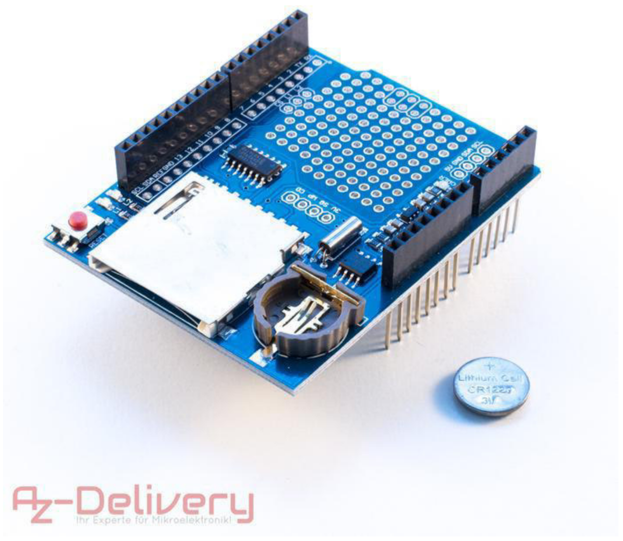
Welcome!

And thank you for purchasing our AZ-Delivery Data Logger module for the Arduino. On the following pages, we will take you through the first steps of the installation process on the Arduino.
We wish you a lot of fun!



This data logger module is suitable for FAT16/FAT32 formatted SD cards. The integrated 3,3V level-shifter-circuit protects the SD card from damage. With the real-time clock, time will not be stopped, even if the Arduino is not being used.

Wiring the module with an Arduino Uno:

The wiring is relatively simple. You should put the module on one of the compatible Arduinos. The following are compatible:

Arduino UNO
Duemilanove
Diecimila
Leonardo
ADK / Mega R3

Or similar Arduinos
After the shield has been plugged in, then the Arduino can be supplied with electric power.

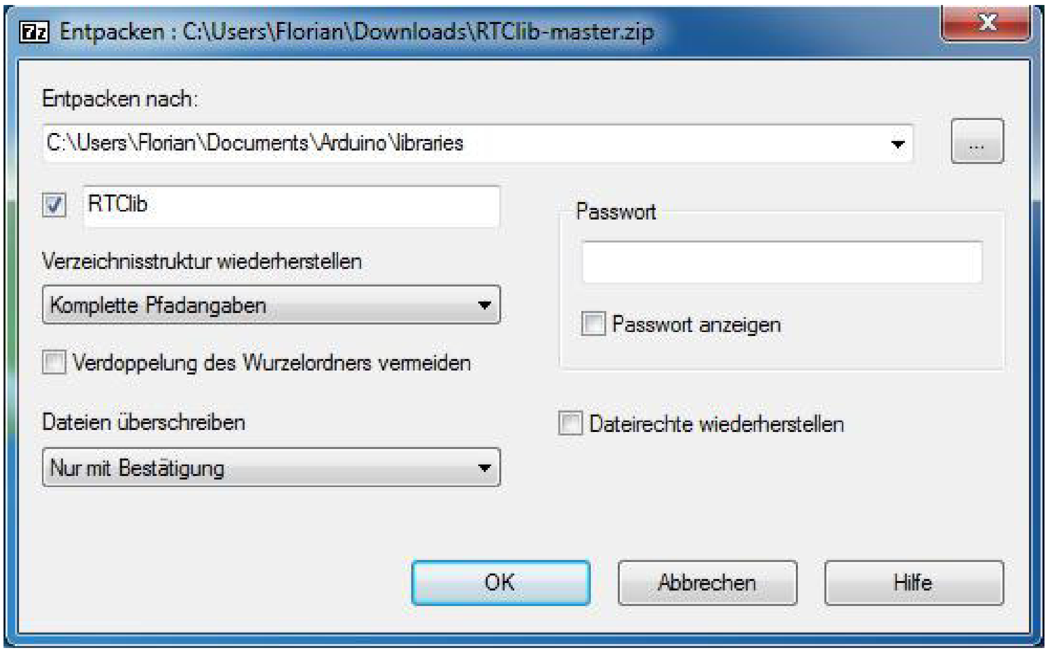
„Programming“ the Arduino:

Before we can start with the programming process, we need to first download and install the corresponding libraries from *git*.

