Willkommen!

Und herzlichen Dank für den Kauf unseres AZ-Delivery DatenLogger Moduls für den Arduino. Auf den folgenden Seiten gehen wir mit dir gemeinsam die ersten Schritte von der Einrichtung auf dem Arduino durch.
Viel Spaß!

Dieses Datenlogger Modul ist für FAT16/FAT32 formatierte SD-Karten geeignet. Die integrierte 3,3V Level-Shifter-Schaltung schützt die SD-Karte vor Beschädigungen. Mit der Echtzeituhr läuft die Zeit selbst dann weiter, auch wenn der Arduino nicht in Betrieb ist.

Verdrahten des Moduls mit einem Arduino Uno:

Die Verdrahtung ist relativ einfach, wir stecken das Modul auf einen der Kompatiblen Arduinos. Kompatibel sind:

Arduino UNO
Duemilanove
Diecimila
Leonardo
ADK / Mega R3

Oder ähnliche Arduinos

Nachdem das Shield aufgesteckt ist kann der Arduino mit Spannung versorgt werden.

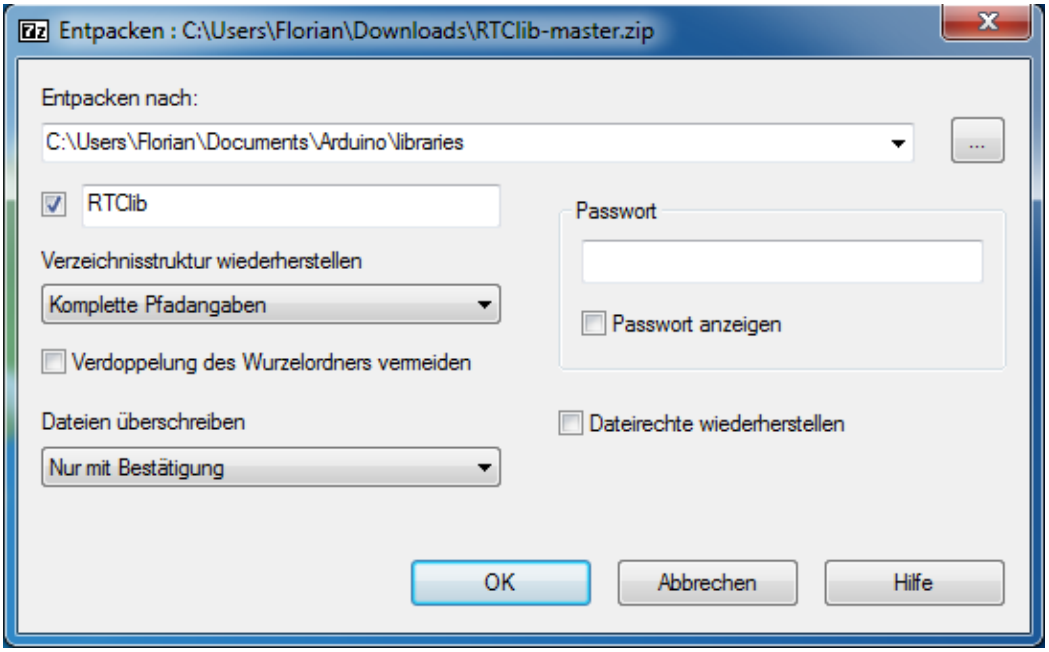
„Programmieren“ des Arduino:

Bevor wir mit dem Programmieren beginnen können, müssen wir zuerst von git die entsprechenden Bibliotheken herunterladen und installieren.

