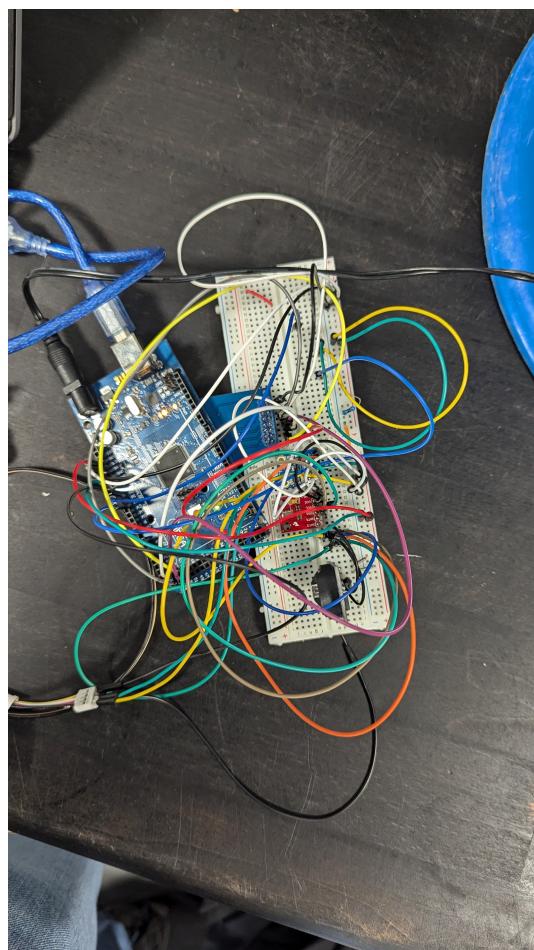


# Arbejdsgruppe i Science klubben

## Sådan byggede vi vores egen datalogger

Af Science klubbens medlemmer

15. januar 2025



Figur 1: Arduinoen i virkeligheden under udvikling

## Indhold

<b>1</b>	<b>Introduktion</b>	<b>3</b>
1.1	Forbindelser i Arduinoen . . . . .	3
1.1.1	Opbygning af datalogger visualiseret . . . . .	5
<b>2</b>	<b>Kode og Flowcharts</b>	<b>6</b>
2.1	Flowcharts . . . . .	6
2.2	Kode beskrivelse . . . . .	8
2.3	Kode valg . . . . .	9
<b>3</b>	<b>RTC Module (Real Time Clock)</b>	<b>10</b>
3.1	Kode . . . . .	10
<b>4</b>	<b>SD Card Module</b>	<b>12</b>
4.1	Kode . . . . .	12
<b>5</b>	<b>Digital temperatur sensor</b>	<b>14</b>
5.1	Kode . . . . .	14
<b>6</b>	<b>pH Sensor</b>	<b>15</b>
6.1	Kode . . . . .	15
<b>7</b>	<b>UV sensor</b>	<b>16</b>
7.1	Kode . . . . .	16
<b>8</b>	<b>Tryksensor</b>	<b>17</b>
8.1	Kode . . . . .	17
<b>9</b>	<b>Bilag</b>	<b>18</b>
<b>10</b>	<b>Koden</b>	<b>18</b>

# 1 Introduktion

Dette projekt er støttet af NNF (Novo Nordisk Fonden) og formålet med det generelle projekt er at lade elever prøve at arbejde som forskere. Dertil har eleverne udviklet en station som skulle indsamle data. Dataet bliver behandlet af lærere på Rybners HTX, som vil blive brugt i undervisningsforløb til folkeskoler og arbejdsopgaver til gymnasiet. Denne arbejdsgruppe arbejder med at opbygge en datalogger, som skal kunne indsamle diverse data via forskellige sensorer. Dette dokument beskriver, hvordan hver sensor er sat op, hvilke kodevalg der er blevet gjort, og hvordan programmet bliver kørt igennem. Hvert kapitel i dokumentet, efter Kode og Flowcharts, angiver en sensor og hvordan den er sat op.

## 1.1 Forbindelser i Arduinoen

Her er et overblik over hvilke porte som er i brug på Arduino Mega'en. Der er blevet brugt et program kaldet Fritzing til at visualisere, hvordan de forskellige sensorer og moduler er forbundet til Arduinoen (Se kapitel 1.1.1). På grund af det høje antal af moduler og kabler er der oplyst herunder for hvert modul, hvor og hvordan det enkelte modul er forbundet til Arduinoen. (Det vil også stå inde under hver moduls sektion)

RTC (Real Time Clock):

- 5V - 5V
- GND - GND
- SCL - Port 21 (SCL)
- SDA - Port 20 (SDA)

SD Card Module:

- +5 - 5V
- +3.3 - 3.3V
- GND - GND
- CS - Port 53 (SS)
- MOSI - Port 51 (MOSI)
- SCK - Port 52 (SCK)
- MISO - Port 50 (MISO)

Digital Temperatur Sensor (DS18B20):

- Rød ledning - 5V
- Sort ledning - GND
- Gul ledning - 5V → 1KΩ resistor → Port 22

pH Sensor:

- V+ - 5V
- G - GND
- PO - A0
- TO - A1

UV Sensor (VEML6075):

- 3Vo - 3.3V
- GND - GND
- SCL - Port 21 (SCL)
- SDA - Port 20 (SDA)