Download the resources from here:
https://github.com/adafruit/RTClib/archive/master.zip

Unpack this zip file (with 7zip) in the folder: [User Directory (C:\User\Florian\) \ My Documents \ Arduino \ libraries \ RTClib

Note: If this folder does not exist, you should simply create it.



Entpacken : C:\Users\Florian\Downloads\RTClib-master.zip

Entpacken nach:
C:\Users\Florian\Documents\Arduino\libraries

[✓] RTClib

Verzeichnisstruktur wiederherstellen
Komplette Pfadangaben

[ ] Verdoppelung des Wurzelordners vermeiden

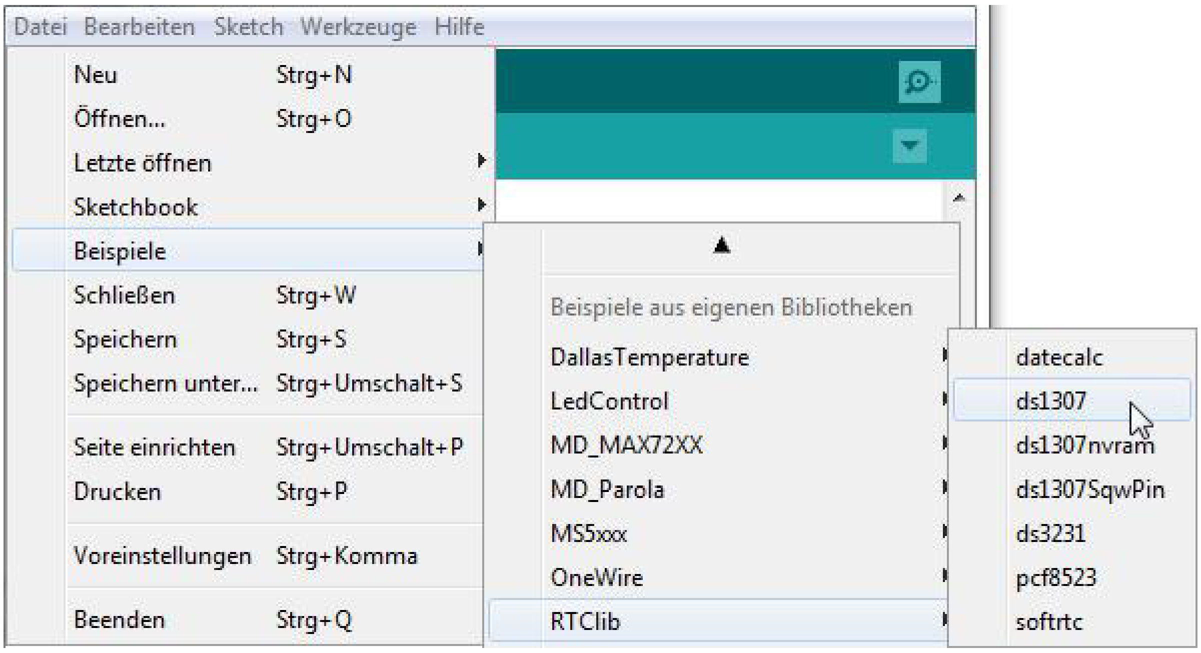
Dateien überschreiben
Nur mit Bestätigung

Passwort

[ ] Passwort anzeigen

[ ] Dateirechte wiederherstellen

After the installation, we start the Arduino software and open an example:

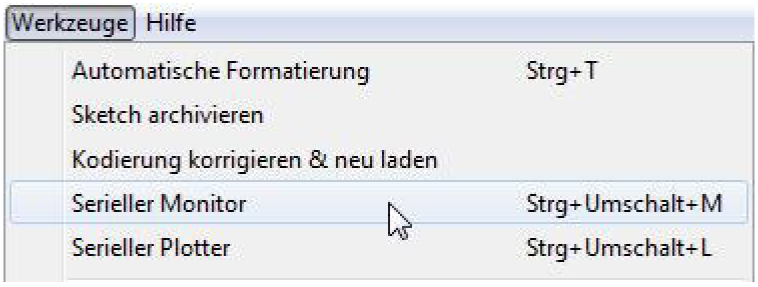


Select under *File* > *Examples* > RTClib > **ds1307**.

