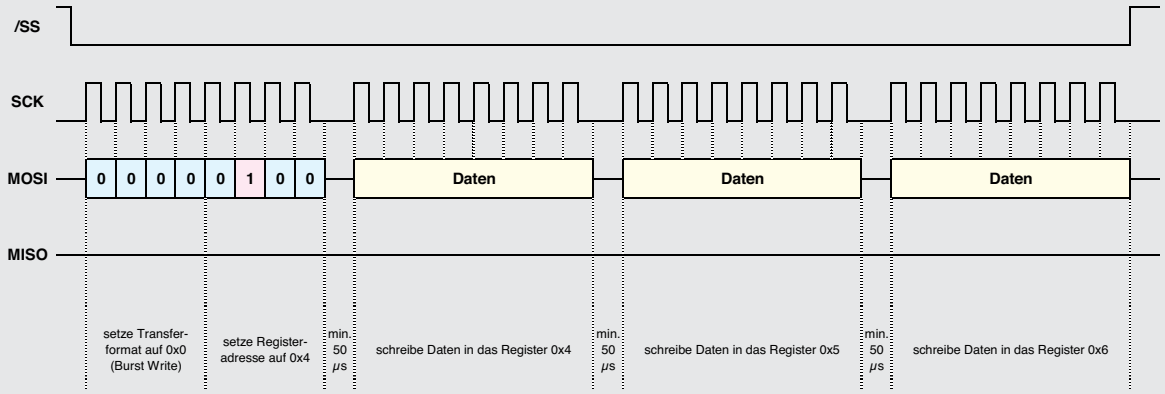
Nachbau

Da alle SMD-Komponenten beim RTC-DCF-Modul bereits vorbestückt sind, müssen nur noch wenige Bauteile

