Seminarkurs: Azamat Winkler

OBD-Auslesegerät

Inhalt

Grundidee:

Die Grundidee war, dass ich einen Motor baue. Er sollte 2 Zylinder haben, 150ccm und wie ein auf den Kopf gestellten Boxermotor aussehen. Aber als ich mich, denn damit auseinandergesetzt habe, wie viel man da berechnen oder beachten muss und es dazu noch ein Vermögen kostet, habe ich mich dazu entschieden dieses Projekt nicht zumachen. Dennoch sollte mein Projekt etwas, mit Autos zu tun haben. Da ich mich in der Informatikklasse befinde und ich mich mit Freunde und Familie unterhalten habe, kam ich dann zu dem Entschluss ein OBD-Auslesegerät zu programmieren. Ich weiß, dass große Firmen oder Werkstätte, höchst professionelle Auslesegeräte haben. Dennoch ist meine Idee, ein einfaches Ausleseprogramm zu programmieren welcher jeder Nutzer leicht verstehen und bedienen kann. Außerdem soll es dem Nutzer frühzeitig Probleme am Fahrzeug mitteilen können. Sodass wenn zum Beispiel der Öldruck, der in Echtzeit überprüft wird, zu hoch ist, eine Warnung an den Fahrer weitergegeben wird. Die Idee mit der Bluetooth-Verbindung, kam zwar später, ist aber ein entscheidender Aspekt des Programmes. Dadurch müsste der normale Nutzer den OBD-Stecker nur einmal einstecken und könnte dann per Bluetooth die Daten live empfangen.

Erstes Konzept:

Erstmals habe ich mich im Internet durch Artikel, Videos, Beiträge und Foren mich über die grundlegenden Kenntnisse, welche zum richtigen Beginnen des Projektes erforderlich sind, informiert. Da ich am Anfang keinen Bluetooth Adapter besaß, hatte ich die Idee, erstmals ein Auto zu simulieren und damit mein Programm mit Werten zu füttern. Deshalb habe ich mich über ein simuliertes Auto im Internet schlau gemacht und einen passenden Beitrag gefunden. Dann habe ich mir einen kleinen Plan gemacht, wie das Programm am Ende aussehen sollte und wie es funktionieren soll. Als ich den Plan fertig gemacht habe, habe ich mir ein paar alte Programme von mir angeschaut, um zu schauen, ob ich noch etwas davon verwenden kann. Dann habe ich mir bestimmte Tools installiert, welche notwendig waren, um das Programm zu entwickeln und angefangen eine Userinterface zu gestalten.

Programmieren:

Als Erstes habe ich das Programm auf dem Java Editor der Schule programmiert. Aber nach einer kurzen Zeit, hat dieser mir mit der Zeit immer mehr Probleme bereitet. Deshalb habe ich, recht am Anfang, das Programmierumfeld auf Oracle verschoben. Als ich dann auf Oracle gewechselt habe, fiel mir das Programmieren auch viel leichter, da es mir auch immer die Fehler besser erklärt und mir die möglichen Lösungen angezeigt hat. Als Erstes habe ich ein Terminal Programm geschrieben, welches ich nach einem Tag durch einen Grafischen User Interface ersetzt habe.

Dann habe ich auf Paint die Grundlage meines Tachos gezeichnet. Dieses Bild habe ich dann hinzugefügt und getestet, ob es auf meinem Computer läuft. Dann habe ich im Internet recherchiert, wie ich den ein Auto simulieren könnte. Und habe dann eine Webseite gefunden, welche diese einfach zur Verfügung gestellt hat. Nachdem ich dann den Auto-Simulator hatte und er funktionsfähig war, habe ich zuerst eine PowerShell geschrieben, welche auf das simulierte Auto zugreifen kann.

Da diese funktioniert hat, habe ich es so umgeschrieben, dass es in meine Java Datei reingepasst hat. Dann habe ich die Simulation auf meinen Laptop umgelagert, was das Projekt realistisch aussehen lassen sollte. Deshalb habe ich einen Socket programmiert, welche auf die statische IP 192[redacted] zugreifen kann. Es greift über den Port [redacted] zu. Man kann das Programm durch ein kleines Wechseln von true und false umschalten. Somit kann ich entscheiden, ob ich das simulierte Auto oder ein echtes Auto als Quelle meiner Werte haben will. Um in der Simulation werte setzen zu können, habe ich diesen Befehl eingegeben.

*„Emulator.answer [‚RPM‘] = ‚<exec>ECU_R_ADDR_E + „04 41 0C %.4X" % int(4 * 500)</exec><writeln />'.*

Jetzt habe ich die Werte im Oracle Terminal empfangen und musste nur noch einen Pixelstrich zeichnen lassen, der mir diesen Wert auch anzeigt.