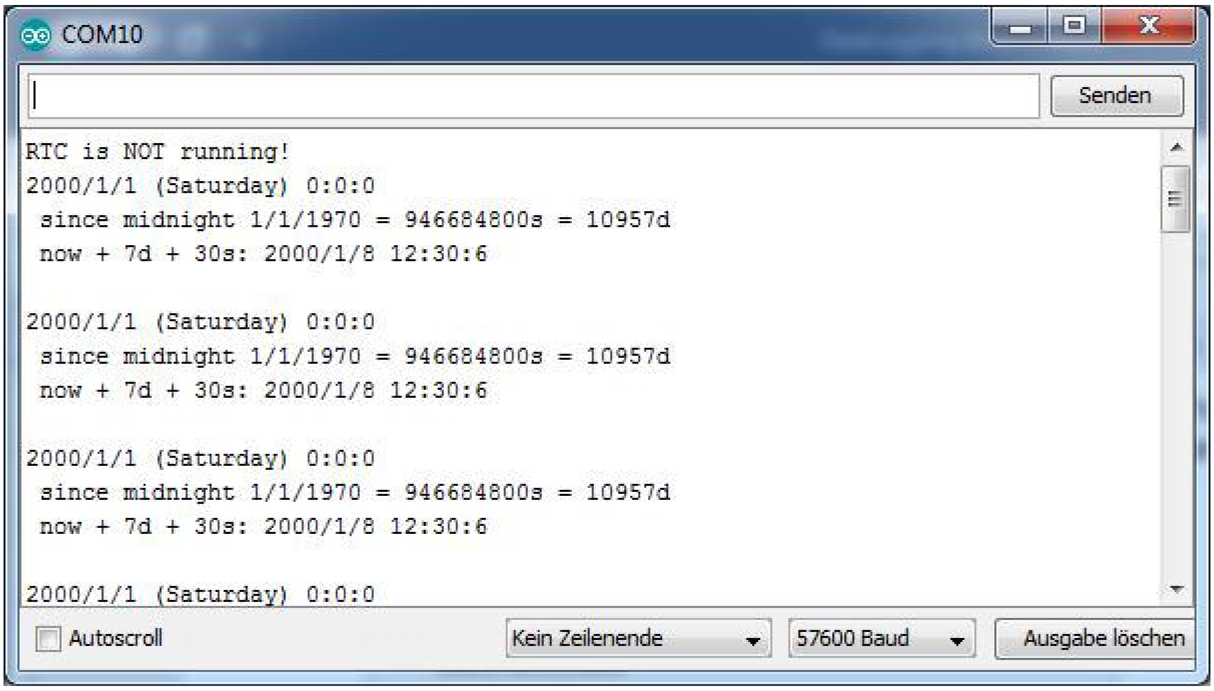
Your Arduino board should already be correctly configured, under the board management. Only then you can begin with the broadcast.

For that, you should click on. After a short period of time, the program will be loaded onto the Arduino, and now you should open the serial monitor in the Arduino software:

Tools > Serial Monitor



After opening, the baud rate must be changed to 57600.



57600 Baud

```
RTC is NOT running!
2000/1/1 (Saturday) 0:0:0
 since midnight 1/1/1970 = 946684800s = 10957d
 now + 7d + 30s: 2000/1/8 12:30:6

2000/1/1 (Saturday) 0:0:0
 since midnight 1/1/1970 = 946684800s = 10957d
 now + 7d + 30s: 2000/1/8 12:30:6

2000/1/1 (Saturday) 0:0:0
 since midnight 1/1/1970 = 946684800s = 10957d
 now + 7d + 30s: 2000/1/8 12:30:6

2000/1/1 (Saturday) 0:0:0
```

After a reset, „RTC is NOT running“ is immediately transmitted, which means that the real-time clock has not yet been activated and configured.

To activate the real-time clock, change a little the code from the example, and erase, from a comment, the "//" characters that are placed at the beginning of the line.

We look for these lines of code:

```
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));
 }
```

We adapt this line

```
// rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
```

as follows:

```
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));
 }
```

Subsequently, the time will be transferred from the computer to the Arduino.

Alternatively, manual time can also specified in the line:

```
rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
```

Now by reset, no error will be shown, and the correct time will be transmitted.