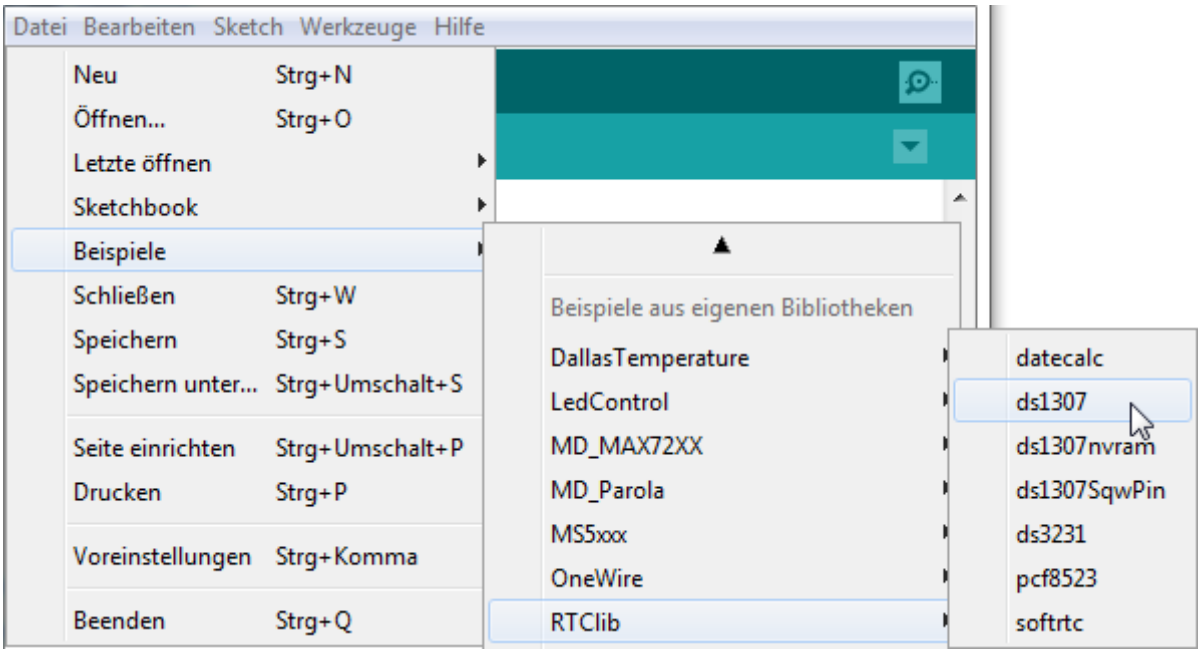
Laden wir die Ressourcen herunter:
https://github.com/adafruit/RTClib/archive/master.zip

Diese Zip Datei entpacken (mit 7zip) wir in den Ordner: [Eigenes Userverzeichnis (C:\Benutzer\Florian\) \ Eigene Dokumente \ Arduino \ libraries \ RTClib

Hinweis: Sollten diese Ordner nicht existieren, dann lege diese einfach neu an.



Nach der Installation starten wir die Arduino-Software und öffnen ein Beispiel:

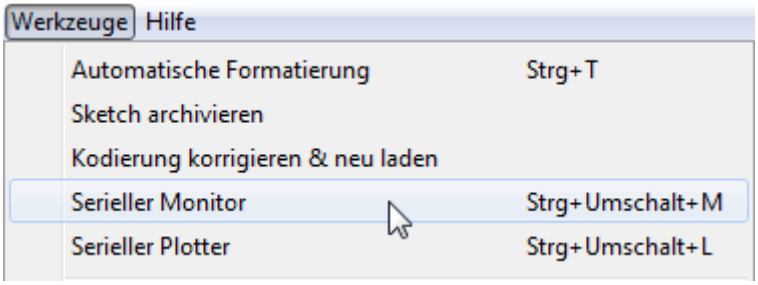


Wähle dazu unter Datei > Beispiele > RTClib > ds1307 aus.

