Willkommen!

Und herzlichen Dank für den Kauf unseres DatenLogger Moduls. Auf den folgenden Seiten gehen wir mit dir gemeinsam die ersten Programmierschritte durch. Zusätzlich erhältst du noch einige wichtige Informationen über dein neues Modul

Viel Spaß!

Der Datenlogger Shield ist ein Aufsatz für den Arduino Uno und geeignet für Anwendungen, bei denen eine unabhängige Datenaufzeichnung bei gleichzeitig großem Speicherplatz ankommt.

Das Modul hat neben einen SD Kartenhalter eine batteriegepufferte Echtzeituhr (DS1307 Z), einen 3,3Volt Level Konverter (74HC125D) für die SD Kartenkommunikation, zwei konfigurierbare rote LED's mit Vorwiederstand und einen großen Prototyping Bereich, auf dem eigene Schaltungen aufgebaut werden können.

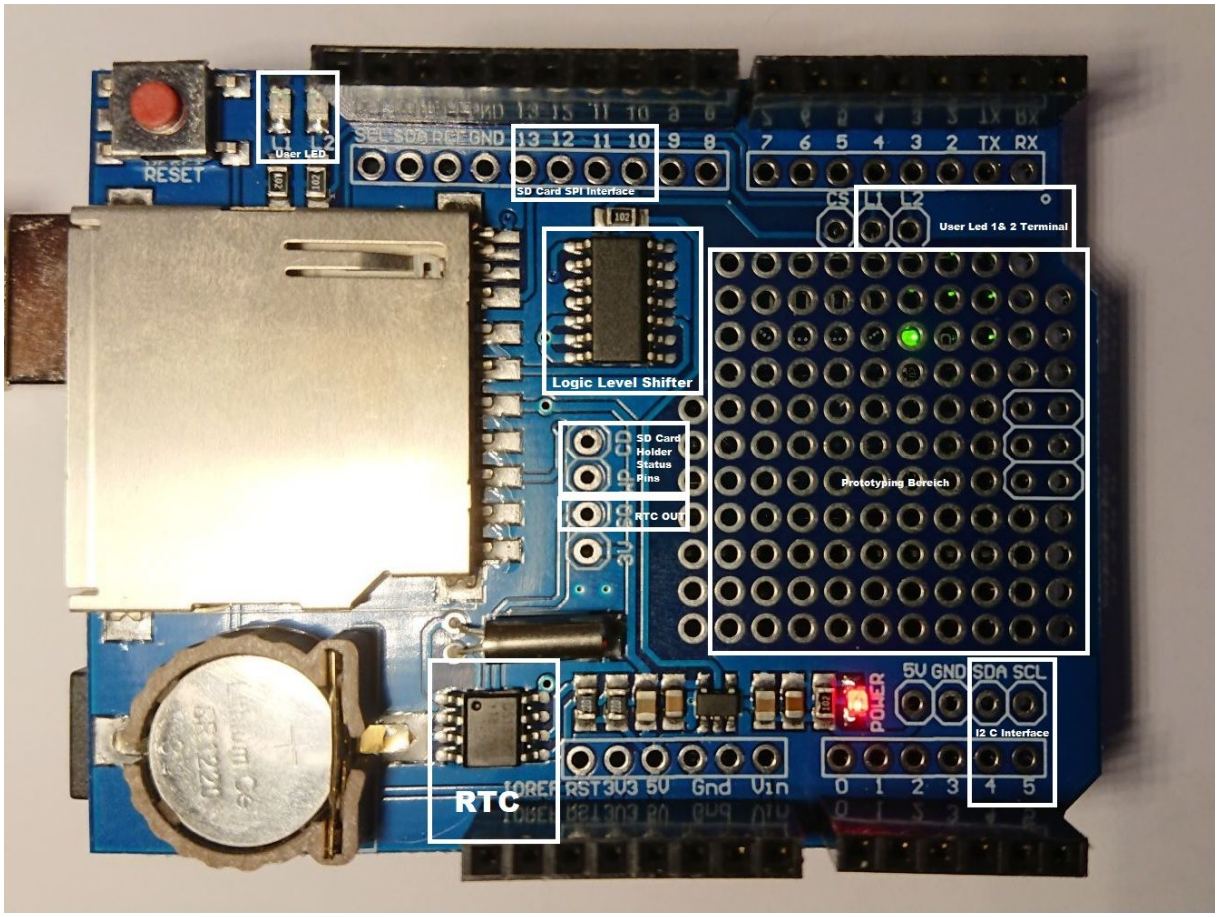
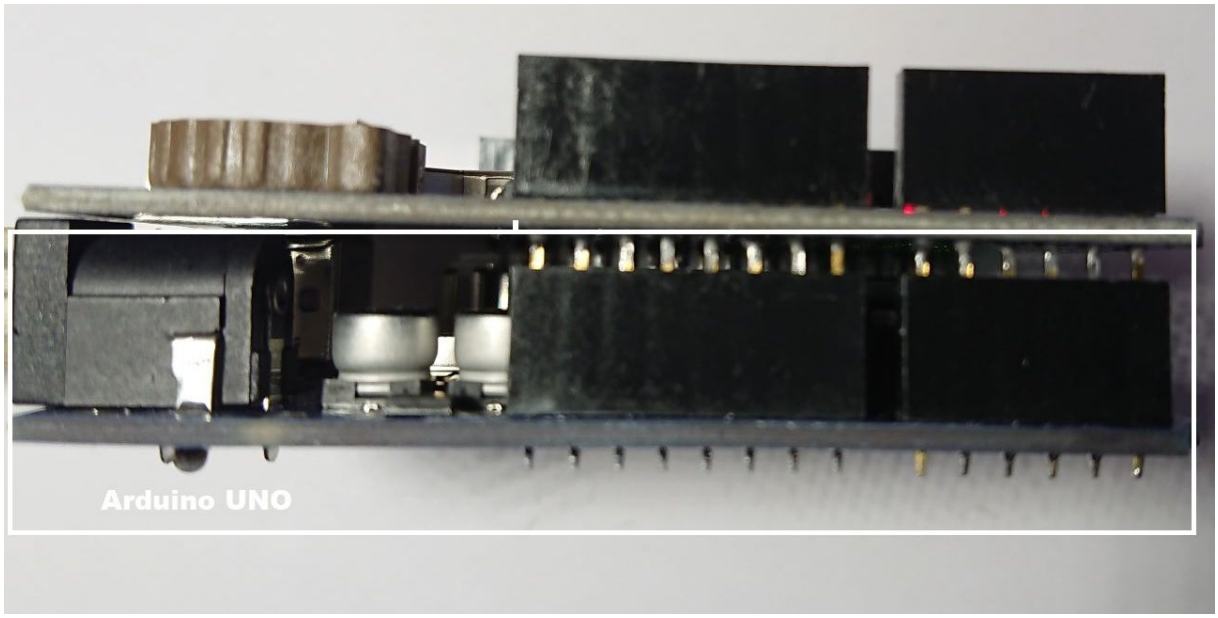


