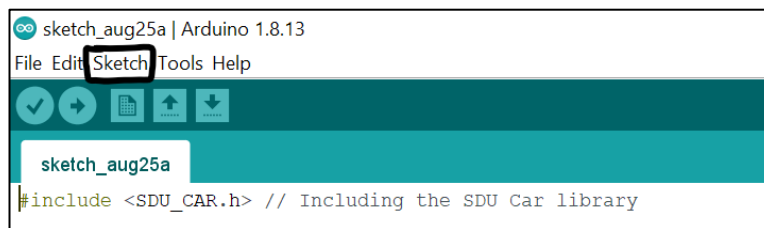


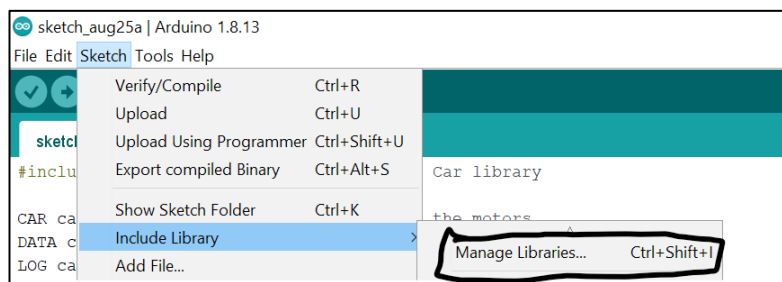
Brug af SDU CAR Biblioteket

Inkluder SDU_CAR library:

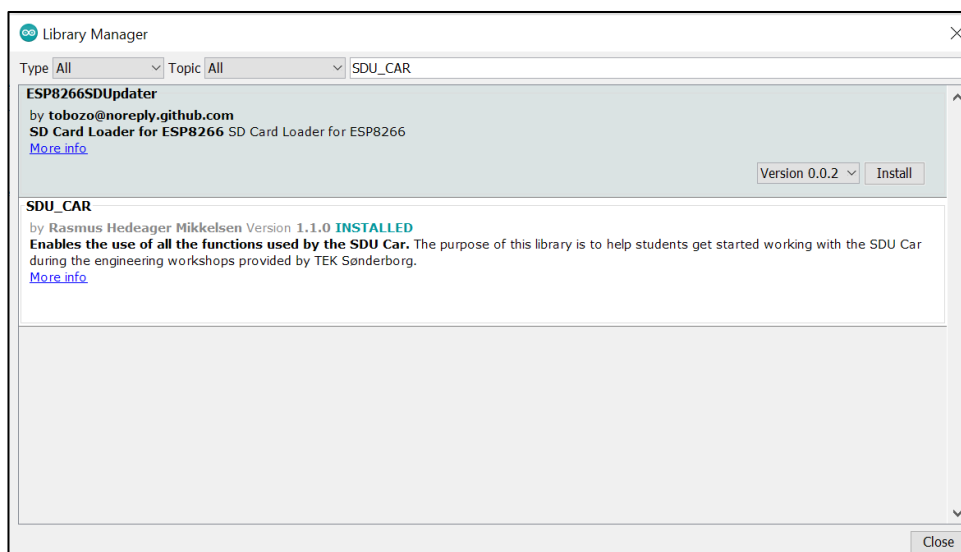
Klik på sketch:



Klik på Manage libraries:

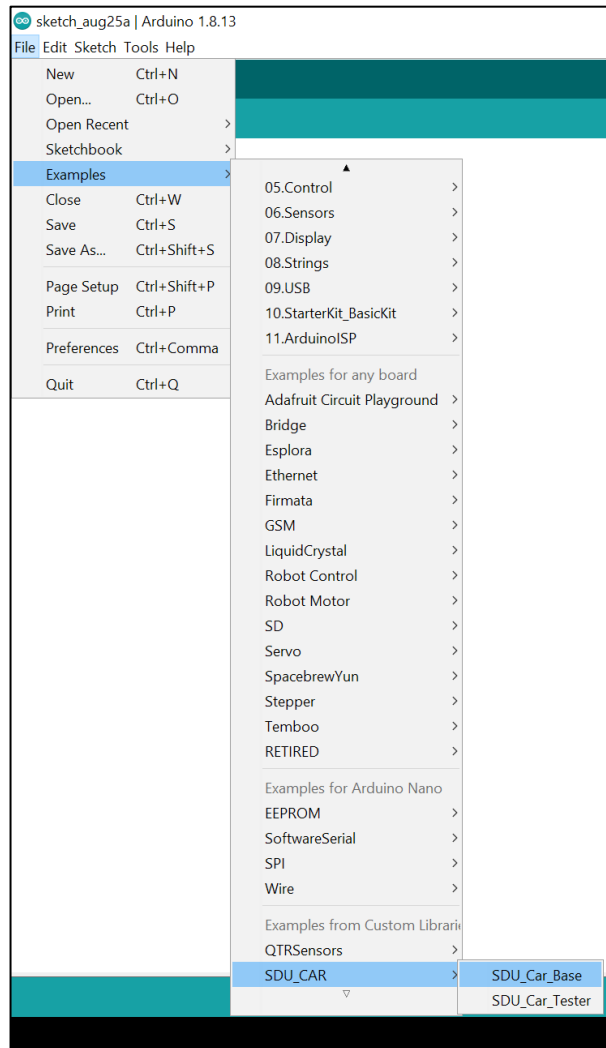


Søg efter SDU_CAR og tryk installer:



Åben en opstartskode:

Dette sikrer at biblioteket bliver indlæst korrekt. Gå nu ind i filer, examples og rul helt ned i bunden hvor du finder Examples from custom Library og vælg SDU_car_Base.



Hvordan bruger man koden?

Indstil hastigheden på hvert hjul

- Man kan indstille hastigheden fra -100 til 100
- Til venstre for komma er det venstre hjul
- Til højre for komma er det højre hjul

Eksempel på brug:

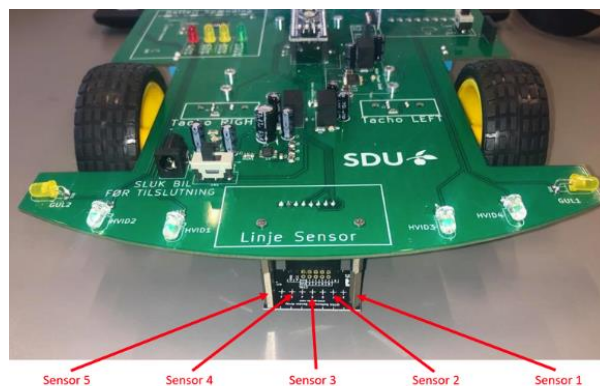
```
car.setCarSpeed(75, -75);
```

Dette sætter hastigheden på det hjulet til venstre til 75 og det højre til -75 (Bemærk, motor kører kun når batteri kontakten står på "ON")

Aflæs linje sensor

Når man kigger på bilen hvor den vil køre væk fra dig, hvis den kørte fremad, er numrene på sensoren i læse rækkefølge. Det vil sige helt ude til venstre er sensor 1 så 2 osv.

På billedet under kører bilen mod dig og derfor er det omvendt rækkefølge:



For så at aflæse sensoren og gemme den i en variabel kan følgende kode bruges:

```
cardata.readLineSensor();  
int sensor4 = cardata.getLineSensor(4);
```

Her bliver sensor 4 aflæst, hvis man vil aflæse sensor 1 ændre du blot 4-tallet til et 1-tal. Du vil modtage en værdi der skal kalibreres så der kan kendes forskel på hvidt gulv og sort baggrund.

Kalibrering af sensor:

```
1 #include <SDU_CAR.h>
2
3 // Sets up the SDU Car library.
4 CAR car;
5 DATA cardata;
6 LOG carlog;
7
8
9 void setup() {
10     Serial.begin(9600); // Communication with the computer.
11
12     cardata.begin(); // Enables reading of sensors!
13 }
14
15
16 void loop() {
17
18     // Reads the line follower sensor.
19     cardata.readLineSensor();
20
21     int sensor1 = cardata.getLineSensor(1); // Gets the value from sensor 1.
22
23     Serial.println(sensor1);
24
25 }
```

Opstart af bibliotek:

For at bruge biblioteket skal man importere biblioteket i sin sketch, dette gøres ved at skrive følgende linjer i toppen af koden:

```
#include <SDU_CAR.h> // Including the SDU Car library

CAR car; // Initializing the controls for the motors
DATA cardata; // Initializing the sensors
LOG carlog; // Initializing the SD Card communication
```