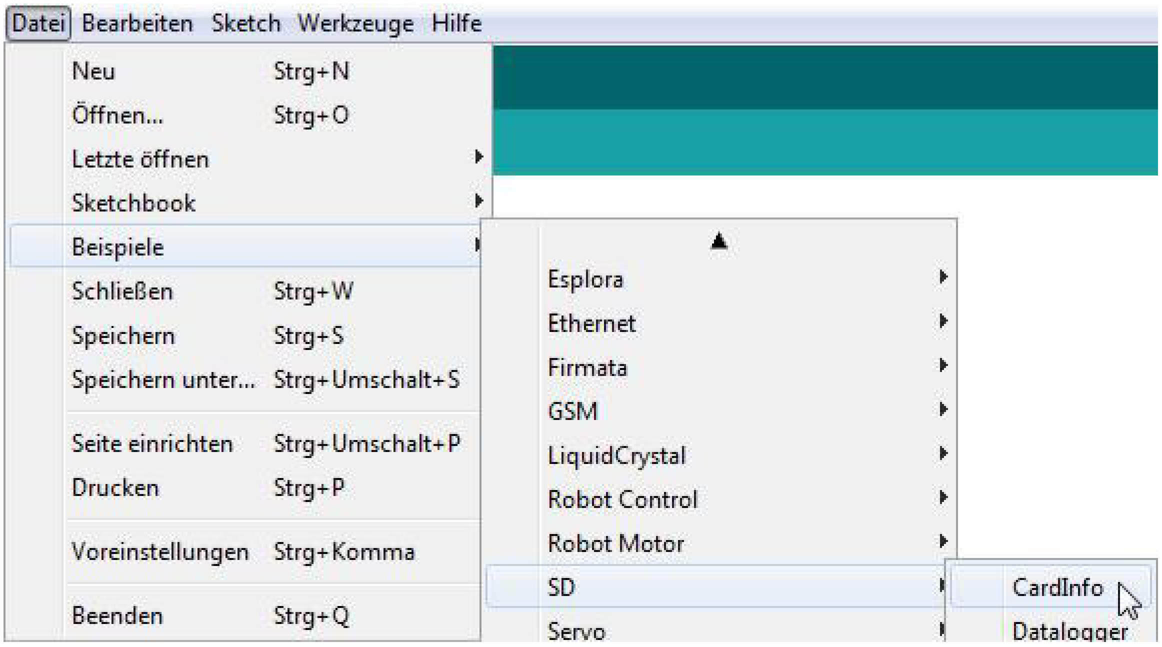
The real-time clock is now running and completely configured. Now we should program the SD card.

„Programming“ the SD card reader:

In order to use an SD card with the Arduino, it has to be firstly formatted as FAT16 or FAT32. For that, we recommend the SDFormatter program:

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

Then we let the SD card information to be shown. To do this, we start:



Select under *File* > *Examples* > *SD* > ***CardInfo***.

Since there are different SD cards shields, we still need to indicate our shield in the code:

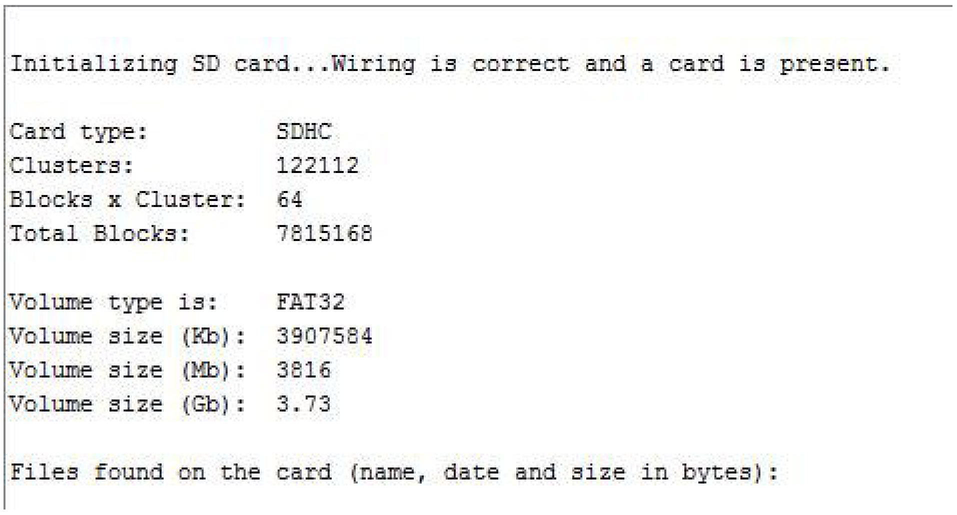
```
//   change this to match your SD shield or module;
//   Arduino Ethernet shield: pin 4
//   Adafruit SD shields and modules: pin 10
//   Sparkfun SD shield: pin 8
//   MKRZero SD: SDCARD_SS_PIN
const int chipSelect = 10;
```