Bild: Das Datenloggermodul mit den verschiedenen Bereichen. Die Echtzeituhr (RTC) ist der SMD DS1307 Z Chip neben der Lithium CR 1220 Batterie.

Das Modul wird auf den Arduino aufgesteckt:



Die Pins des Moduls dürfen nicht überstehen!

Das Pinmapping des Moduls ist:

Modulfunktion	Arduino Port
Echtzeituhr (I2C Interface)	A4 (Analog Port)
Echtzeituhr (I2C Interface)	A5 (Analog Port)
SD Kartenkommunikation (Chip Select)	10
SD Kartenkommunikation (SPI Interface)	11
SD Kartenkommunikation (SPI Interface)	12
SD Kartenkommunikation (SPI Interface)	13

Für eigene Erweiterungen beachte bitte, dass die beiden Busse (SPI und I2C) durch das Modul verwendet werden bereits belegt sind.

Die SD Kartenkommunikation findet über den SPI Bus statt, die Echtzeituhrkommunikation über I2C. Daher ist der RTC Chip auch über die I2C Schnittstelle mit dem Arduino verbunden.

Beschreiben und Lesen von SD Karten

Um die Basisfunktion des Moduls, das lesen und schreiben von SD Karten nutzen zu können, müssen wir die Bibliothek „SD“ einbinden.