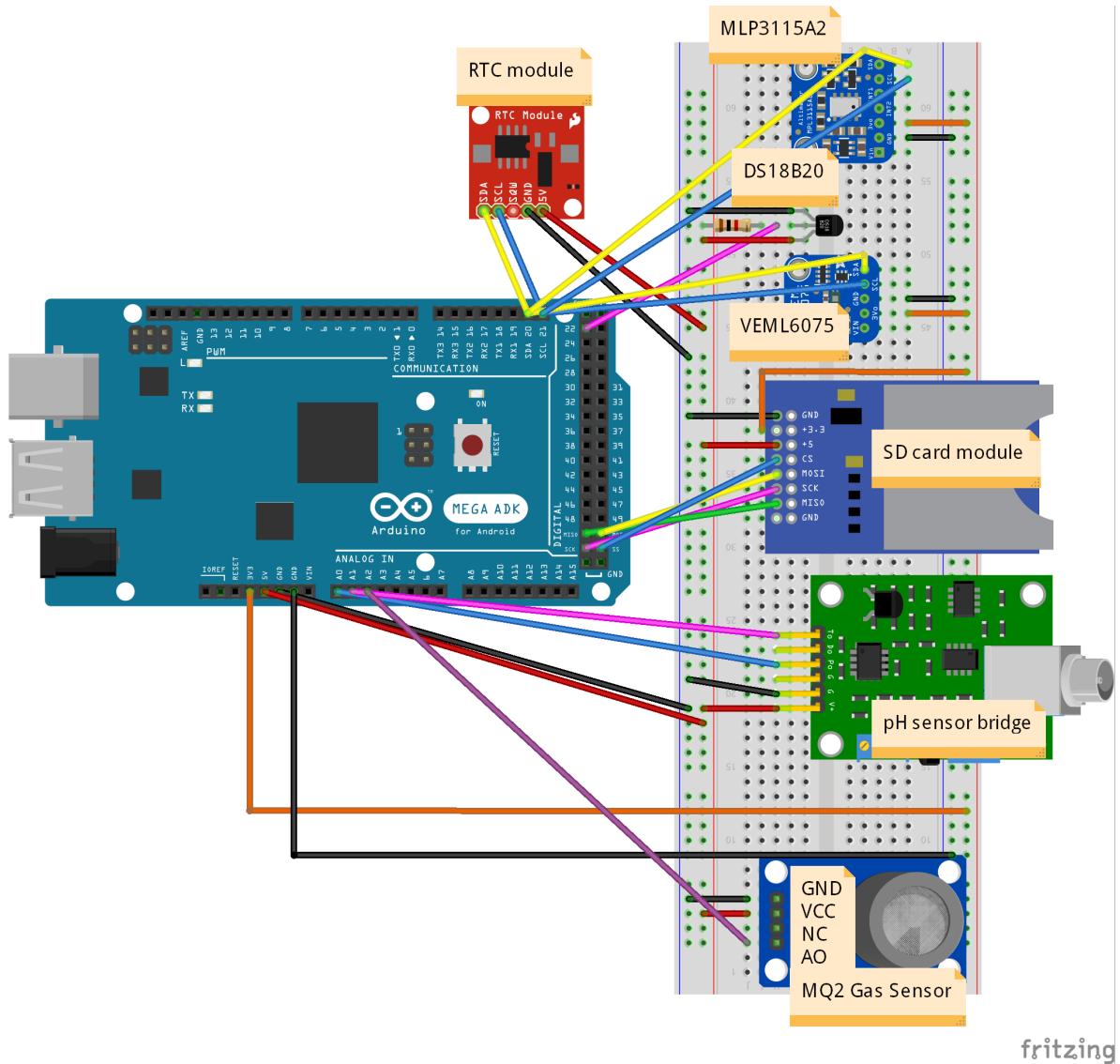
Tryksensor (MLP3115A2):

- 3Vo - 3.3V
- GND - GND
- SCL - Port 21 (SCL)
- SDA - Port 20 (SDA)

MQ2 Gas Sensor:

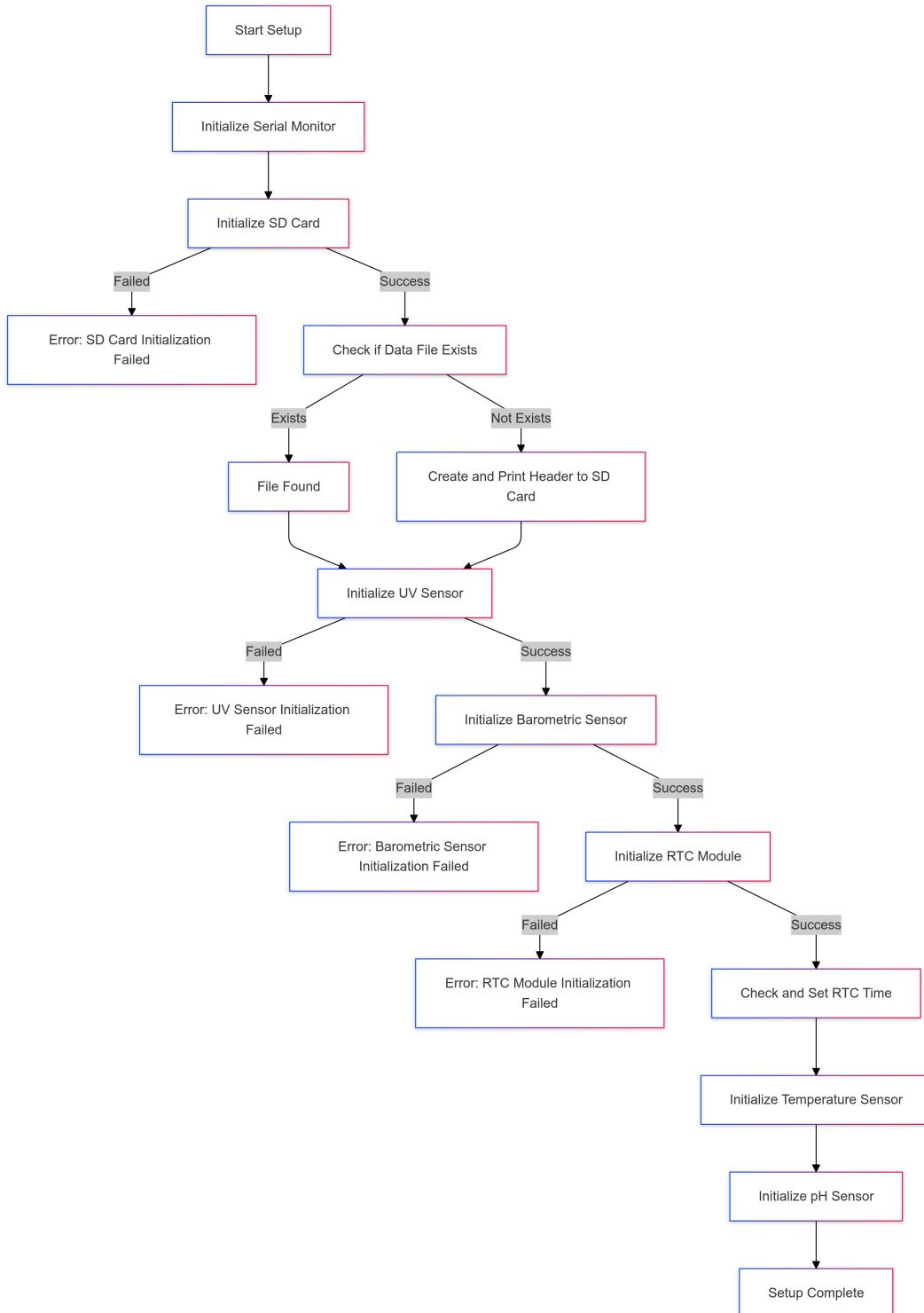
- VCC - 5V
- GND - GND
- NC - N/A
- SIG(AO) - A2