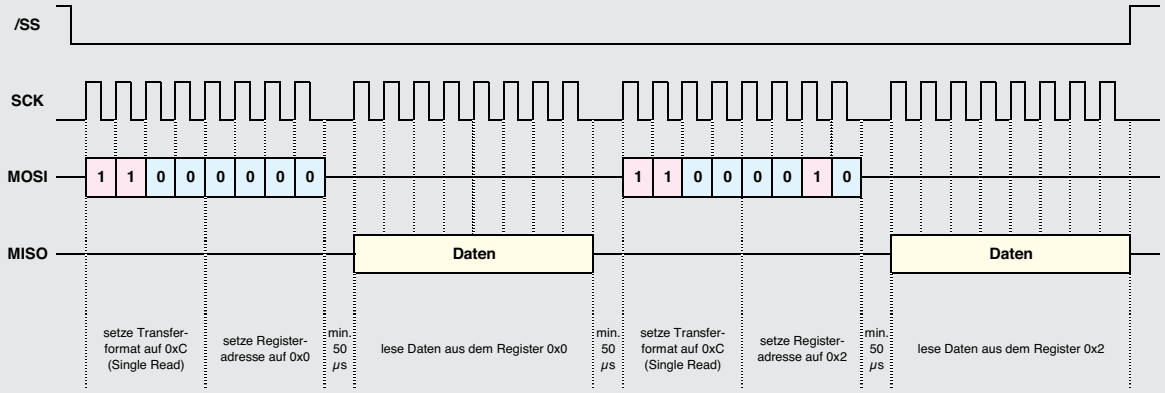
(1) Single Write



(2) Burst Write



(3) Single Read



(4) Burst Read

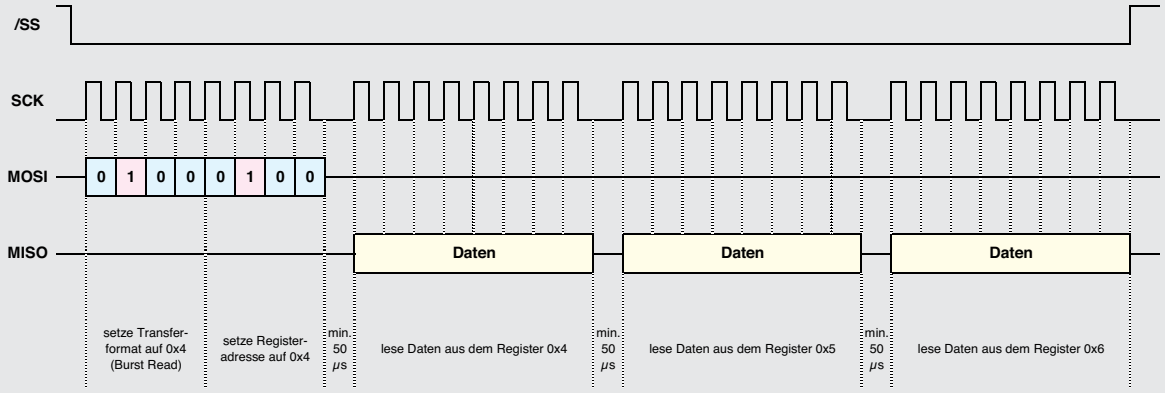
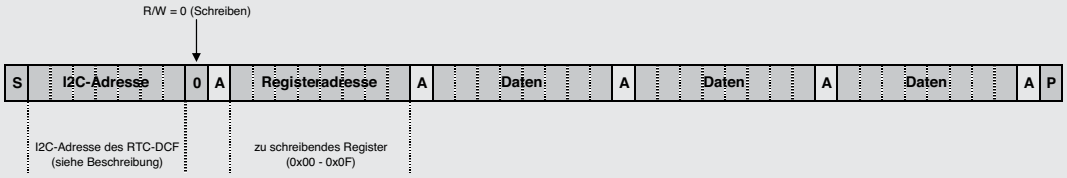
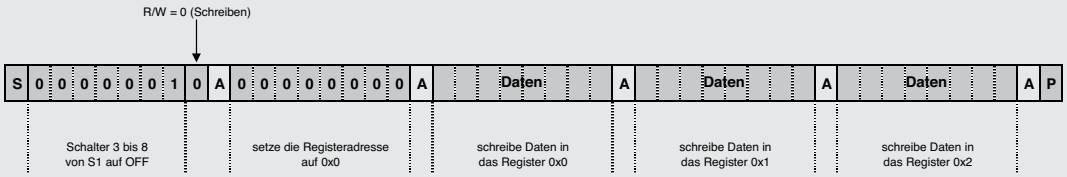


