Wetterdaten aufzeichnen

Die mit dem Temperatursensor DHT22 gemessene Temperatur soll mit aktuellem Datum und aktueller Zeit auf eine SD-Karte gespeichert werden.

So soll es aussehen:

```
Samstag 5.1.2019 14:28:44 Uhr
Temperatur:              20,90
Luftfeuchtigkeit in %:   51,90
Schreibe Messdaten in Messung.csv:
-----------------------------------
Abgeschlossen.

Samstag 5.1.2019 14:29:44 Uhr
Temperatur:              20,90
Luftfeuchtigkeit in %:   51,90
Schreibe Messdaten in Messung.csv:
-----------------------------------
Abgeschlossen.
```

Anzeige im Seriellen Monitor

Datei Bearbeiten Ansicht Einfügen Format Vorlagen Tabelle Daten Extras

Liberation Sans 10

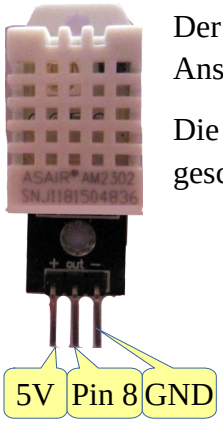
A1

	A	B	C	D
1				
2	Datum	Zeit	Temperatur in °C	Luftfeuchtigkeit in %
3	Samstag 5.1.2019	13:22:23 Uhr	21,3	51
4				
5	Datum	Zeit	Temperatur in °C	Luftfeuchtigkeit in %
6	Samstag 5.1.2019	13:23:23 Uhr	21,3	51
7				
8	Datum	Zeit	Temperatur in °C	Luftfeuchtigkeit in %
9	Samstag 5.1.2019	13:24:24 Uhr	21,3	51,1

Tabelle in Libreoffice

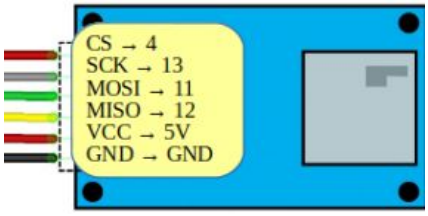
Benötigte Bauteile:

- RTC-Modul DS3231
- Temperatur-/Feuchtigkeitssensor DHT22
- SD-Kartenleser
- Leitungsdrähte



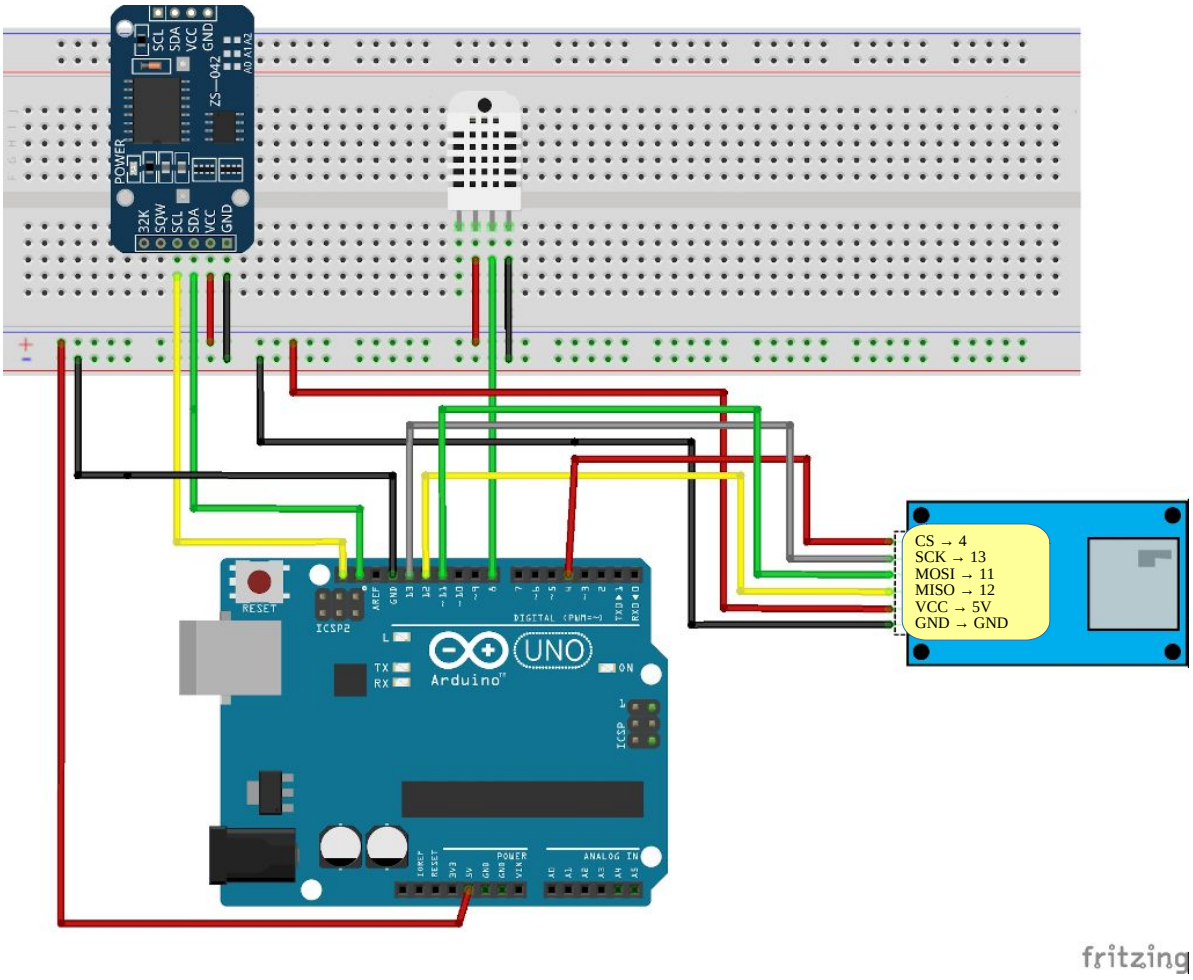
Der Sensor DHT22 misst die Temperatur und die Luftfeuchtigkeit. Er besitzt drei Anschlüsse.