### 1.1.1 Opbygning af datalogger visualiseret

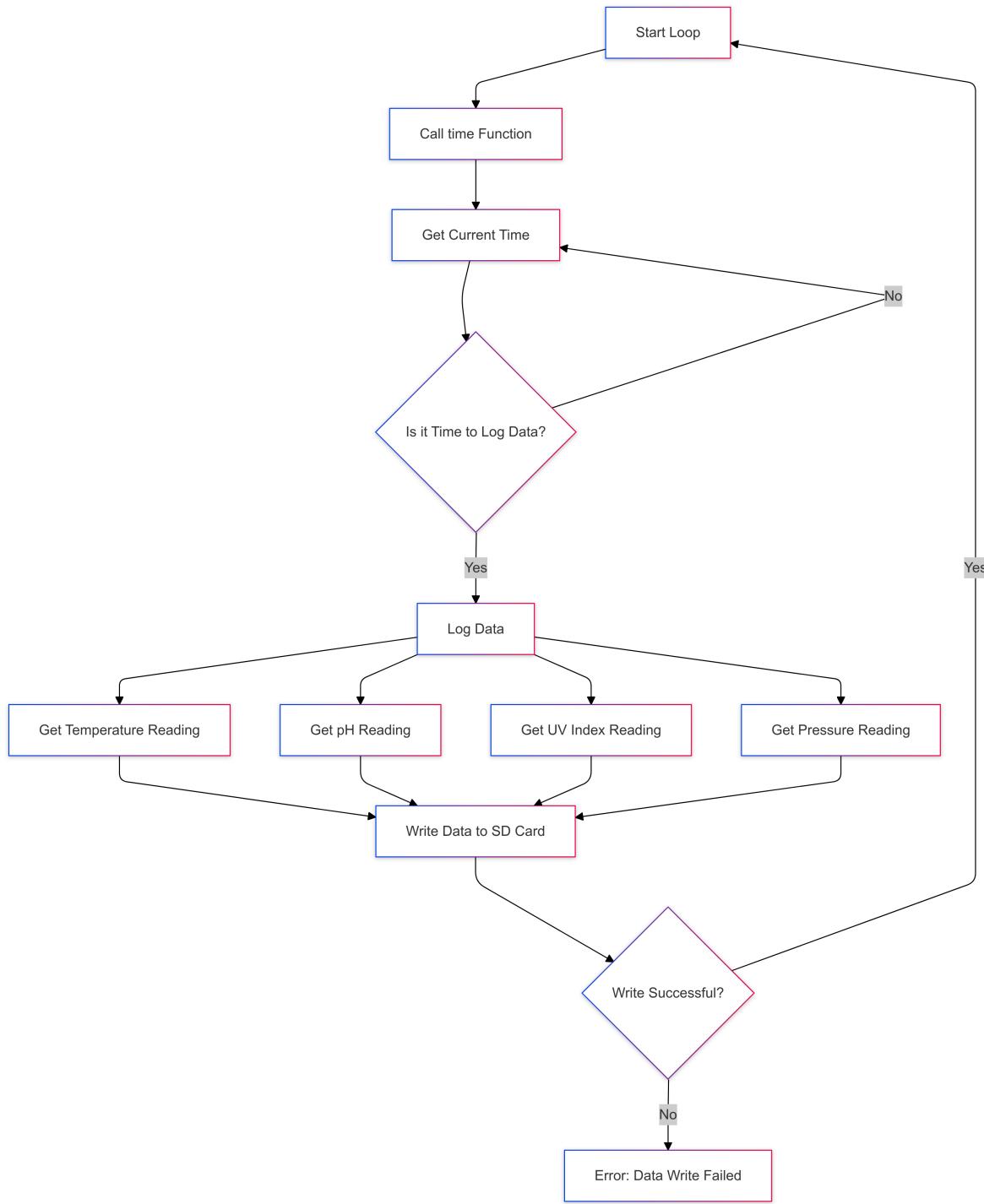


## 2 Kode og Flowcharts

### 2.1 Flowcharts



Figur 3: Flowchart til setup funktionen



Figur 4: Flowchart til loop funktionen

## 2.2 Kode beskrivelse

Til koden er der importeret 9 biblioteker, hvor 7 af dem er til direkte brug af alle modulerne forbundet til Arduinoen.

Lige før setup funktionen bliver der defineret nogle globale variabler. Disse variabler er numrene på de pins, hvor modulerne er forbundet, navn på data filen som skal skrives til, tidsintervallet mellem måling af data, og nogle instanser, som giver nogle funktioner man kan bruge til at kommunikere med modulerne og sensorerne. En instans giver den defineret variabel nogle funktioner, som kan bruges. Det er funktioner, som bliver brugt ofte, og derfor er det lavet til noget, som kan kopieres let. Eksempelvis er der i biblioteket, der hedder "Adafruit\_VEML6075.h" defineret en klasse, som hedder "Adafruit\_VEML6075". Denne klasse har nogle funktioner i sig, og når man laver en instans af dette, hvilket f.eks. er gjort med variablen "uv", så kan man tilgå alle klassens funktioner via variablen.

```
1 Adafruit_VEML6075 uv = Adafruit_VEML6075();
```

Setup funktionen er en funktion, som kun bliver kørt én gang, når man starter Arduinoen eller genstarter koden. I setup funktionen sendes beskeder til alle modulerne som bruges, så de starter op og kan bruges. Et eksempel er, hvor der bliver kaldt "uv.begin()". Som forklaret ovenover, er uv en instans af en klasse med mange funktioner i, og .begin() er en af funktionerne, som kan bruges.

I setup bliver der tjekket om der allerede findes en fil ved det navn, der er defineret. Man kan vælge at slette filen, hvis man gerne vil det. Det er der dog ikke behov for ved denne datalogger. Der bliver tjekket, om der findes en fil ved det defineret navn, fordi der skal tilføje en header linje til data filen, hvis det er tilfældet. Hvis Arduinoen startes op, og der allerede findes en fil ved det defineret navn, betyder det, at der allerede er data i filen. Derfor tilføjes der ikke en header linje. Hvis der ikke allerede er en fil ved det defineret navn, oprettes der en fil, og tilføjes en header linje, hvilket er navnene på dataet i hver kolonne, for at forklare, hvad de forskellige kolonners data er.

```
1 void header() {
2     int attempts = 0;
3     while (!writeToSDCard("Date, Time, Temperature, pH Value, UV
4         Index, Pressure (hPa)\n")) {
5         attempts++;
6         // If the function could not print to the SD card after 10
7         // tries, then it gives up
8         if (attempts > 10) {
9             Serial.println("Could not print the header to the SD card"
10                );
11             return; // Exits the function
12         }
13         delay(500); // Adds a delay of 0.5 sec before retrying
14     }
15     Serial.println("Header printed successfully");
16 }
```

Loop funktionen er en funktion, som Arduinoen kører igen og igen. Fra denne funktion bliver man sendt videre til en anden funktion, som hedder "time". I time funktionen bliver RTC modulet kaldt, for at få tiden givet. Efter dette kører et loop for at vente x minutter, så der er en pause mellem målingerne fra modulerne. Det var muligt at bruge den indbygget delay() funktion til at vente en mængde tid, men det ville ikke være muligt at køre noget andet på Arduinoen imens. Efter tiden er gået, bliver logdata funktionen kaldt, og to tekst værdier bliver givet til funktionen. Det er dato og tid, så det kan skrives ind i filen på SD kortet sammen med dataet.

Til hver af sensor modulerne er der lavet en funktion, som giver værdien for hver af sensorerne (For at se koden, gå ned til modulets beskrivelse nedenunder og se kode eksemplet). Disse værdier bliver sendt videre til writeToSDCard funktionen. Alle værdier fra sensorerne samt dato og tid bliver sat sammen til én stor tekst værdi (String) og sendt videre til writeToSDCard funktionen.

```
1 writeToSDCard(date + " " + time + "," + temperature + "," +
    pHValue + "," + uvValue + "," + pressureValue + "\n");
```

writeToSDCard tager én tekst værdi. Denne tekst værdi bliver sat direkte ind i filen på SD kortet vha. writeToSDCard funktionen. Den starter med at "åbne" SD kortet med ".open()" funktionen, og den forbindelse bliver gemt i en variabel, som gør det muligt at skrive til filen. For flere detaljer om at skrive til et SD kort, se sektion 4.1. Funktionen giver en sandt/falsk værdi tilbage (Boolean værdi), for at gøre det muligt at tjekke om der er skrevet til SD kortet. Den værdi bliver kun brugt i header funktionen, for at prøve at skrive kolonneoverskrifterne igen, hvis den fejler. Når writeToSDCard funktionen er færdig, så går koden bagud og gør logdata funktionen færdig, og time funktionen færdig. Arduinoen er nu tilbage i loop funktionen, som nu også er færdig, hvilket betyder, at den kalder loop funktionen igen. (For at se koden, se filen "Datalogger.ino")

