CAMPUS 02 GRAZ
FACHHOCHSCHULE DER WIRTSCHAFT

CAN FAHRZEUG DATENLOGGER

Florian Strauß
Lukas Stocker
Severin Gruber

Inhalt

- CAN-Bus
- OBD-2 Schnittstelle
- Hardware Aufbau CAN-Fahrzeuglogger
- Software CAN-Fahrzeuglogger

CAN BUS Grundlagen

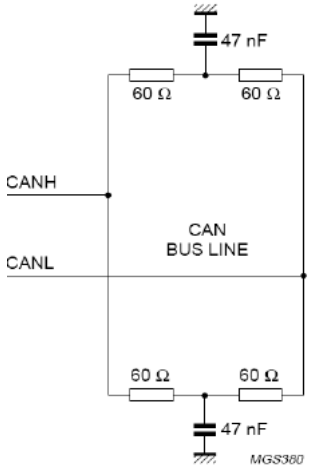
Eigenschaften Bus-System

- Bussystem für Automotive-Bereich und Industrieanwendungen
- Gleichwertige Busteilnehmer
- Übertragungsrate 125-500kBit/s
- Buszugriff wird immer 1 Teilnehmer gewährt
  - Bitweise Arbitrierung
  - Keine Taktleitung notwendig, Taktinformation über Bit Stuffing

CAN BUS Grundlagen

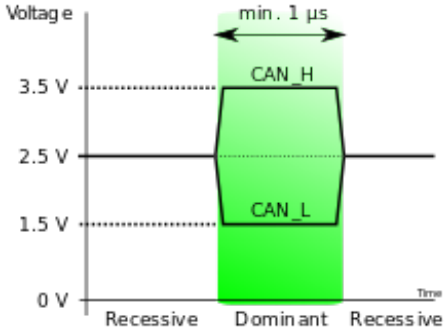
Physikalischer Aufbau

- „Wired And“ Verknüpfung
- 2 Leitungen: CAN High, CAN Low
  - Abschlusswiderstände 120 Ω
  - „Split Termination“



Buspegel

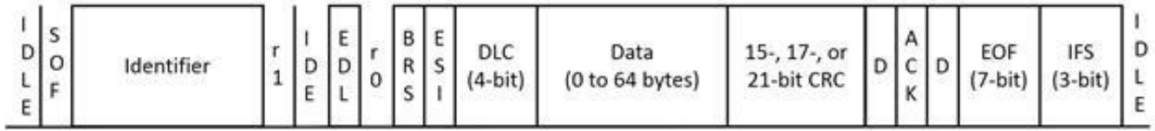
- Logisch „0“: Dominant
  - UCAN_H - UCAN_L > 0.9 V
- Logisch „1“: Rezessiv
  - UCAN_H - UCAN_L < 0.5 V



CAN BUS Grundlagen

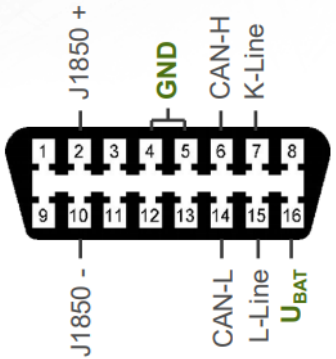
Nachrichtenprotokoll (Bsp. „CAN FD“)

- Identifier (11 Bit bzw. 29 Bit in der „Extended“ Version)
- Error Frame
- Data Frame (Datenanforderungsrahmen, 0 bis 64 bytes)
- Overload Frame
- Interframe Space (zur Trennung zweier Rahmen)



OBD-2 Schnittstelle

- On-Board-Diagnose System für Fahrzeuge
- Zugriff auf Informationen von abgasrelevanten Bauteilen und anderen Elektronikkomponenten
- Genormter 16 poliger Stecker nach ISO 15031-3 / SAE J1962
- Unterschiedliche Protokolle → häufig CAN



OBD Stecker, Quelle: http://www.emotive.de/documents/WebcastsProtected/Transport-Diagnoseprotokolle.pdf