Possibly, the baud rate can now still be adjusted.

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

Or the baud rate in the serial monitor needs to be adjusted (to 9600 Baud).

If everything has been correctly completed, then the SD card will be recognized:

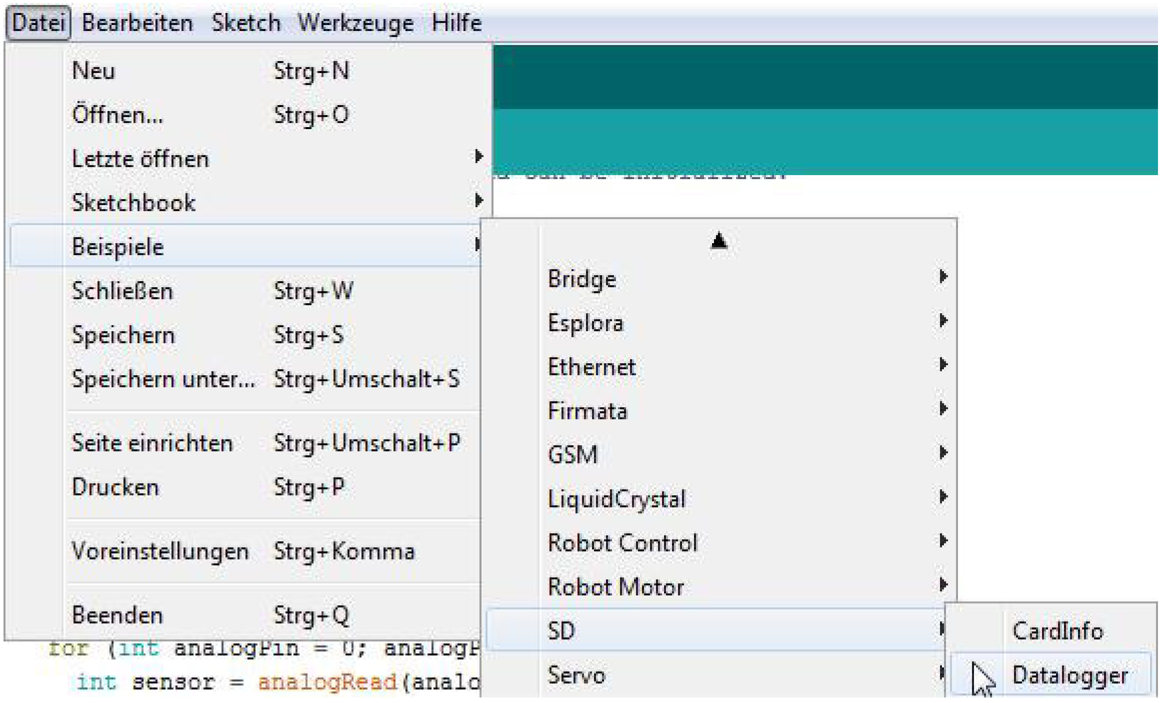

```
Initializing SD card...Wiring is correct and a card is present.

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):
```

Now we can also write data on the SD card. For that, there is the example DataLogger:



Here adjust again the shield:

const int chipSelect = 10;

And, if possible, also change the baud rate. After the upload, the values from the analogue input 0, 1 and 2 are written on the SD card in a file, named „datalog.txt“.

You did it! Your Datalogger has the current time and it also writes test results on the SD card!

Now it is time to learn and actualize your own projects.

For more hardware, our online store is always at your disposal:

https://az-delivery.de

Enjoy!
Imprint

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