### 2.3 Kode valg

Koden er opdelt i mange funktioner, fordi det gør det lettere at holde styr på hvad de enkelte dele gør. Når der er fejl er det også lidt lettere at finde delene med fejl. Der er lavet mange fejlhåndteringer for at koden kan skrive, hvilke fejl der opstår, og hvor. Den bliver brugt mange biblioteker skrevet af andre. Fordelene med dette er, at man ikke behøver at skrive meget kode selv. Ulempene er dog, at man ikke kan være sikker på, at det virker.

Dataet bliver skrevet til en .csv fil på SD kortet. Det er, fordi en .csv fil kan åbnes i et Excel ark. Hvis man vælger at skrive et program til at behandle dataet, er .csv meget let at arbejde med.

Husk at ændre "delayInMinutes" variablen til det antal minutter I vil have mellem hver måling.

Download the source code: [!\[\]\(147b0c7dce349edf02b6b21226344f99\_img.jpg\)](#)

### 3 RTC Module (Real Time Clock)

RTC er et modul, som kan holde styr på tiden meget præcist. Modulet bruges til at få en tid og en dato til hver måling af sensorerne.

Sensoren har 4 pins der skal forbindes med Arduinoen. 5V, Ground, SDA og SCL. På Arduino Mega er port 20 ment som en SDA port og port 21 er ment som en SCL port. Hvis der er flere sensorer, som har SDA og SCL pins, så kan de alle sammen samles og tilkobles til port 20 og 21.

#### 3.1 Kode

For at RTC modulet kan virke, er der brug for biblioteket, der hedder "RTCLib.h". Herefter defineres i koden, før setup funktionen, hvilke porte SCL og SDA pins er tilkoblet til og der laves en instans af typen "RTC\_DS1307". Herefter initialiseres RTC modulet med en af instansens funktioner, "begin()". Herefter bliver der lavet tests for at se, om modulet virker. For at få tiden og dato'en, kalder man en anden af objektets funktioner, "now()". Den funktion giver en "DateTime" værdi tilbage, som man kan gå ind i og få mere specifikke tidsoplysninger, såsom årstal, minuttal eller sekunder.

```

1 #include <RTCLib.h>
2 #include <Wire.h>
3 #include <SPI.h>
4
5 RTC_DS1307 rtc;
6
7 // RTC config
8 const int rtcSDA = 20;
9 const int rtcSCL = 21;
10
11 void setup() {
12     // Console config
13     Serial.begin(9600);
14
15     // Initializes the rtc module
16     if (!rtc.begin()) {
17         Serial.println("Couldn't find RTC");
18         while (1)
19             ;
20     }
21
22     // Checks the time from the module
23     DateTime now = rtc.now();
24     if (now.year() < 2000) {
25         Serial.println("RTC lost power, let's set the time!");
26         // Following line sets the RTC to the date & time this
27         // sketch was compiled
28         rtc.adjust(DateTime(F(__DATE__), F(__TIME__))); // This is
29             // the line that sets the time to the time the code was
30             // compiled, if you want to change it you can do it here!
31 }
```

```
29 }
30
31 void loop() {
32     DateTime time = rtc.now();
33     // Print YYYY-MM-DD
34     Serial.println(String(time.year()) + "—" + time.month() + "—"
35                   + time.day());
36     // Print HH-MM-SS
37     Serial.println(String(time.hour()) + "—" + time.minute() + "—"
38                   + time.second());
39     delay(1000);
40 }
```

## 4 SD Card Module

SD modulet er det vigtigste modul i hele dataloggeren. Den gør det muligt at skabe en forbindelse mellem Arduinoen og SD-kortet. Dette betyder også, at hvis modulet ikke virker, så vil selve dataloggeren ikke virke. Når den måler data, skal den have et eksternt lager, for at lagre dataet.

Sensoren har 7 pins, der skal forbindes med Arduinoen. 5V, 3.3V (optional), Ground, CS, MOSI, SCK, MISO. På Arduino Mega er porte 50 – 52 reserveret til netop disse funktioner. SCK til 52, MOSI til 51, MISO til 50. CS kan forbindes til hvilken som helst pin på Arduinoen, men koden nedenunder er skrevet således, at CS skal være forbundet til port 53.

Dette modul har skabt mange problemer undervejs i udviklingen. Dette er en tjekliste for at teste modulet, hvis den ikke virker:

- Sørg for at der er nok strøm til Arduinoen, så der vil være nok strøm til modulet
- Sørg for, at ledningerne sidder ordenligt
- Sørg for, at SD kortet fungerer, er formateret korrekt og er sat rigtigt i modulet

Hvis modulet stopper med at fungere efter der er påsat flere sensorer, prøv at flytte SD Kort Modulet over til en tom Arduino. Hvis den ikke virker på den tomme Arduino, betyder det højest sandsynligt, at modulet ikke virker.