Bild 2: Die Kommunikation über die SPI-Schnittstelle

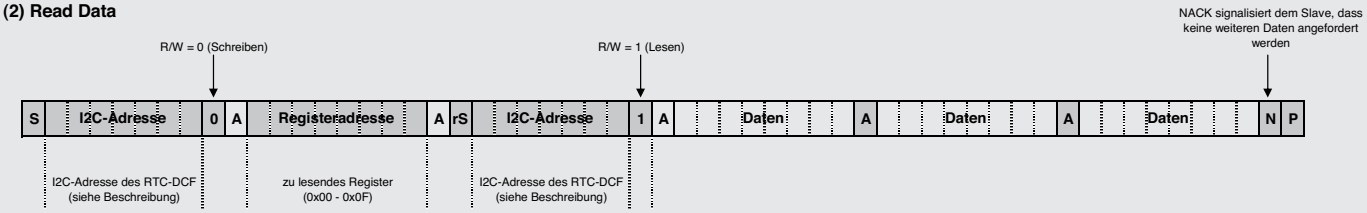
(1) Write Data



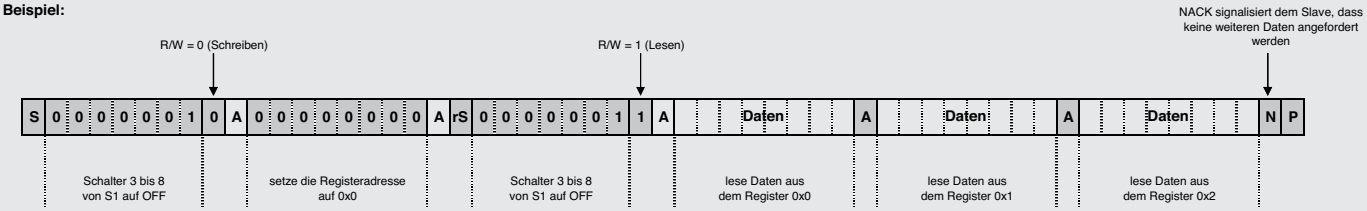
Beispiel:



(2) Read Data



Beispiel:



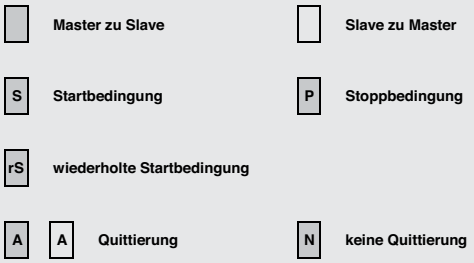


Bild 3: Die Kommunikation über die I²C-Schnittstelle

Stückliste

Widerstände:

1 kΩ/SMD/0603	R5
1,5 kΩ/SMD/0603	R2, R4
4,7 kΩ/SMD/0603	R7, R9, R11, R13, R15, R17
10 kΩ/SMD/0603	R1, R3, R6, R8, R10, R12, R14, R16
Polyswitch/13,2 V/0,75 A/SMD/1812	R18, R19

Kondensatoren:

12 pF/SMD/0603	C8, C9
10 nF/SMD/0603	C5, C7, C12
100 nF/SMD/0603	C1, C2, C4, C11
1 µF/SMD/0603	C3, C6
10 µF/6,3 V/Tantal/SMD	C10

Halbleiter:

ELV121170/SMD	IC1
IRLML2502PbF/SMD	T1–T8
LED/rot/SMD/PLCC-2 Gehäuse	D1

Sonstiges:

Fertiggerät DCF-2	DCF1
Quarz, 32,768 kHz, ±20 ppm	Q1
Buchsenleisten, 1x 6-polig, gerade, print	ST1, ST2
Buchsenleisten, 1x 8-polig, gerade, print	ST3, ST4
Stiftleisten, 1x 6-polig, gerade, print	ST5, ST6
Mini-DIP-Schalter, 8-polig, liegend, SMD	S1
2 Kabelbinder, 90 x 2,5 mm	