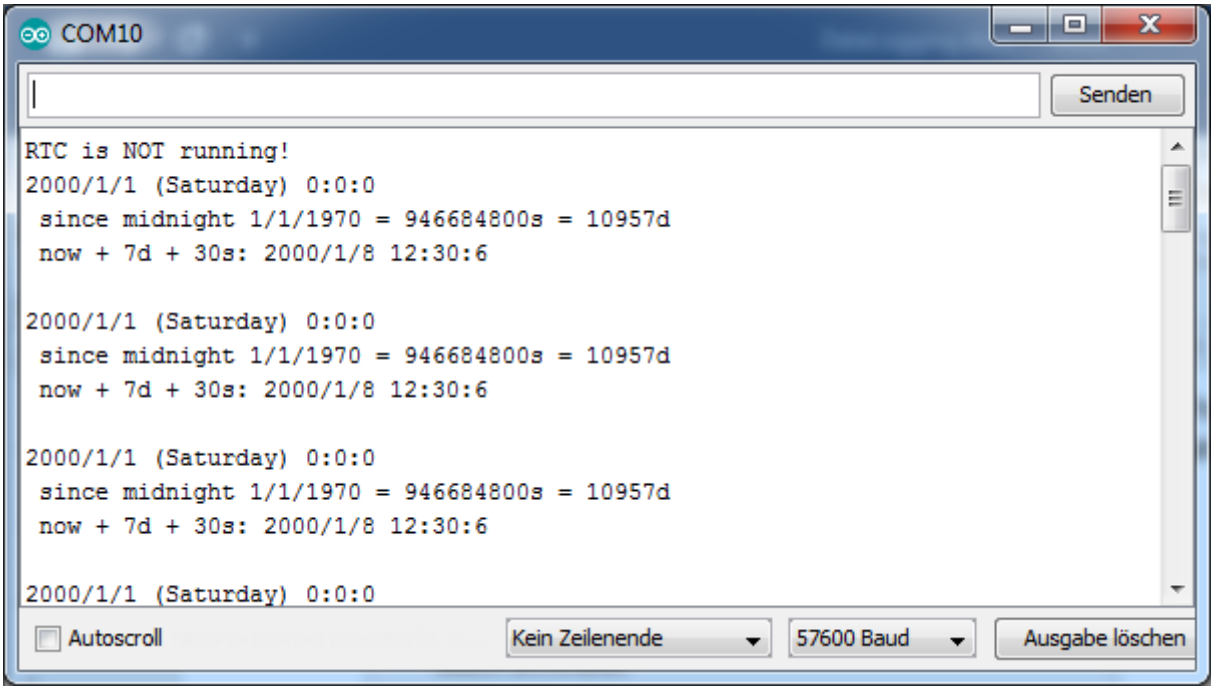
Dein Arduino Board unter der Boardverwaltung sollte auch schon richtig konfiguriert sein, dann kannst du mit dem Übertragen beginnen.

Dazu klicken wir oben auf  . Nach kurzer Zeit wird das Programm auf den Arduino geladen und nun Öffnen wir den Serial Monitor in der Arduino Software:

Werkzeuge > Serial Monitor



Nach dem öffnen muss unten rechts noch die Baudrate auf 57600 Baud umgestellt werden.



Nach einem Reset wird sofort „RTC is NOT running“ übertragen, das bedeutet, das die Echtzeituhr noch nicht aktiviert und konfiguriert wurde.

Zum Aktivieren der Echtzeituhr ändern wir den Code aus dem Beispiel etwas ab und nehmen bei einem Kommentar die „//“ Zeichen am Zeilenanfang heraus.

Suchen wir diese Codezeilen:

```
if (! rtc.isrunning()) {
   Serial.println("RTC is NOT running!");
   // following line sets the RTC to the date & time this sketch was compiled
   // rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
   // This line sets the RTC with an explicit date & time, for example to set
   // January 21, 2014 at 3am you would call:
   // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
 }
```

Die Zeile `// rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));` passen wir nun so an:

```
if (! rtc.isrunning()) {
   Serial.println("RTC is NOT running!");
   // following line sets the RTC to the date & time this sketch was compiled
   rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
   // This line sets the RTC with an explicit date & time, for example to set
   // January 21, 2014 at 3am you would call:
   // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
 }
```

Es wird anschließend die Uhrzeit vom Computer auf den Arduino übernommen.