### 4.1 Kode

Til SD Kort Modulet er der brug for biblioteket "SD.h". Herefter defineres hvilken port man har koblet CS pin til, variablen er kaldt sdCard i koden nedenunder, og man definerer et navn for filen man vil skrive til på SD kortet, variablen er kaldt dataFileName i koden nedenunder. For at skrive til en fil, skal man bruge en variabel af typen "File", hvilket blev kaldt dataFile. I setup funktionen bliver "SD.open()" kaldt, hvor sdCard variablen bliver brugt til at oprette forbindelse til modulet.

I loop funktionen bliver dataFile sat til output fra "SD.open()", som tager to argumenter. Den ene er filnavnet (dataFileName) og hvilke rettigheder der er behov for til filen (FILE\_READ eller FILE\_WRITE). Hvis det lykkedes at forbinde til SD kortet gennem "open()", så bliver der skrevet til dataFile med ".print()" funktionen. Herefter bruger man ".flush()" til at sende dataet til SD kortet, og lukker SD kortets forbindelse med ".close()". Hvis det ikke lykkedes at forbinde til SD kortet, bliver der skrevet til konsollen, at der er problemer med SD kortet.

```

1 #include <Wire.h>
2 #include <SPI.h>
3 #include <SD.h>
4
5 // SD card config
6 const int sdCard = 53; // This is the pin for
    the SD card reader (CS)
7 const char* dataFileName = "data.dat"; // Name of the file you
    want to save to.

```

```
8 // Variable to write to the SD card
9 File dataFile;
10
11
12 void setup() {
13     // Console config
14     Serial.begin(9600);
15
16     // Initializing the SD card reader
17     if (!SD.begin(sdCard)) {
18         Serial.println("initialization failed!");
19         while (1)
20             ;
21     }
22     Serial.println("initialization done.");
23 }
24
25 void loop() {
26     // Establishes connection to the SD card
27     dataFile = SD.open(dataFileName, FILE_WRITE);
28     // Check if file opened successfully
29     if (dataFile)
30     {
31         dataFile.print("Test\n");
32         Serial.println("Data written to the file.");
33         dataFile.flush(); // Flush the data to the SD card
34         // Close the file
35         dataFile.close();
36     } else {
37         Serial.println("Error opening the file.");
38     }
39     delay(1000);
40 }
```

## 5 Digital temperatur sensor

En digital temperatur måler bliver brugt til at måle temperaturen. En analog temperatur måler kunne ikke bruges, fordi det var ikke muligt at kalibrere en analog temperaturmåler, således den ville give en korrekt måling, hvilket er grunden til at en digital temperatur måler bliver brugt.

Sensoren har 3 pins. En rød 5V, en sort GND og en gul data pin, hvilket er forbundet til 5V via en  $1\text{K}\Omega$  resistor.

### 5.1 Kode

DS18B20 temperatur sensoren skal bruge to biblioteker, "OneWire.h" og "DallasTemperature.h". I koden defineres en OneWire instans med porten for data pin forbundet til Arduinoen. Herefter bruges en reference til OneWire instansen i en DallasTemperature instans. Denne variabel er kaldt "sensors".

I setup funktionen bruges ".begin()" metoden på "sensors". I loop funktionen bruges ".requestTemperatures()" metoden fra "sensors", som får temperatursensoren til at tage en måling. Herefter kan man kalde ".getTempCByIndex(0)" til at få værdien på målingen, så længe man kun har én temperatursensor koblet på den ene port.

```

1 #include <OneWire.h>
2 #include <DallasTemperature.h>
3 #include <SPI.h>
4 #include <Wire.h>
5
6 // DS18B20 config
7 const int tempDigitalPin = 22;           // Data pin where the
   temperature sensor is connected
8 OneWire oneWire(tempDigitalPin);         // Setup a oneWire
   instance to communicate with any OneWire devices
9 DallasTemperature sensors(&oneWire);    // Pass our oneWire
   reference to Dallas Temperature sensor
10
11 void setup() {
12     // Console config
13     Serial.begin(9600);
14     // Initializes the digital temperature sensor
15     sensors.begin();
16 }
17
18 void loop() {
19     // Getting temperature from sensor
20     sensors.requestTemperatures();
21     float tempC = sensors.getTempCByIndex(0);
22
23     Serial.println(String("Temperature in Celsius: ") + tempC);
24     delay(1000);
25 }
```

## 6 pH Sensor

Som pH modul blev der brugt en 'PH4502C'. Modulet har tilkoblet en sonde, som mäter pH i en væske.

På sensoren blev der tilkoblet 4 pins til Arduinoen. V+ til 5V, G til GND, PO til A0 og TO til A1. PO giver et analog output man kan omregne til en pH værdi. TO giver et analog output til at omregne til en temperatur. Temperaturmålingerne giver dog ikke korrekte målinger, og derfor bliver der brugt en digital temperaturmåler.

### 6.1 Kode

Til pH modulet hører et bibliotek med, som hedder "ph4502c\_sensor.h". Før setup funktionen defineres en instans af biblioteket, hvor man angiver portene, som PO og TO er koblet til. I setup starter man forbindelsen til modulet via ".init()" funktionen fra instansen. I loop kan man bruge ".read\_ph\_level()" fra instansen for at få pH værdien.

```
1 #include <Wire.h>
2 #include <SPI.h>
3 #include <ph4502c_sensor.h>
4
5 // PH sensor config
6 const uint8_t PO = A0;
7 const uint8_t TO = A1;
8 PH4502C_Sensor ph4502c(PO, TO);
9
10 void setup() {
11     // Console config
12     Serial.begin(9600);
13     // Initialize the PH sensor
14     ph4502c.init();
15 }
16
17 void loop() {
18     // Getting pH value from sensor
19     float pH = ph4502c.read_ph_level();
20     Serial.println(String("pH value is: ") + pH);
21     delay(1000);
22 }
```

## 7 UV sensor

En Adafruit VEML6075 bliver brugt som UV sensoren til dataloggeren.