OBD Protokoll

- Baudrate von 250 oder 500 kbit/s
- Unterschiedliche Diagnose Requests („Test Modes“, SID)
- Tester sendet Anfrage an Adresse 0x7DF
- Angefragter Wert steht im PID Feld (z.B. 0x09 für Motorlast)
- Steuergerät antwortet mit angefragten Wert

Beispiel Abfrage aktuelle Motorlast

Anfrage vom Tester:

CAN ID	OBD SID	OBD PID
7DF	**01**	**09**
Diagnose Request	Abfrage von Messdaten	Motorlast

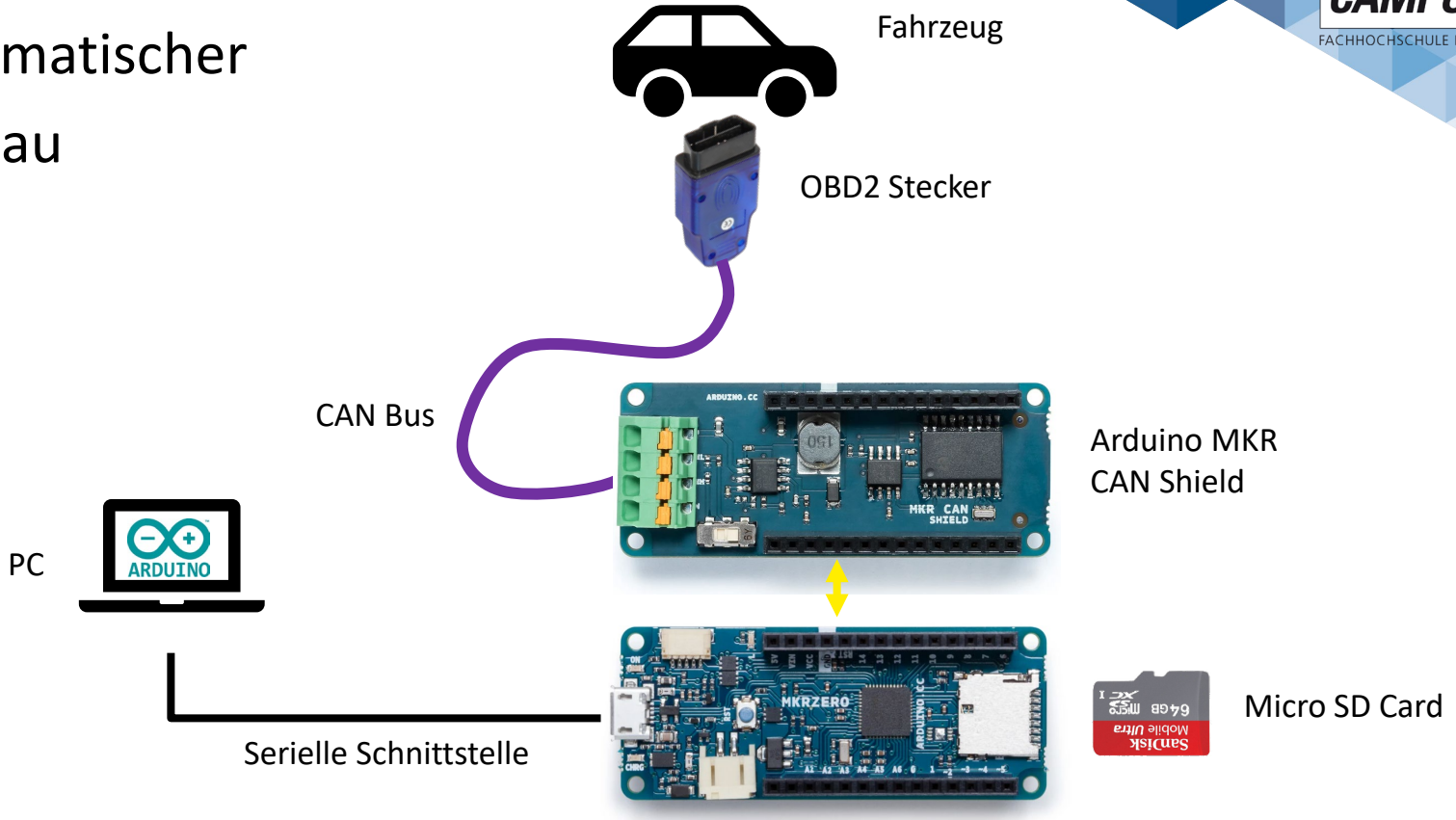
Antwort vom Steuergerät:

CAN ID	OBD SID	OBD PID	OBD Data
7E8	**41**	**09**	**A7**
Adresse des Testers	SID + 0x40	Motorlast	Aktueller Wert

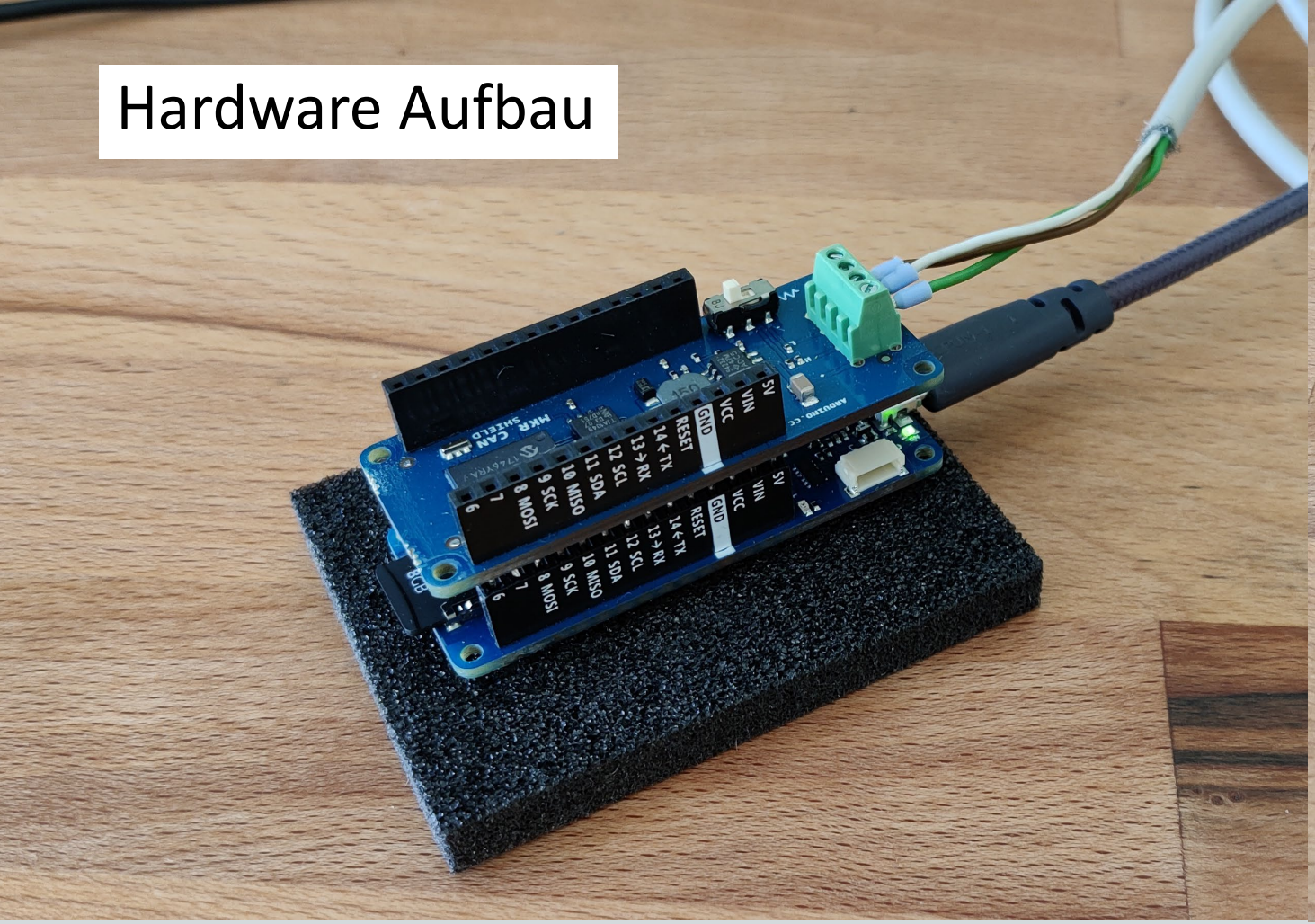
Berechnung:

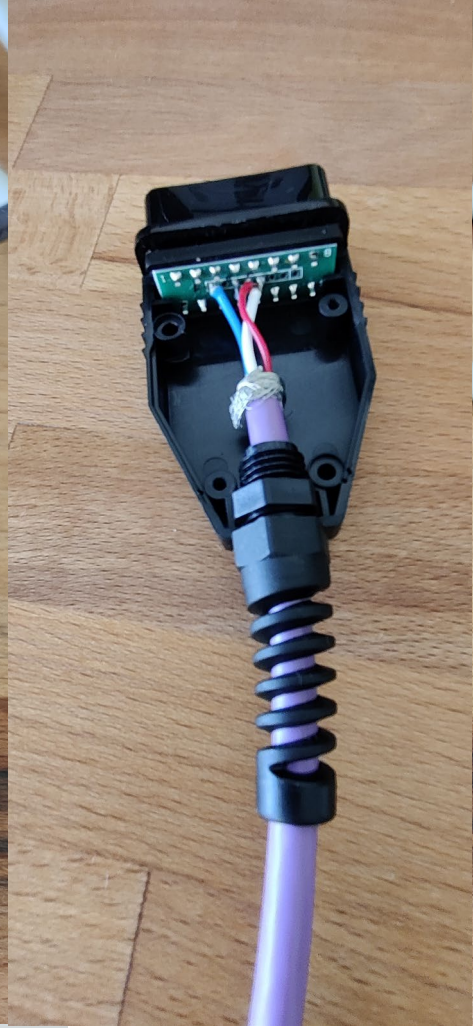
$$Motorlast = \frac{0xA7}{256} * 100 = 65,23\ \%$$