Alternativ kann auch in der Zeile `rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));` eine Manuelle Zeit angegeben werden.

Es wird nun bei einem Reset kein Fehler mehr angezeigt und die richtige Uhrzeit übertragen.

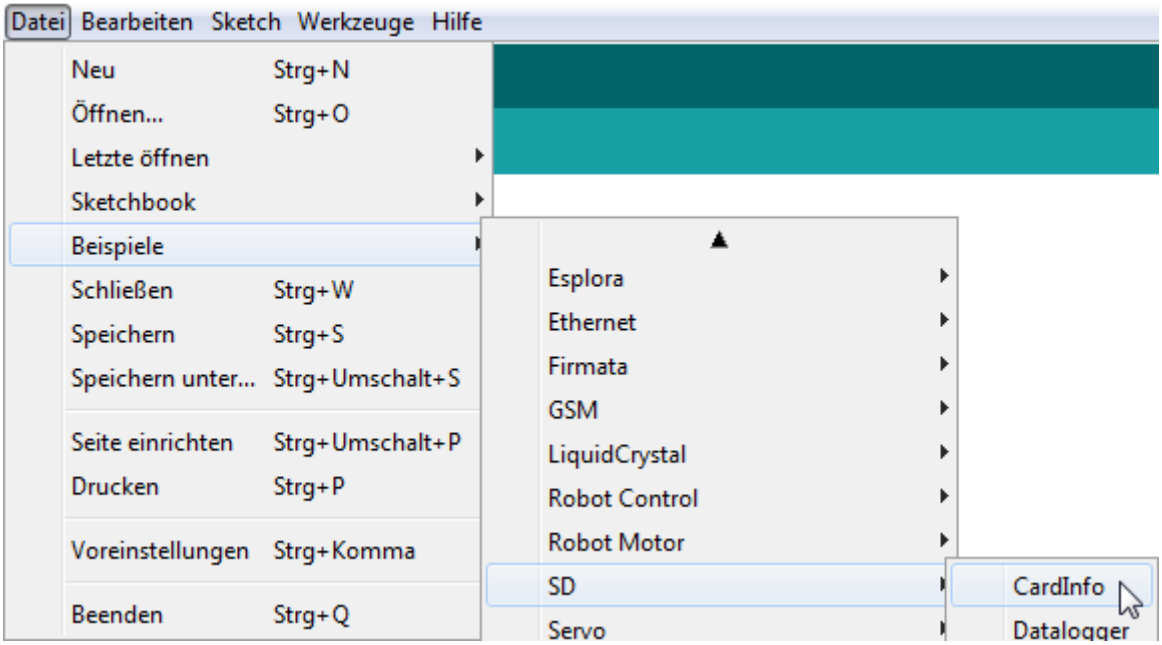
Die Echtzeituhr läuft nun und ist fertig konfiguriert, nun programmieren wir die SD-Karte.

„Programmieren“ des SD-Kartenlesers:

Um eine SD Karte mit dem Arduino verwenden zu können muss diese erst als FAT16 oder FAT32 formatiert werden. Dazu empfiehlt sich das Programm SDFormatter:

https://www.sdcard.org/downloads/formatter_4/

Anschließend lassen wir die SD-Karteninformationen anzeigen. Dazu starten wir:



Wähle dazu unter Datei > Beispiele > SD > CardInfo aus.

Da es verschiedene SD-Karten Shields gibt, müssen wir im Code noch unser Shield angeben:

```
// change this to match your SD shield or module;
// Arduino Ethernet shield: pin 4
// Adafruit SD shields and modules: pin 10
// Sparkfun SD shield: pin 8
// MKRZero SD: SDCARD_SS_PIN
const int chipSelect = 10;
```

Evtl. kann nun noch die Baudrate angepasst werden.

```
Serial.println (57600);
```

Oder wir müssen im Serial Monitor die Baudrate anpassen (auf 9600 Baud).