Diese Bibliothek ist standardmäßig in jeder Arduino DIE ab Version 1.80 verfügbar und muss nicht separat eingebunden werden!

Eine komplette Befehlsreferenz der SD Karten – Bibliothek findest du auf:

https://www.arduino.cc/en/Reference/SD

Um nun ein ersten Funktionstest unseres Moduls zu bekommen, stecken wir eine formatierte FAT32 SD Karte in den SD Leserschacht des Moduls.

Wichtig: Die SD Karte MUSS FAT32 formatiert sein. Andere Formate werden von der Arduino eigenen SD Bibliothek nicht unterstützt!

Auch nicht formatierte Karten oder mehrere Partitionen auf der SD Karte werden nicht erkannt. Wir laden nun den Beispielsketch auf

https://www.arduino.cc/en/Tutorial/CardInfo

in unsere IDE oder laden das Beispiel im Editor selbst über Datei -> Beispiele –> SD -> CardInfo in unsere IDE ein. Wir passen die Zeile chipSelect wie folgt an:

```
const int chipSelect = 10;
```

Falls wir alles richtig gemacht haben erhalten wir nun folgende Ausgabe auf der Seriellen Schnittstelle beim Einlegen einer formatierten SD Karte:

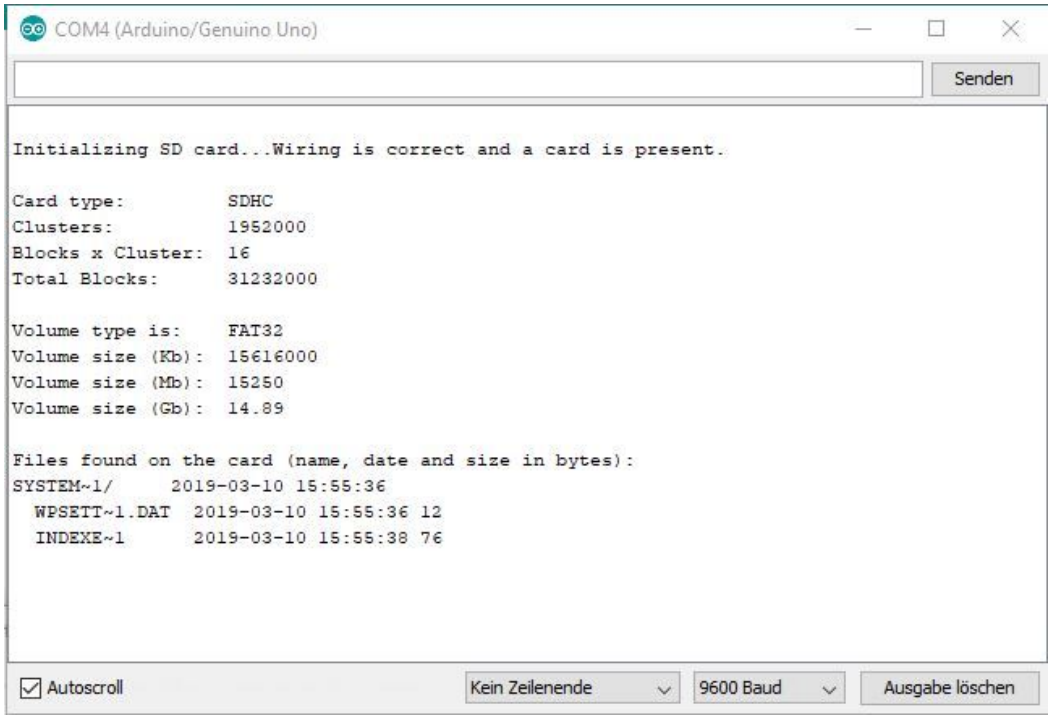


Bild: Beispiel einer 16 GB Karte

Schreiben und Lesen der DS1307 EchtzeitUhr

Um die Echtzeituhr (RTC) in unserem Sketch verwenden zu können, sind einige Vorbereitungen zu treffen. Damit wir nicht alle Grundfunktionen der RTC aufwändig selbst programmieren müssen, empfiehlt es sich, die fertige RTC Bibliothek von Adafruit zu verwenden. Dazu erstellen wir in dem Ordner Eigene Dateien/Dokumente/Arduino/libraries/ einen neuen Unterordner mit dem Namen RTClib, und kopieren die Datei RTClib-master.zip in den neu erstellten Ordner. Die Datei laden wir vorher von GitHub unter der URL