I din setup skal du inkludere følgende to linjer kode, disse gøre det muligt at skrive til SD Kort samt læse diverse sensore:

```
void setup() {

    car.begin(); // Initializing the motors and latch
    cardata.begin(); // Initializing tachometer and communication with MEMS
    carlog.begin(); // Initializing communication with the SD Card

}
```

Brug af biblioteket:

Få bilen til at gøre noget:

Indstilling af hastigheden på bilen:

Dette er funktionen i biblioteket. Ikke hvordan man bruger den (Husk dette når du læser dokumentationen!)

```
void CAR::setCarSpeed(int left_speed, int right_speed) // Controls the motor speed
```

Tager to parametre som input. Venste- og højre hastighed. Hastigheden er i procent, det vil sige at fremad er plus, og bagud er minus. Det vil sige at du kan sætte det venstre hjul til at køre 75% og højre til -75% for at dreje rundt på stedet.

Eksempel på brug:

```
car.setCarSpeed(75, -75) // Notice that you need to call the class 'car' in order to
                          // use this function
```

Brug af sensorer:

Brug af accelerometer:

```
void DATA::readAccel() // Reads the accelerometer and stores the values for later use

float DATA::getAccel(accel_data_dir_t dir) // Returns the value of the accelerometer
                                           // in the specified direction
```

Når du skal aflæse værdien af accelerometeret skal du først læse dataen fra MEMS sensoren, herefter kan du hente de individuelle akser ned ved efterfølgende at kalde funktionen getAccel().

Eksempel på brug:

```
cardata.readAccel(); // Reads the data from the MEMS sensor, required!

float xAxis = cardata.getAccel(x); // Returns the data from the x axis,
                                   // parameters: [x, y, z]
```

Brug af tachometer:

```
unsigned int DATA::getTachoLeft(void) // Returns no. of ticks on the left tachometer

unsigned int DATA::getTachoRight(void) // Returns no. of ticks on the right tachometer

float DATA::getDistLeft(void) // Returns distance driven on left wheel in meters

float DATA::getDistRight(void) // Returns distance driven on right wheel in meters

void DATA::resetTacho(void) // Resets both tacho counters, including distance driven
```

Brug den eller de funktioner som er nødvendige. Skal du tracke at du køre baglæns skal dette gøres ved at nulstille tachometeret og huske at den kørte distance er baglæns, da tachometeret vil tælle op uanset retning.

Eksempel på brug:

```
unsigned int leftWheelTicks = cardata.getTachoLeft(); // Set the variable to the number
                                                       // of ticks

float distanceDriven = cardata.getDistRight(); // Sets the variable to the distance
                                               // driven on the right wheel

cardata.resetTacho();
```

Aflæsning af batterispænding:

```
float DATA::getBatteryVoltage(void) // Returns the battery voltage
```

Funktionen returnerer batterispændingen.

Eksempel på brug:

```
float batteryVoltage = cardata.getBatteryVoltage(); // Set the variable to the battery  
// voltage
```

Aflæsning af linje sensor:

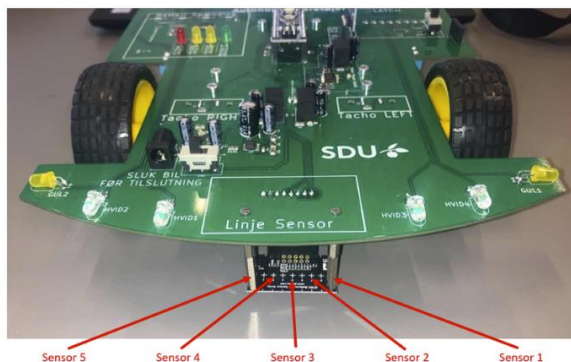
```
void DATA::getLineSensor() // Reads the line sensor and stores the values for later use  
  
int DATA::getLineSensor(char sensor_number) // Returns the value of the sensor based  
// on the sensor number: [1, 2, 3, 4, 5]
```

Når du skal aflæse værdien af fra linje sensoren, skal du først læse dataen fra sensoren, dette gør du med kaldet, readLineSensor(), herefter kan du hente de individuelle sensorværdier ned, ved at kalde getLineSensor(x).