Wenn wir alles richtig gemacht haben, wird die SD Karte erkannt:

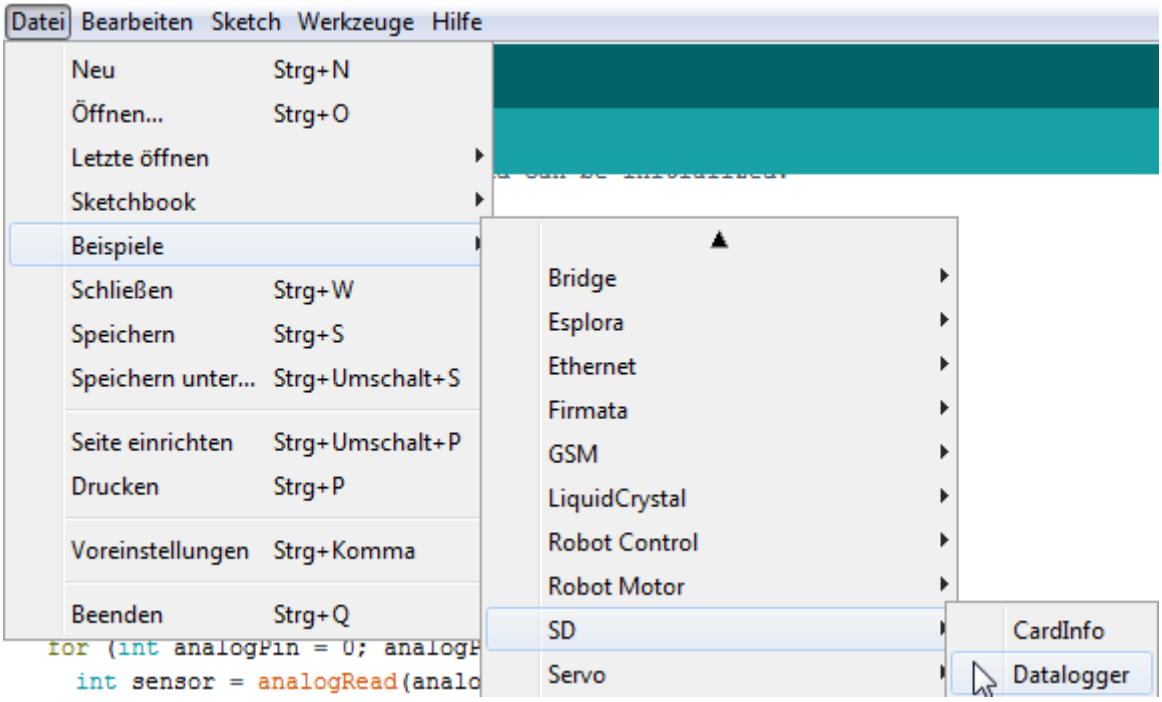
```
Initializing SD card...Wiring is correct and a card is present.

Card type:         SDHC
Clusters:          122112
Blocks x Cluster:  64
Total Blocks:      7815168

Volume type is:    FAT32
Volume size (Kb):  3907584
Volume size (Mb):  3816
Volume size (Gb):  3.73

Files found on the card (name, date and size in bytes):
```

Nun können wir auch daten auf die SD-Karte schreiben. Dafür gibt es das Beispiel DataLogger:



Hier wieder das Shield einstellen:

const int chipSelect = 10;

und evtl. die Baudrate ändern. Nach dem Upload werden die Werte vom Analogeingang 0, 1 und 2 auf die SD Karte in eine Datei „datalog.txt“ geschrieben.

Du hast es geschafft dein Datenlogger hat eine aktuelle Zeit und schreibt Messwerte auf die SD-Karte!

Ab jetzt heißt es lernen und eigene Projekte verwirklichen.

Und für mehr Hardware sorgt natürlich dein Online-Shop auf:

https://az-delivery.de

Viel Spaß!
Impressum

https://az-delivery.de/pages/about-us