Der er forbundet 4 pins fra sensoren til Arduinoen. 3Vo til 3.3V, GND til GND, SCL til SCL porten og SDA til SDA porten. På Arduino Mega er port 20 ment som en SDA port og port 21 er ment som en SCL port. Hvis der er flere sensorer, som har SDA og SCL pins, så kan de alle sammen samles og tilkobles til port 20 og 21.

### 7.1 Kode

Adafruit har lavet et bibliotek til Arduino til denne sensor, som hedder "Adafruit\_VEML6075.h". Før setup funktionen definerer man en instans af en type defineret i biblioteket. I setup funktionen bliver den instans brugt til at skabe en forbindelse mellem Arduinoen og sensoren via ".begin()" funktionen. I loop kan man få UV indexet ved at kalde ".readUVI()" funktionen til instansen.

```
1 #include <Adafruit_VEML6075.h>
2 #include <Wire.h>
3 #include <SPI.h>
4
5 Adafruit_VEML6075 uv = Adafruit_VEML6075();
6
7 void setup() {
8     // Console config
9     Serial.begin(9600);
10    // Initializing UV sensor
11    if (!uv.begin()) {
12        Serial.println("Failed to communicate with VEML6075 sensor,
13                      check wiring?");
14        while (1) { delay(100); } // Infinite loop
15    }
16
17 void loop() {
18     // Getting uv index from sensor
19     float uv_value = uv.readUVI();
20     Serial.println(String("The UV index is: ") + uv_value);
21     delay(1000);
22 }
```

## 8 Tryksensor

Den valgte tryksensoren kan både bruges som en barometrisk tryk sensor og som en højdemåler. Dog for at højdemålinger fra denne sensor skal passe, skal sensoren kalibreres hele tiden. Sensoren, som bliver brugt, er en Adafruit MPL3115A2 sensor.

Sensoren har 4 pins forbundet til Arduinoen. 3Vo til 3.3V, GND til GND, SCL til SCL porten og SDA til SDA porten. På Arduino Mega er port 20 ment som en SDA port og port 21 er ment som en SCL port. Hvis der er flere sensorer, som har SDA og SCL pins, så kan de alle sammen samles og tilkobles til port 20 og 21.

### 8.1 Kode

Der bruges et bibliotek fra Adafruit til denne sensor, som hedder "Adafruit\_MPL3115A2.h". Før setup funktionen defineres en instans af et objekt fra biblioteket. I setup funktionen bliver der skabt en forbindelse mellem sensoren og Arduinoen med ".begin()" funktionen gennem den defineret instans. Herefter kan man sætte en værdi for trykket ved havniveau, hvis man vil måle højden. Denne værdi skal dog opdateres en gang imellem, hvis man vil måle den rigtige højde. I loop funktionen bliver funktionen ".getPressure()" brugt fra instansen, hvor den giver trykket som output i hPa.

```

1 #include <Adafruit_MPL3115A2.h>
2 #include <Wire.h>
3 #include <SPI.h>
4
5 // Pressure sensor
6 Adafruit_MPL3115A2 baro;
7
8 void setup() {
9     // Console config
10    Serial.begin(9600);
11    // Initializing barometric sensor
12    if (!baro.begin()) {
13        Serial.println("Could not find sensor. Check wiring.");
14        while(1);
15    }
16    baro.setSeaPressure(1013.26);
17}
18 void loop() {
19     // Getting pressure value from sensor
20     float pressure = baro.getPressure();
21     Serial.println(String("The pressure is: ") + pressure + "hPa")
22     ;
23 }
```

## 9 Bilag

## 10 Koden

```

1 #include <Wire.h>
2 #include <SPI.h>
3 #include <RTClib.h>
4 #include <SD.h>
5 #include <OneWire.h>
6 #include <DallasTemperature.h>
7 #include <ph4502c_sensor.h>
8 #include <Adafruit_VEML6075.h>
9 #include <Adafruit_MPL3115A2.h>

10 RTC_DS1307 rtc;

11 // RTC config
12 const int rtcSDA = 20;
13 const int rtcSCL = 21;
14 // We desided to take a reading every 10 minutes, but if you
15 // want to change it this is the place to do it!
16 const int delayInMinutes = 10;

17 // SD card config
18 const int sdCard = 53; // This is the pin for
19 // the SD card reader (CS)
20 const char* dataFileName = "data.csv"; // Name of the file
21 // you want to save to.
22 File dataFile;

23 /* Analog pin for analog temperature sensor (Couldn't get it
24 to work)
25 const int termistorPin = A0;
26 */

27 // Digital temperature sensor config (DS18B20)
28 const int tempDigitalPin = 22; // Data pin connection
29 // for sensor
30 OneWire oneWire(tempDigitalPin); // Setup a oneWire
31 // instance to communicate with any OneWire devices
32 DallasTemperature tempSensors(&oneWire); // Pass our oneWire
33 // reference to Dallas Temperature sensor

34 // PH sensor config
35 const uint8_t P0 = A0;
36 const uint8_t T0 = A1;
37 PH4502C_Sensor ph4502c(P0, T0);

38 // UV sensor
39 Adafruit_VEML6075 uv = Adafruit_VEML6075();