Schematischer Aufbau

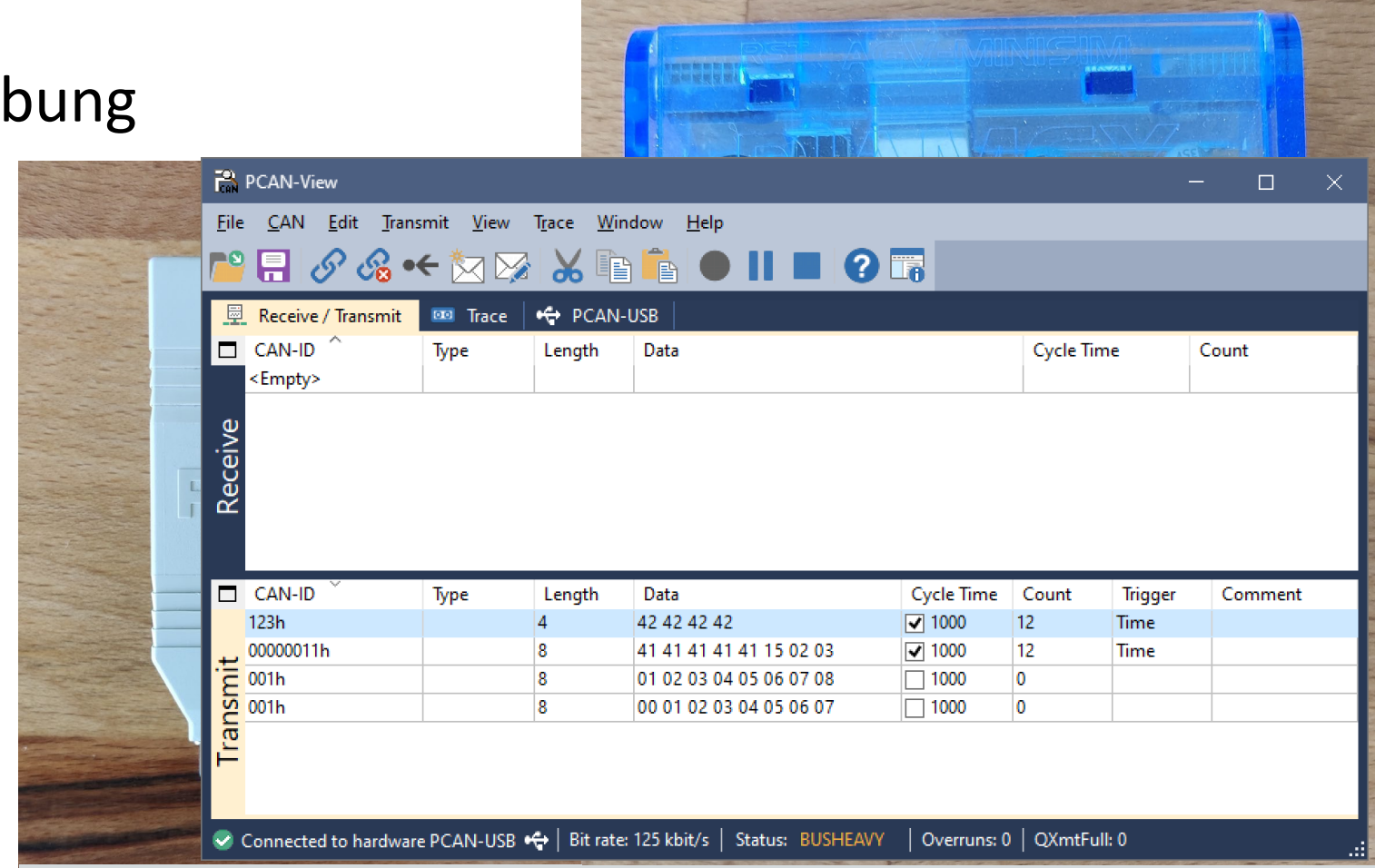


Hardware Aufbau





Testumgebung



Herzlichen Dank an die Firma AVL für die Bereitstellung der Utensilien

PCAN USB Adapter
https://www.peak-system.com/PCAN-USB.199.0.html

DIAMEX OBD2 Simulator
https://www.diamex.de/dxshop/Diamex-OBD2-Simulator-alle-Protokolle

Software

Verwendete Bibliotheken:


```
#include <OBD2.h>    //Zugriff auf OBD2 Protokoll
#include <CAN.h>     //CAN Kommunikation
#include <SPI.h>     //Zugriff auf Arduino PINs
#include <SD.h>      //Zugriff auf SD Karte
```

CAN und OBD2 Bibliothek: https://github.com/sandeepmistry

Setup Block

- Serielle Verbindung starten für Monitoring
- Starten der SD Karte für Logging
- Starten der OBD2 Kommunikation
- Erstellen eines Arrays der von der ECU unterstützten OBD2 PIDs

```
11 void setup() {
12
13   //File dataFile = SD.open("datalog.txt", FILE_WRITE);
14
15   Serial.begin(9600);
16
17   while (!Serial) {
18     ;
19   }
20   Serial.print("Initializing SD card...");
21   if (!SD.begin(chipSelect)) {
22     Serial.println("Card failed, or not present");
23     while (1);
24   }
25   Serial.println("card initialized.");
26
27   Serial.println("OBD2 Receiver");
28   if (!OBD2.begin()) {
29     Serial.println("Starting OBD2 connection failed!");
30     while (1);
31   }
32
33   //check for supported PIDs and set corresponding array value to true
34   for (int i = 1 ; i < 200; i++) {
35     if (OBD2.pidSupported(i)) {
36       PIDs[i] = true;
37       Serial.println("PID" + String(i) + " supported.");
38     }
39   }
40 }
```

Loop Block

- Starten der OBD2 Verbindung falls deaktiviert
- Durchschreiten von 200 Werten, bei richtigem Eintrag in Array der unterstützten PIDs durchführen der Hilfsfunktion „processPID“
- Durchführung jede Sekunde

```
42 void loop() {
43
44   if (!OBD2.begin()) {
45     OBD2.begin();
46   }
47   //empty datastring on each run
48   String datafile = "";
49
50   // loop through PIDs 0 to 199, reading and printing the values for supported PIDs
51   for (int j = 0; j < 200; j++) {
52     if (PIDs[j] = true) {
53       processPid(j);
54     }
55   }
56
57   // querry for new values with 1Hz
58   delay(1000);
59 }
```