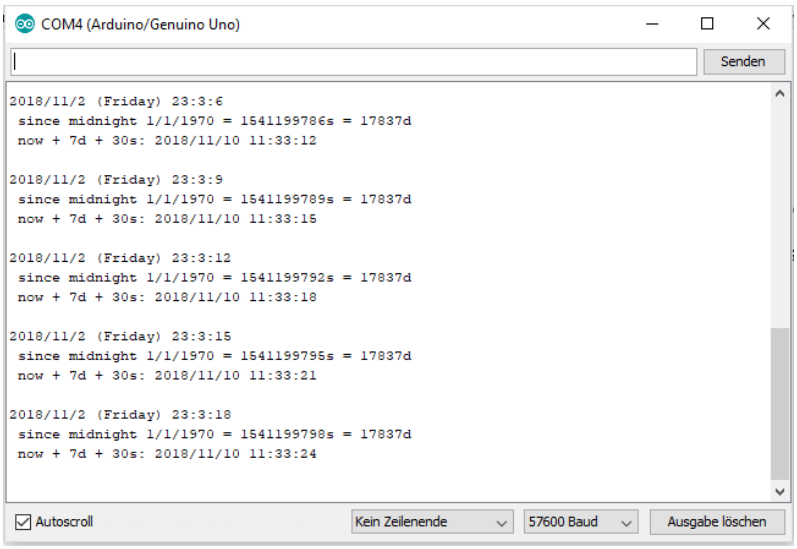
https://github.com/adafruit/RTClib

herunter. (Auf der Webseite auf "Clone or Download" klicken. Dann Download Zip auswählen)

Wir starten wir danach unsere Arduino DIE und binden unter Sketch -> Bibliothek einbinden -> ZIP. Bibliothek einfügen unsere vorher heruntergeladene ZIP Datei ein. Unter Datei -> Beispiele >-RTCLib wählen wir nun den Code für den DS1307 aus. Da die RTC im Kaufzustand und bei Batteriewechsel deaktiviert ist und zudem die falsche Uhrzeit Datum enthält, müssen wir noch folgende Zeilen im Code aktivieren (// entfernen):

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

Durch Aktivierung der Zeile wird initial einmalig das aktuelle Datum und die aktuelle Uhrzeit in die RTC geschrieben. Diese speichert die Uhrzeit und die Uhrzeit fängt Chipintern an zu laufen. Wir kompilieren daher den Code und führen Ihn auf unserem Arduino mit aufgesetztem Datenlogger auf. Wenn alles funktioniert hat sehen wir folgende Ausgabe:

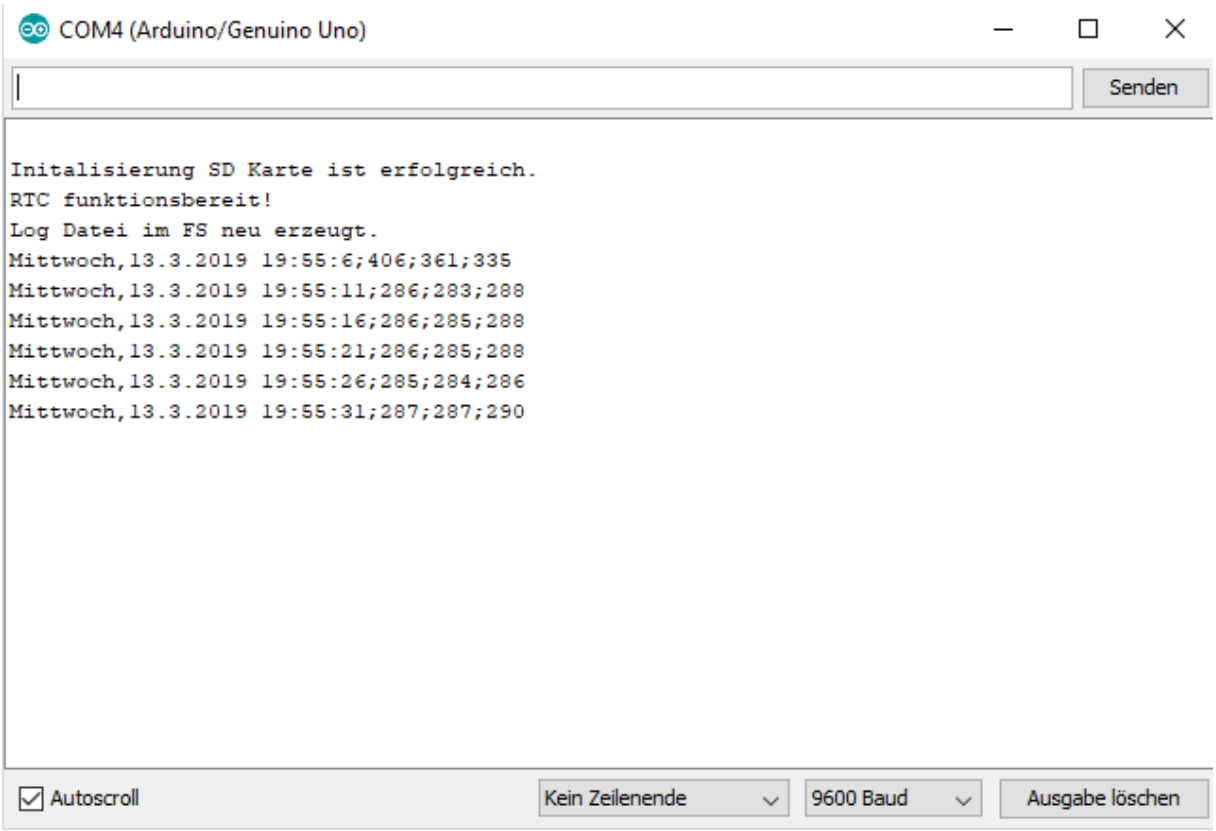


Wichtig: Bei einem Batteriewechsel muss diese Prozedur wiederholt werden!

Nun können wir die beiden Teile (SD Karte schreiben und Uhrzeit auslesen) miteinander kombinieren. Im nachfolgenden Sketch wird das gemacht:

```
#include <SPI.h>  // Einbinden der SD Bibliothek
#include <SD.h>   // Einbinden der SD Bibliothek
#include <Wire.h> // Einbinden der SPI Bibliothek
#include "RTClib.h" // Einbinden der Adafruit DS1307 RTC Bibliothek
#define MeasureDelay 5000 // Pause in ms zwischen einzelnen Messungen
const String LogFileName = "/Daten.csv";
const int chipSelect = 10; // Für Datenlogger Modul am Arduino Uno so lassen
RTC_DS1307 rtc;
char daysOfTheWeek[7][12] = {"Sonntag", "Montag", "Dienstag", "Mittwoch", "Donnerstag", "Freitag", "Samstag"};
void setup()
{
  Serial.begin(9600); // Öffne serielle Schnittsatelle
  while (!Serial) {   // Warte auf seriellen Port
  }
  Serial.print("\nInitalisierung SD Karte ");
  if (!SD.begin(chipSelect))
   {
    Serial.println("ist fehlgeschlagen!");
    while (1);
   }
  Serial.println("ist erfolgreich.");
  if (! rtc.begin())
   {
    Serial.println("RTC nicht an I2C Bus gefunden!");
    while (1);
   }
  if (! rtc.isrunning())
   {
    Serial.println("RTC laueft nicht! Bitte initalisieren!");
    while (1);
   }
  Serial.println("RTC funktionsbereit!");
  if (!(SD.exists(LogFileName)))
   {
    File LogFilePtr = SD.open(LogFileName,FILE_WRITE); // Datei zum Schreiben öffnen. Es kann jeweils nur 1 Datei
// gleichzeitig geschrieben werden !
    if (LogFilePtr) // Wenn Datei zum schreiben bereit ist, schreibe Kopfzeile (Header) hinein
     {
      LogFilePtr.println("Zeitstempel;Wert Analogport 1;Wert Analogport 2;Wert Analogport 3");
      Serial.println("Log Datei im FS neu erzeugt.");
      LogFilePtr.close();  // Datei schließen
     }
   }
}
void loop(void)
{
  DateTime now = rtc.now();   // Hole aktuelle Uhrzeit aus RTC
  String dataString = "";     // erzeuge neuen Eintrag für CSV Datei
  dataString = String(daysOfTheWeek[now.dayOfTheWeek()]) + "," + (String(now.day())) + "." + now.month();
  dataString += String(".") + now.year() + " " + now.hour() +":" + now.minute()+ ":" + now.second()+ ";"; // Erzeuge
//Zeitstempel
  // Lese die Werte der übriggebliebenen 3 Analogeingänge und füge sie dem String hinzu (A4 und A5 für I2C Bus
//genutzt)
  for (int analogPin = 0; analogPin < 3; analogPin++)
    {
    int sensor = analogRead(analogPin);
    dataString += String(sensor);
    if (analogPin < 2)
      {
       dataString += ";";
      }
    }
  File LogFilePtr = SD.open(LogFileName, FILE_WRITE); // Datei zum Schreiben öffnen.
  if (LogFilePtr) // Wenn Datei zum schtreiben bereit ist, schreibe hinein
   {
    LogFilePtr.println(dataString);
    LogFilePtr.close();  // Datei schließen
    Serial.println(dataString); // Aus serieller schnittstelle auch ausgeben
   }
   else
   {
    Serial.println("Fehler beim oeffnen");
   }
  delay(MeasureDelay);
}
```