Eksempel på brug:

HUSK AT KALDE LÆSEFUNKTION, ELLERS VIRKER DET IKKE!

```
cardata.readLineSensor(); // Read the line sensor  
  
int sensor4 = cardata.getLineSensor(4); // Set the var to the value of line sensor 4  
// to the variable
```

Her ses positionen af line sensors:

Aflæsning af tid:

```
float DATA::t(void) // Returns the time in seconds
```

Funktionen returnerer tiden der er gået siden arduinoen blev tændt i sekunder

Eksempel på brug:

```
float secsSinceBoot = cardata.t(); // Set the var time in seconds since Arduino boot
```

Logning af data til SD Kort

```
void LOG::log(const String& logdata) // Logs data parsed in the function to the SD Card
```

Når der skal logges data til SD Kort skal det gøres på en meget bestemt måde. Det foregår via en String(tekststreng). Fuldstændig som hvis du ville lave en serial print:

```
Serial.println("Hello World!");
```

Hvad der står mellem paranteserne er hvad der bliver skrevet i konsollen. På samme måde vil der blive logget til SD Kortet:

```
carlog.log("Hello World!");
```

Ønsker du at logge værdier fra en sensor på sd kortet skal disse konverteres til en streng, dette kan gøres således:

```
String(value, numberOfDigits) // Converts a numerical value to a string
```


Hvor value er den værdi du ønsker at lave til en streng, og numberOfDigits er antallet af cifre du ønsker efter kommaet. Nu kan værdien af sensoren bruges sammen med tekst i din log:

```
float distanceDriven = cardata.getDistRight(); // Reading the distance driven

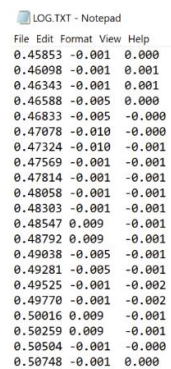
carlog.log("This is distance driven: " + String(distanceDriven, 2) + "cm");

### Output to SD Card: This is distance driven: 420,69cm
```

Skal dataen bruges til databehandling kan man bruge "\t" mellem forskellige data, så kan computeren selv opdele data efterfølgende:

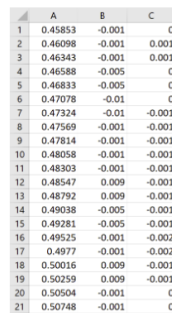
```
carlog.log(String(data1, 2) + "\t" + String(data2, 2) + "\t" + String(data3, 2));
```

Eksempel på hvordan en logfil ser ud lavet på denne måde:



```
LOG.TXT - Notepad
File Edit Format View Help
0.45853 -0.001 0.000
0.46098 -0.001 0.001
0.46343 -0.001 0.001
0.46588 -0.005 0.000
0.46833 -0.005 -0.000
0.47078 -0.010 -0.000
0.47324 -0.010 -0.001
0.47569 -0.001 -0.001
0.47814 -0.001 -0.001
0.48058 -0.001 -0.001
0.48303 -0.001 -0.001
0.48547 0.009 -0.001
0.48792 0.009 -0.001
0.49038 -0.005 -0.001
0.49281 -0.005 -0.001
0.49525 -0.001 -0.002
0.49770 -0.001 -0.002
0.50016 0.009 -0.001
0.50259 0.009 -0.001
0.50504 -0.001 -0.000
0.50748 -0.001 0.000
```

Og efterfølgende kopieret in i Excel:



	A	B	C
1	0.45853	-0.001	0
2	0.46098	-0.001	0.001
3	0.46343	-0.001	0.001
4	0.46588	-0.005	0
5	0.46833	-0.005	0
6	0.47078	-0.01	0
7	0.47324	-0.01	-0.001
8	0.47569	-0.001	-0.001
9	0.47814	-0.001	-0.001
10	0.48058	-0.001	-0.001
11	0.48303	-0.001	-0.001
12	0.48547	0.009	-0.001
13	0.48792	0.009	-0.001
14	0.49038	-0.005	-0.001
15	0.49281	-0.005	-0.001
16	0.49525	-0.001	-0.002
17	0.4977	-0.001	-0.002
18	0.50016	0.009	-0.001
19	0.50259	0.009	-0.001
20	0.50504	-0.001	0
21	0.50748	-0.001	0

Skal tiden der er gået logges kan du bruge følgende:

```
carlog.log(String(cardata.t(), 4) + "\t" + String(data, 2)); // Logs the first column as time
// in seconds, followed by data
```