Funktion processPID

- Überspringen von nicht verfügbaren PIDs
- Speichern der relevanten Informationen in Hilfsvariable „datafile“
- Ausgabe der Variable an die Serielle Schnittstelle
- Ausgabe an die SD Karte

```
65 String processPid(int pid) {
66
67   //File dataFile = SD.open("datalog.txt", FILE_WRITE);
68
69   if (!OBD2.pidSupported(pid)) {
70     // PID not supported, continue to next one ...
71     return "PID" + String(pid) + " not supported";
72   }
73   else {
74     File dataFile = SD.open("datalog.txt", FILE_WRITE);
75     //build string containing Data
76     String datafile = "PID" +
77                       String(pid) + ": "
78                       +  OBD2.pidName(pid) +
79                       ": " + String(OBD2.pidRead(pid))
80                       + " " + String(OBD2.pidUnits(pid));
81
82     //output string to SD card for logging
83     Serial.println(datafile);
84
85     dataFile.println(datafile);
86     dataFile.close();
87
88     return datafile;
89   }
```

COM6

Send

```
17:02:28.540 -> PID12: Engine RPM: 1456.00 rpm
17:02:28.633 -> PID13: Vehicle speed: 39.00 km/h
17:02:28.679 -> PID15: Intake air temperature: 72.00 °C
17:02:28.725 -> PID16: MAF air flow rate: 83.20 grams/sec
17:02:28.818 ->
17:02:28.864 -> PID20: Oxygen Sensor 1 - Short term fuel trim: 6.25 %
17:02:28.910 ->
17:02:30.021 ->
17:02:30.067 ->
17:02:30.160 ->
17:02:30.206 -> PID4: Calculated engine load: 25.10 %
17:02:30.251 -> PID5: Engine coolant temperature: 17.00 °C
17:02:30.343 -> PID10: Fuel pressure: 192.00 kPa
17:02:30.390 -> PID11: Intake manifold absolute pressure: 96.00 kPa
17:02:30.436 -> PID12: Engine RPM: 1456.00 rpm
17:02:30.528 -> PID13: Vehicle speed: 39.00 km/h
```

Autoscroll Show timestamp

Newline 9600 baud Clear output

G:\datalog.txt - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

datalog.txt

```
22061 PID10: Fuel pressure: 192.00 kPa
22062 PID11: Intake manifold absolute pressure: 96.00 kPa
22063 PID12: Engine RPM: 1456.00 rpm
22064 PID13: Vehicle speed: 39.00 km/h
22065 PID15: Intake air temperature: 72.00 °C
22066 PID16: MAF air flow rate: 83.20 grams/sec
22067
22068 PID20: Oxygen Sensor 1 - Short term fuel trim: 6.25 %
22069
22070
22071
22072
22073 PID4: Calculated engine load: 25.10 %
22074 PID5: Engine coolant temperature: 17.00 °C
22075 PID10: Fuel pressure: 192.00 kPa
22076 PID11: Intake manifold absolute pressure: 96.00 kPa
22077 PID12: Engine RPM: 1456.00 rpm
22078 PID13: Vehicle speed: 39.00 km/h
22079 PID15: Intake air temperature: 72.00 °C
22080 PID16: MAF air flow rate: 83.20 grams/sec
22081
22082 PID20: Oxygen Sensor 1 - Short term fuel trim: 6.25 %
22083
22084
22085
22086
22087 PID4: Calculated engine load: 25.10 %
22088 PID5: Engine coolant temperature: 17.00 °C
22089 PID10: Fuel pressure: 192.00 kPa
22090 PID11: Intake manifold absolute pressure: 96.00 kPa
22091 PID12: Engine RPM: 1456.00 rpm
22092 PID13: Vehicle speed: 39.00 km/h
22093 PID15: Intake air temperature: 72.00 °C
22094 PID16: MAF air flow rate: 83.20 grams/sec
22095
22096 PID20: Oxygen Sensor 1 - Short term fuel trim: 6.25 %
```

Herzlichen Dank für Ihre Aufmerksamkeit!