Die Version mit vier Pins muss in der Reihenfolge 5V → Daten (Pin) → leer → GND geschaltet werden.



Achte auf die Pin-Belegung des SD-Kartenlesers!
Die SD-Karte muss mit FAT32 formatiert sein!

Baue die Schaltung auf.

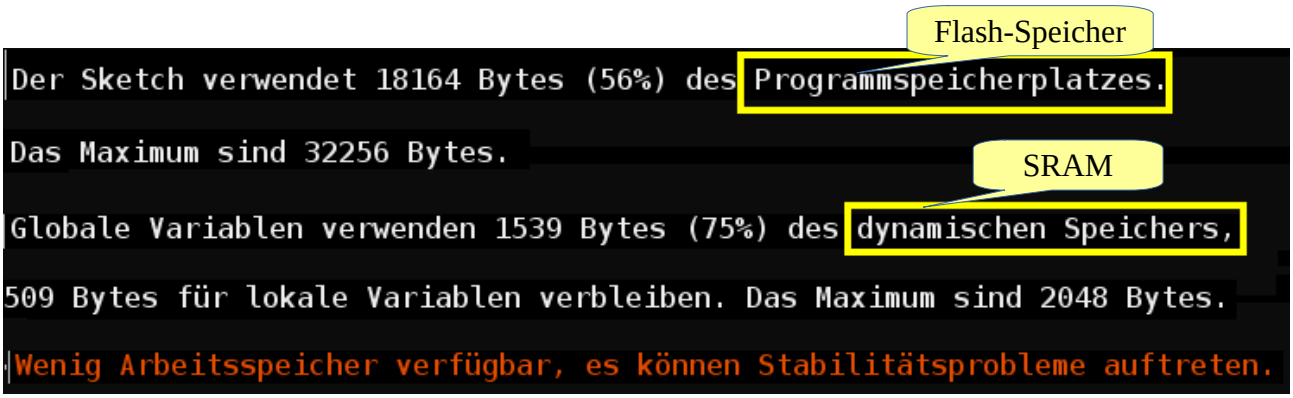


Je größer die Programme werden, je mehr Variable verwendet werden, desto größer ist der Speicherbedarf im RAM. Übersteigt er die Grenze von 75%, wird eine Warnung ausgegeben.

Der physische Speicher kann nicht vergrößert werden, es gib aber Strategien, den Speicherbedarf des Programms zu verringern. Eine soll hier vorgestellt werden.

Der Arduino verfügt über drei Speicherplätze:

Flash 32 kB (32256 Bytes)	5 kB sind für den Bootloader reserviert, der Rest kann für das Programm verwendet werden der gespeicherte Inhalt bleibt nach dem Ausschalten erhalten
SRAM 2 kB (2048 Bytes)	(static random access memory) hier werden die Variablen/Arrays erstellt und verändert der Inhalt wird beim Ausschalten gelöscht
EEPROM 1 kB (1024 Bytes)	hier können Informationen gespeichert werden, die auch nach dem Ausschalten noch zur Verfügung stehen sollen



In den meisten Programmen werden print-Anweisungen ausgeführt. Jede dieser Anweisungen belegt Speicherplatz im Programmspeicher. Da es sich aber nicht um Variable handelt, ist die dauerhafte Speicherung nicht notwendig.

Das F-Makro sorgt dafür, dass der Text nicht im Programmspeicher verbleibt, er wird vielmehr im SRAM abgelegt und stellt so Speicherplatz im Programmspeicher zur Verfügung.