Da es bis jetzt nur den Wert aufgezeichnet hat, den ich eingegeben habe, habe ich die Tachonadel so programmiert, dass er zum Wert hin in 10er-Schritten hoch und runter zeichnet. Somit sah es dem Auto ähnlicher.

Erster Test:

Der erste Test lief erfolgreich und verlief ohne Probleme. Es war ermutigend zu sehen, dass das Java-Programm fehlerfrei gestartet werden konnte. Ich konnte das simulierte Auto mithilfe einer spezifischen PowerShell-Datei starten und Werte abfragen. Das war ein wichtiger Schritt, um sicherzustellen, dass die grundlegende Kommunikation zwischen dem Programm und dem simulierten Auto funktioniert.

Allerdings musste ich bisher die Werte des Autos manuell eingeben, was es schwierig macht, die echte Funktionalität des Programmes zu überprüfen. Eine Möglichkeit wäre, eine Schnittstelle zu implementieren, über die die Werte automatisch von einem Auto oder einer Datenquelle (Laptop) abgerufen werden können. Das könnte ich beispielsweise durch einen Sensor-Array beziehungsweise meinem OBD-Auslesegerät machen. Aber dazu komme ich später noch mal. Dadurch könnte ich sicherstellen, dass die Werte realistisch sind. Als WLAN-Host habe ich für den ersten Meilenstein einen kleinen SIM-Karten-Router verwendet, um eine beständige Verbindung aufzubauen und aufrechtzuerhalten. Das ist praktisch, da ich dadurch eine drahtlose Verbindung zu meinem Laptop besitze. Dadurch wirkt es realistischer. Natürlich hätte ich eine kabelgebundene Version machen können, welche weitaus effizienter wäre, was aber nicht zu meinen zukünftigen Plänen passt.

Rückblick:

Nachdem sie mir die Tipps gegeben haben, nahm ich mir Zeit, um rückblickend über meine bisherigen Fehler nachzudenken. Ich schrieb sie auf und überlegte, wie ich die erhaltenen Ratschläge am besten umsetzen könnte. Dabei überlegte ich mir, wie weit ich an dem Programm weiter Programmieren will, ob ich es noch so weit Programmieren soll, dass mein Programm in der Lage ist, die Werte eines echten Autos auszulesen. Da ich erst 1-2 Jahre Erfahrung mit der Programmiersprache Java habe, hatte ich mir überlegt, es nur grafisch weiterzuentwickeln. Es wäre eine Herausforderung, die mich dazu veranlassen würde, meine Fähigkeiten auch außerhalb der Schulzeit zu verbessern.

Nach langem Abwägen kam ich zu dem Entschluss, dass ich es mindestens versuchen werde. Da ich in der Lage war, mein bisheriges Programm erfolgreich zu

Programmieren und vorzustellen, und zwar innerhalb einer recht akzeptablen Zeitspanne, gab mir das die Zuversicht, dass ich auch diese neue Weiterentwicklung meines Programmes noch in diesem Jahr so erarbeiten kann, sodass ich es am Ende vollständig vorstellen kann. Um mich besser zu organisieren und strukturieren, beschloss ich, einen kleinen Plan zu erstellen. Ich skizzierte grob, wie das Programm aussehen sollte und wie es funktionieren könnte. Diesen Plan können sie in meiner PAP-Zeichnung sehen. Es war wichtig, eine klare Vorstellung von den Schritten zu haben, die erforderlich sind, um mein Ziel zu erreichen. So wie sie es mir empfohlen haben. Dieser Plan diente als Leitfaden. Die PAP bot mir die Möglichkeit, meine Ideen strukturiert zu dokumentieren und meinen Fortschritt zu verfolgen.

Die Reflexion über meine Fehler, das positive Vertrauen in meine Fähigkeiten und die Erstellung eines Plans mit einer groben Idee haben mir geholfen, meine nächsten Schritte klarer anzugehen und mich auf die Fertigstellung meines Projekts zu fokussieren. Es war mir wichtig, mich selbst herauszufordern, um zu schauen, wie weit ich dieses Projekt nach vorne bringen kann. Wie sich herausgestellt hat mit Erfolg

Schwierigkeiten:

Natürlich gab es bis dorthin schon eine Menge kleiner Schwierigkeiten. Ein Problem war, dass ich die Tachonadel richtig platziert bekomme. Ich habe anfangs die Nadel Per Pixel Einstellung an ihren richtigen Platz gebracht. Danach habe ich mit kurzer Absprache meines Vaters entschieden, dass ich einfach die Größe des Bildes verwenden werde und dann einfach die Tachonadel in die Mitte zusetzen. Auch musste ich die Position der Nadel mit Pi berechnen. Zum Glück hatten wir dieses Thema mit Cosinus und Sinus kurz zuvor in Mathe, was es mir um einiges erleichtert hat.