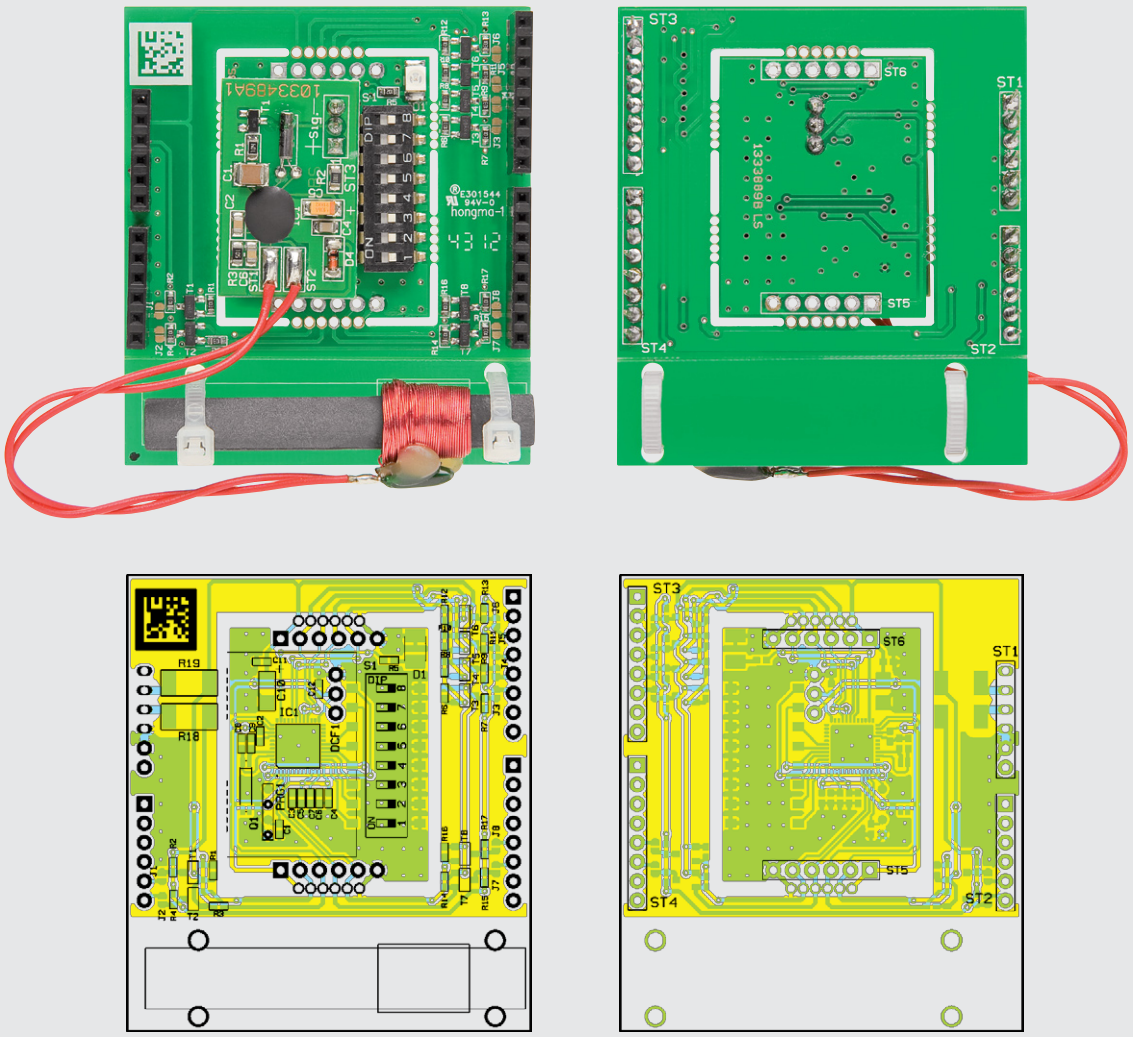


Bild 4: Die komplett bestückte Platine des RTC-DCF mit zugehörigen Bestückungsplänen, links Oberseite, rechts Unterseite

bestückt werden. Als Orientierung dienen dabei das Platinenfoto sowie der zugehörige Bestückungsdruck (Bild 4).

Soll die Platine als Arduino-Shield dienen, sind zunächst die Buchsenleisten ST1 bis ST4 zu montieren, wobei diese so einzusetzen sind, dass die Stifte auf der nicht bestückten Seite der Platine liegen. Anschließend werden die Buchsenleisten von der Unterseite verlötet (Bild 5).

Soll die Platine hingegen in einer eigenen Schaltung betrieben werden, ist diese zunächst vorsichtig aus dem Rahmen zu entfernen. Danach werden die beiden Stiftleisten ST5 und ST6 so montiert, dass die Stifte entsprechend nach unten zeigen (Bild 6).