Beispiele:

```
Serial.println(F("Initialisierung abgeschlossen"));
Serial.println(F("Schreibe Messdaten in die Datei Messung.csv: "));
Serial.println(F("-----------------------------------"));
```

Benötigte Bibliotheken:

Sketch → Bibliothek einbinden → Bibliotheken verwalten

Im Kopf des Programms werden die benötigten Bibliotheken eingebunden und die Variablen definiert. Beachte die Kommentare:

```
// Bibliothek für die SD-Karte
# include <SdFat.h>

// Bezeichnung der SD-Karte
SdFat SD;

// Bibliothek für die Kommunikation des Arduino mit Peripheriegeräten
# include <SPI.h>

// Bibliothek für den DHT22
# include <SimpleDHT.h>

// Bibliothek für das RTC-Modul
# include <RTClib.h>

RTC_DS3231 rtc;

// Bezeichnung der Textdatei
File Temperaturmessung;

// Datenpin für das SD-Kartenmodul
int DatenPin = 4;

int SENSOR_DHT22 = 8;

// Namen des Sensors (dht22)
SimpleDHT22 dht22(SENSOR_DHT22);

int SensorLesen;

// Trennzeichen für die CSV-Datei
const String TrennZeichen = ";";

String AktuellesDatum;
String AktuelleZeit;
```

Der setup-Teil initialisiert die SD-Karte, startet das RTC-Modul und legt den pinMode für den Sensor fest.

```
void setup()
{
  pinMode(SENSOR_DHT22, INPUT);
  Serial.begin(9600);
  Serial.println(F("Initialisiere SD-Karte"));
  if (!SD.begin(DatenPin))
  {
    Serial.println(F("Initialisierung fehlgeschlagen!"));
  }
  else Serial.println(F("Initialisierung abgeschlossen"));
  rtc.begin();
  /*
    wenn Datum und Zeit nicht korrekt -> Datum/Zeit setzen
    Jahr, Monat, Tag, Stunde, Minute, Sekunde
    rtc.adjust(DateTime(2018, 10, 25, 17, 28, 30));
  */
}
```

Der loop-Teil: Beachte die Kommentare.

```
void loop()
{
  DateTime aktuell = rtc.now();
  float Temperatur;
  float Luftfeuchtigkeit;

  // Daten lesen
  dht22.read2(&Temperatur, &Luftfeuchtigkeit, NULL);

  ZeitAusgeben(aktuell);

  // in Strings umwandeln, . durch , ersetzen
  String AnzeigeTemperatur = String(Temperatur);
  AnzeigeTemperatur.replace(".", ",");

  String AnzeigeLuftfeuchtigkeit = String(Luftfeuchtigkeit);
  AnzeigeLuftfeuchtigkeit.replace(".", ",");

  Serial.print(F("Temperatur:\t\t"));
  Serial.println(AnzeigeTemperatur);

  Serial.print(F("Luftfeuchtigkeit in %:\t"));
  Serial.println(AnzeigeLuftfeuchtigkeit);
```

```
  /*
    Datei zum Schreíben (FILE_WRITE) öffnen
    wenn sie noch nicht existiert, wird sie erstellt
    wenn Schreiben nicht möglich -> Fehlermeldung
  */
  if (!Temperaturmessung.open("Messung.csv", FILE_WRITE))
  {
    Serial.print(F("Datei kann nicht geöffnet werden!"));
  }

  // wenn die Datei geöffnet werden konnte ...
  if (Temperaturmessung)
  {
    Serial.println(F("Schreibe Messdaten in die Datei Messung.csv: "));
    Serial.println(F("-----------------------------------"));

    // Überschrift in Datei schreiben
    schreibeUeberschrift();
    Temperaturmessung.print(AktuellesDatum + TrennZeichen);
    Temperaturmessung.print(AktuelleZeit + TrennZeichen);

    Temperaturmessung.print(AnzeigeTemperatur + TrennZeichen);
    Temperaturmessung.print(AnzeigeLuftfeuchtigkeit);
    Temperaturmessung.println();

    // Schreibfehler abfragen
    if (!Temperaturmessung.sync() || Temperaturmessung.getWriteError())
    {
      Serial.print(F("Schreibfehler!"));
    }
    else
    {
      // Datei schließen
      Temperaturmessung.close();
      Serial.println(F("Abgeschlossen."));
      Serial.println();
    }

  }

  // Zeit bis zur nächsten Messung (eine Minute)
  delay(60000);
}
```

Im loop-Teil werden zwei Methoden aufgerufen:

```
void ZeitAusgeben(DateTime aktuell)
{
  char Datum[] = "DD.MM.YYYY";
  AktuellesDatum = aktuell.toString(Datum);

  char Zeit[] = "Uhrzeit: hh:mm:ss";
  AktuelleZeit = aktuell.toString(Zeit);
  Serial.println(AktuellesDatum + " " + AktuelleZeit);
}

void schreibeUeberschrift()
{
  Temperaturmessung.println();
  Temperaturmessung.print(F("Datum"));
  // Trennzeichen für die CSV-Datei
  Temperaturmessung.print(TrennZeichen);

  Temperaturmessung.print(F("Zeit"));
  Temperaturmessung.print(TrennZeichen);

  Temperaturmessung.print(F("Temperatur in °C"));
  Temperaturmessung.print(TrennZeichen);

  Temperaturmessung.print(F("Luftfeuchtigkeit in %"));
  Temperaturmessung.println();
}
```