Das Recherchieren für mein Projekt hat sehr viel Zeit in Anspruch genommen. Etwa 30 % meiner verwendeten Zeit wurde für das Recherchieren benötigt. Da es sehr aufwendig war, hat mich die Recherchen dazu gezwungen, mich tiefer in die verschiedenen Themen einzulesen, um die besten Lösungsansätze zu finden. Aber dadurch habe ich so viel zum Programmieren dazu gelernt, dass ich keine einzige Minute davon bereu.

Die für mich größte Herausforderung war, dass ich das komplette Programm neu schreiben musste, obwohl es funktioniert hat. Das Problem war, dass das Programm sehr ruckelig und ineffizient gelaufen ist. Generell war der Code sehr unübersichtlich geschrieben und viel größer als nötig, das führte letztendlich dazu, dass die Performance des Programms nicht wie erhofft war. Der größte Grund, warum das Programm ruckelig gelaufen ist, war, dass ich das ganze Programm auf einem Thread laufen lassen habe, welche die Werte abgefragt hat und sie dann auf meinem Bildschirm projiziert haben. Dadurch entstand ein Engpass von Daten, welche dazu führten, dass die Aktualisierung der Werte nicht flüssig verlief, was den Strich stottern lassen hat. Um dieses Problem zu lösen, habe ich eine neue Struktur angewandt, welche für die Abfrage einen eigenen Thread hatte und einen Thread für das Projizieren der Tachonadel. Somit habe ich es geschafft, dass ich nur noch eine Verzögerung von 100 Millisekunden habe. Diesen Wert habe ich gesetzt, da mein Handy selbst nicht in der Lage ist, mit niedrigeren Verzögerungszeiten zu arbeiten.

Auch war es eine Schwierigkeit für mich, in der Schule richtig zuarbeiten. Das lag daran, dass ich mich in der Schule nicht gut konzentrieren konnte, da die Umgebung recht laut war. Deshalb habe ich mich dazu entschieden, dass ich den Großteil meines Projektes in den Schulferien erarbeiten werde, da ich mich zu Hause wirklich den ganzen Tag auf das Programmieren konzentrieren konnte und mich dabei niemand gestört hat. So habe ich den größten Teil meines Projektes in den Herbst und Winterferien erarbeitet, indem ich mal 3-5 Tage 6-7 Stunden durchgearbeitet habe. Diese oben genannte Probleme konnte ich lösen, indem ich mir einfach die Zeit dazu genommen habe. (Genauere Daten über die Arbeit Aufteilung: Siehe Zeitplan).

Doch zusammengefasst gab es keine größeren Probleme, die mir wirklich Kopfschmerzen zugefügt haben.

Zeitplan:

- 4 Stunden Planung (In der Schule vor den Herbstferien)
- 25 Stunden Programmieren/ Recherchieren (In den Herbstferien)
- 6 Stunden Optimierung (In der Schule zischen Herbst und Winterferien)
- 12 Stunden Programmieren/ Recherchieren (In den Winterferien)
- 6 Stunden Optimierung/ Ausarbeitung (Inder Schule nach den Winterferien)
- 10 Stunden fertigstellen der Ausarbeitung (Nach den Winterferien)

Änderungen:

Nach der Absprache mit dem Lehrer habe ich dann wieder meine Programmierplattform gewechselt. Und zwar auf Android Studio. Denn ich hatte jetzt vor, daraus ein Handy Programm zu gestalten. Erst mal habe ich das Programm so weit geschrieben, sodass es gleich aussah wie auf dem Computer. Dann habe ich mir im Internet einen Bluetooth-Adapter bestellt. In der Zwischenzeit habe ich einen zusätzlichen Tacho für die Geschwindigkeit programmiert. Da ich schon eine Vorlage von meinem ersten Tacho hatte. War es nicht mehr so schwer und ich musste nur kleine Details verändern. Als dann der Bluetooth-Adapter kam, habe ich angefangen, den Bluetooth Socket zu Programmieren. Es hat fast eine ganze Woche gedauert, da ich bis dorthin noch nie etwas programmiert habe, indem ich Werte aus einem Bluetooth Adapter verarbeiten muss. Erstmals habe ich dem Bluetooth Gerät eine UUID (Universally Unique Identifier) zugeteilt. Und mich mit ihm verbunden. Der Bluetooth-Adapter hat von sich aus einem Code, welcher man einmalig beim Verbinden eingeben muss. Der Code lautet 1234.