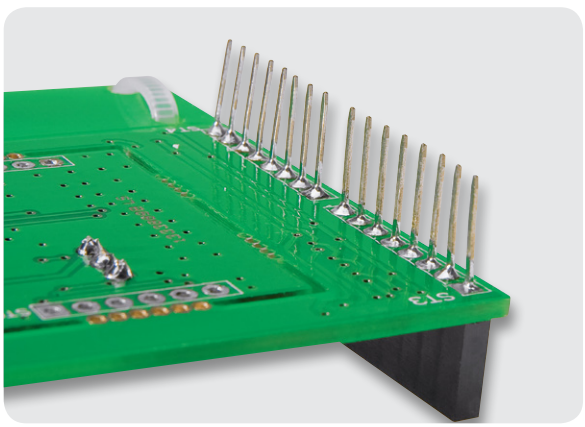


Bild 5: Die Stiftleisten der Arduino-Verbinder werden auf der Platinenunterseite verlötet

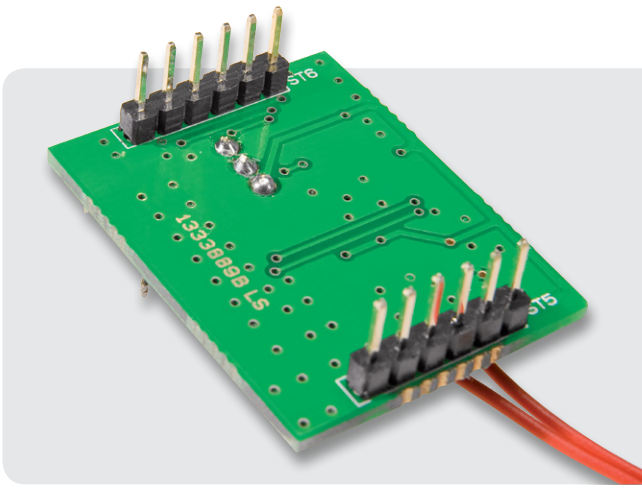


Bild 6: Wird die Baugruppe als Breakout-Board eingesetzt, sind die Stiftleisten von unten einzusetzen und auf der Oberseite zu verlöten.

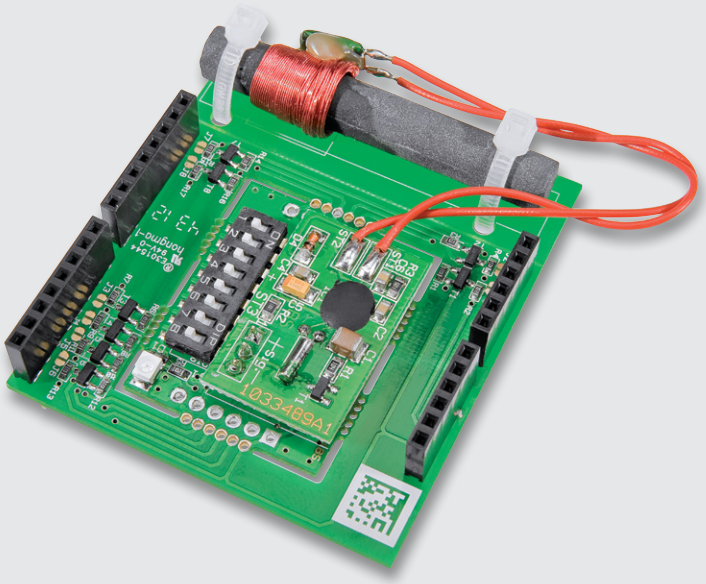


Bild 7: So wird das DCF-Modul aufgesetzt und verlötet.

Im nächsten Schritt folgt der Einbau des DCF-Moduls, welches mit seinen drei Stiften von oben in die entsprechenden Bohrungen des RTC-DCF eingesetzt und von der Unterseite verlötet wird (Bild 7).

Bei der Nutzung als Arduino-Shield wird die Ferrit-Antenne abschließend mit den beiden Kabelbindern auf dem freistehenden Bereich der Platine fixiert (Bild 8). Bei Verwendung der Platine ohne Rahmen ist die Antenne innerhalb der eigenen Schaltung an einer geeigneten Stelle zu fixieren.

Bild 9 zeigt die fertig aufgebaute Baugruppe, aufgesteckt auf ein Arduino-Board.