Die Funktion dayOfTheWeek() generiert Werte zwischen 0 und 6. Der Wert 0 steht hierbei für Sonntag. Nachdem wir den o.g. Sketch in unserer Arduino DIE kompiliert haben und auf unseren Arduino mit Datenloggershield und FAt32 formatierter SD Karte hochgeladen haben, sehen wir folgende Ausgabe an unserer seriellen Schnittstelle:



Herzlichen Glückwunsch! Dein Datenloggershield arbeitet nun korrekt und schreibt in die Datei „daten.csv“ auf der SD-Karte alle 5 Sekunden einen neuen Eintrag mit Uhrzeit und den Werten der AnalogPorts 0 bis 2.

Wenn die SD-Karte in einen SD-Kartenleser am Personal-Computer eingelegt wird, ist eine neue Microsoft Excel kompatible csv Datei auf dem Datenträger sichtbar:

Diese kann normal mit Excel geöffnet werden und ist wie folgt formatiert:

Zeitstempel	Wert Analogport 1	Wert Analogport 2	Wert Analogport 3
Mittwoch,13.3.2019 19:55:6	406	361	335
Mittwoch,13.3.2019 19:55:11	286	283	288
Mittwoch,13.3.2019 19:55:16	286	285	288
Mittwoch,13.3.2019 19:55:21	286	285	288
Mittwoch,13.3.2019 19:55:26	285	284	286
Mittwoch,13.3.2019 19:55:31	287	287	290
Mittwoch,13.3.2019 19:55:36	284	282	286
Mittwoch,13.3.2019 19:55:41	287	285	288
Mittwoch,13.3.2019 19:55:46	286	285	288
Mittwoch,13.3.2019 19:55:51	284	283	287
Mittwoch,13.3.2019 19:55:56	288	287	290
Mittwoch,13.3.2019 19:56:2	288	287	290
Mittwoch,13.3.2019 19:56:7	285	283	288

Der Wert „Analogport 1“ ist der AD Konversationswert von A0, der Wert „Analogport 2“ ist der AD Konversationswert von A1 und der Wert „Analogport 3“ ist der AD Konversationswert von A2.

Weitere Hinweise zu der Echtzeituhr:

Die Echtzeituhr hat einen typischen Gangfehler von ca. 1 Minute pro Monat. Sobald die Echtzeituhr die Stromversorgung UND den Batterieanschluss verliert, verliert die Uhr die aktuelle Uhrzeit und bleibt stehen.

Ab jetzt heißt es Experimentieren.

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

https://az-delivery.de

Viel Spaß!
Impressum
https://az-delivery.de/pages/about-us