```

```
40
41 // Pressure sensor
42 Adafruit_MPL3115A2 baro;
43
44 // Gas sensor A2 pin
45 const uint8_t gasSensorPin = A2;
46
47 void setup() {
48     Serial.begin(9600);
49
50     if (!SD.begin(sdCard)) {
51         Serial.println("initialization failed!");
52         while (1)
53             ;
54     }
55     if (!SD.exists(dataFileName)) {
56         // Prints the header to the sd card
57         header();
58     }
59     Serial.println("initialization done.");
60
61     // Initializing UV sensor
62     if (!uv.begin()) {
63         Serial.println("Failed to communicate with VEML6075 sensor
64             , check wiring?");
65         while (1) { delay(100); }
66     }
67     // Initializing barometric sensor
68     if (!baro.begin()) {
69         Serial.println("Could not find sensor. Check wiring.");
70         while(1);
71     }
72     baro.setSeaPressure(1013.26);
73
74     // Initializes the rtc module
75     if (!rtc.begin()) {
76         Serial.println("Couldn't find RTC");
77         while (1)
78             ;
79     }
80
81     DateTime now = rtc.now();
82     if (now.year() < 2000) {
83         Serial.println("RTC lost power, let's set the time!");
84         // Following line sets the RTC to the date & time this
85         // sketch was compiled
86         rtc.adjust(DateTime(F(__DATE__)), F(__TIME__)); // This
87         // is the line that sets the time to the time the code was
88         // compiled, if you want to change it you can do it here!
89     }
90 
```

```
86
87     // Initializes the digital temperature sensor
88     tempSensors.begin();
89
90     // Initialize the PH sensor
91     ph4502c.init();
92 }
93
94 void loop() {
95     time();
96 }
97
98 void time() {
99     DateTime now = rtc.now();
100
101     int startTime = now.minute();
102     while ((startTime + delayInMinutes) % 60 != now.minute()) {
103         now = rtc.now(); // Update the current time
104         Serial.println("Waiting...");
105         delay(1000);
106     }
107
108     // Keeps the time and date in separate strings
109     String date = String(now.year()) + "/" + now.month() + "/" +
110         now.day();
111     String time = String(now.hour()) + ":" + now.minute() + ":" +
112         now.second();
113     // Uses the time and date as arguments in the logdata
114     // function
115     logdata(date, time);
116 }
117
118 void logdata(String date, String time) {
119     Serial.println("Logging data...");
120     // Gets data from temperature sensor and pH sensor
121     float temperature = temp();
122     float pHValue = pH();
123     float uvValue = UV();
124     float pressureValue = pressure();
125     /* For debugging
126     Serial.print("Temperature: ");
127     Serial.println(temperature);
128     Serial.print("pH Value read: ");
129     Serial.println(pHValue);
130     Serial.print("UV Index Reading: ");
131     Serial.println(uvValue);
132     Serial.print("Pressure: ");
133     Serial.println(pressureValue);
134     Serial.print("Time: ");
135     Serial.println(time);
```

```
133  /*
134
135 // .csv format
136 writeToSDCard(date + ";" + time + ";" + temperature + ";" +
137   pHValue + ";" + uvValue + ";" + pressureValue + "\n");
138   // Can add Humidity, Soil Moisture, Light Intensity,
139   Water Level
140 }
141
142 bool writeToSDCard(String data) {
143   dataFile = SD.open(dataFileName, FILE_WRITE);
144   Serial.println(data);
145   // Check if the file opened successfully
146   if (dataFile) {
147     // Write data to the file
148     dataFile.print(data);
149     Serial.println("Data written to the file.");
150     dataFile.flush(); // Flush the data to the SD card
151     // Close the file
152     dataFile.close();
153     return true;
154   } else {
155     Serial.println("Error opening the file.");
156     Serial.print("Filename: ");
157     Serial.println(dataFileName);
158     return false;
159   }
160 }
161 // This prints the header to the SD card
162 void header() {
163   int attempts = 0;
164   while (!writeToSDCard("Date;Time;Temperature;pH Value;UV
165   Index;Pressure (hPa)\n")) {
166     attempts++;
167     // If the function could not print to the SD card after 10
168     // tries, then it gives up
169     if (attempts > 10) {
170       Serial.println("Could not print the header to the SD
171       card");
172       return; // Exits the function
173     }
174     delay(500); // Adds a delay of 0.5 sec before retrying
175   }
176   Serial.println("Header printed successfully");
177 }
178
179 float temp() {
180   // Failed analog temperature sensor
181   /*
182   // Microvolts input from the termistor
```

```
177     int resistance = analogRead(termistorPin);
178     // Conversion to Celsius
179     float temperature = (-0.1637931 * resistance) + 169.67;
180     Serial.println(resistance);
181     Serial.println(temperature);
182     */
183
184     // Digital temp sensor
185     tempSensors.requestTemperatures();
186     float tempC = tempSensors.getTempCByIndex(0);
187
188     return tempC;
189 }
190
191 float pH() {
192     /* Failed temperature sensor on pH sensor (T0 pin)
193     float temp = ph4502c.read_temp();
194     Serial.println("pH temp: " + temp);
195     End test */
196
197     float pH = ph4502c.read_ph_level();
198     return pH;
199 }
200
201 float UV() {
202     return uv.readUVI();
203 }
204
205 float pressure() {
206     return baro.getPressure();
207 }
```