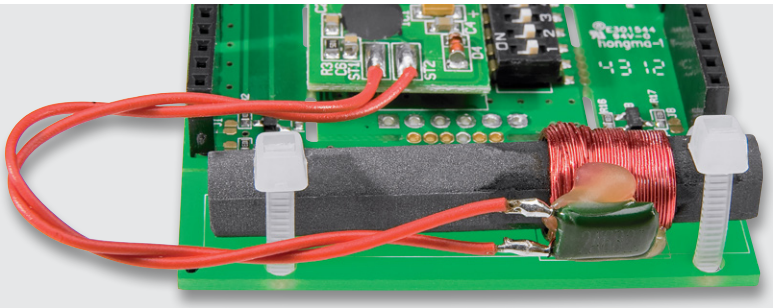


Bild 8: Die Ferritantenne des DCF77-Empfängers wird mit Kabelbindern auf der Arduino-Platine befestigt.

Hinweis zum Betrieb

Für den Betrieb des RTC-DCF sind folgende wichtige Hinweise zu beachten:

- Das Modul ist zum Schutz vor elektrostatischen Entladungen, wie sie bei Berührung durch Personen häufig auftreten, in ein geschlossenes Gehäuse einzubauen.
- Auch die Signal-LED darf nicht berührbar sein. Soll die LED außerhalb des Gehäuses sichtbar sein, muss ein transparentes Gehäuse, ein transparentes Gehäusefenster oder ein Lichtleiter verwendet werden.
- Die Versorgungsspannung von max. 3,6 V darf in keinem Fall überschritten werden.
- Wird das Breakout-Board verwendet, ist dieses direkt in eine Schaltung ohne zusätzliche Leitungen einzubauen.

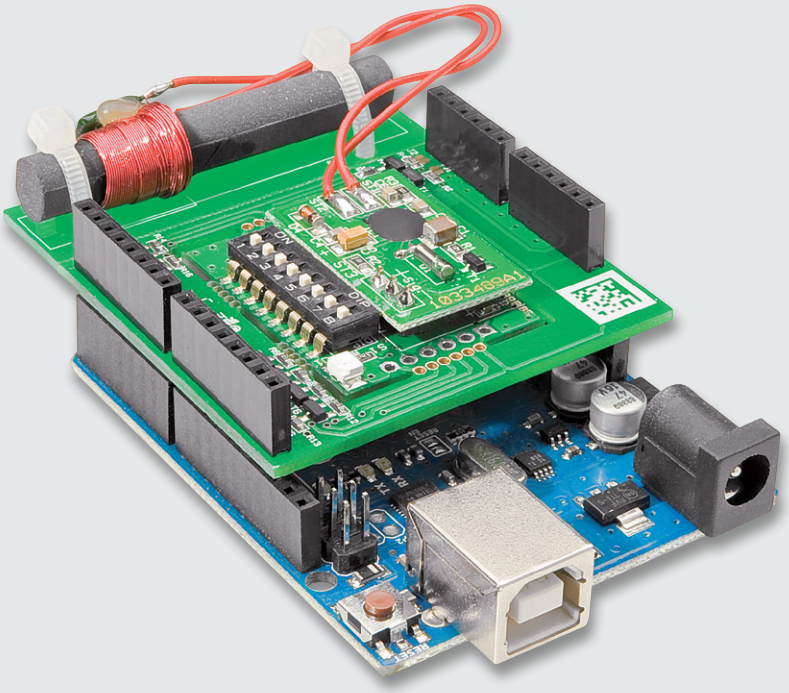


Bild 9: Das RTC-DCF-Modul in Aktion, aufgesteckt auf ein Arduino-Board.

Wichtiger Hinweis:

Zur Gewährleistung der elektrischen Sicherheit muss es sich bei der speisenden Quelle um eine Sicherheits-Schutzkleinspannung handeln.
Außerdem muss es sich um eine Quelle begrenzter Leistung gemäß EN60950-1 handeln, die nicht mehr als 15 W liefern kann.
Üblicherweise werden beide Forderungen von handelsüblichen Steckernetzteilen mit entsprechender Leistung erfüllt.