Im Folgendem Abschnitt werde ich im Detail erklären, wie mein Bluetooth klasse funktioniert. Der Code ist eine Implementierung, welche meinem Android-Handy erlaubt, aus dem OBD-II Bluetooth-Adapter verschiedene Daten wie Drehzahlen, Geschwindigkeit, Wassertemperatur des Motors, Lufttemperatur und die Motorbelastung auszulesen und zu verwerten. Ich verwende in meinem Code eine API, damit ich den Bluetooth-Adapter verwalten und bedienen kann. Meine init Funktion (Siehe Quellcode im Anhang) initialisiert den Adapter und zeigt mir geradezu am Handy Display an, ob die Initialisierung erfolgreich war. Es gibt mir auch eine Nachricht, ob mein Bluetooth angeschaltet ist oder nicht. Danach suche ich, ob es unter meinen Geräten einer mit dem Namen obdii existiert. Wenn dieses Gerät gefunden wurde, erstellt mein Programm ein Socket und ich habe dann zugriff auf den Adapter. Als ich fertig war, bin ich dann in den Keller gegangen, um weiter zu programmieren. Das war der einzige Ort, an dem ich eine stabile Internetverbindung habe und das Bluetooth Gerät nicht zu weit von mir entfernt ist. Nachdem ich Nahe genug am Auto war, habe ich den von oben erwähnten zweiten Thread erstell (BThread). Dieser ist nur dafür da, um Befehle zu verschicken und Daten zu

empfangen. Der Thread fragt alle 100 ms die Werte von der Drehzahl (mm_rpm), Geschwindigkeit (mm_speed), Wassertemperatur (mm_watertemp), Lufttemperatur (mm_airtemp) und Motorlast (mm_engload) ab. Auch diese Daten werden bearbeitet und auf das Allernötigste gekürzt, sodass am Ende nur noch eine 4–5-stellige Hex Zahl dort stehen hat. Bestimmte Werte mussten subtrahiert oder dividiert werden. Die Drehzahlen müssen durch 4 geteilt werden, bei den Temperaturen müssen jeweils 40 grad subtrahiert werden, da der Zähler bei minus 30 Grad anfängt zu zählen. Warum die Motorlast durch 2,55 dividiert werden muss, weiß ich auch nicht. In meinem Code gibt es keinen Code, der Fehler ausbessert oder behebt, sondern es wird in den meisten Fällen eine Null zurückgegeben. Am Anfang habe ich die Variablen sehr kompliziert benannt, weshalb ich dann mittendrin entschieden habe, sie umzubenennen. Die Temperaturen und Motorlast habe ich anfangs unter meine 2 Tacho anzeigen platziert. Aber weil alles sehr sperrig war, habe ich mich dann dazu entschieden, diese Werte auf einer eigenen User Interface zu stellen. Also habe ich eine 2te User Interface gemacht und im Internet recherchiert was ich für Möglichkeiten haben, die UI zu switchen. Am Ende habe ich das Swipen umgesetzt, sodass ich mit dem Finger immer hin und her switchen kann. Dazu habe ich noch ein kleines Programm geschrieben, was es mir vereinfacht, rauszufinden, welche Sensoren das Auto hat und welche Befehle ich dem OBD-Gerät schicken kann. Ich habe nur so wenige Daten bei mir anzeigen lassen, da unser Auto nicht viele Sensoren hat und ich deshalb nur wenige Anzeigen lassen kann. Bei moderneren Autos könnte ich wahrscheinlich viel mehr Daten über das Auto entnehmen und mir anzeigen lassen. Eine zusätzliche Idee ist, dass die Werte, wenn sie in gefährliche maßen steigen, in Farben aufleuchten sollen. So habe ich eingestellt, dass wenn die Motorlast über 90 %, die Drehzahlen über 3500 Umdrehungen, die Wassertemperatur über 110 Grad oder die Lufttemperatur über 50 Grad geht, der jeweilige Wert Rot angezeigt wird. Auch habe ich eingestellt, dass, wenn die Wasser Temperatur über 70 Grad geht, der Wert grün wird, was mir Signalisieren soll, dass der Motor endlich warm gefahren wurde. Zuletzt habe ich dann 2 kleine 4 Ecke im Tacho hinzugefügt, welche mir Signalisieren, wann ich Hoch und Runterschalten soll. Ich habe es nach Logik aufgebaut. Wenn die Drehzahl und die Motorlast Hoch ist, bedeutet es, dass du wahrscheinlich keine starken Höhen oder Tiefen fährst und dass es Zeit ist hochzuschalten. Das gleiche Prinzip, wenn die Motorlast hoch, aber die Drehzahlen niedrig sind. Das weißt uns darauf hin, dass das Auto wahrscheinlich bergauf fährt. Durch ein Licht wird dann dem

Fahrer signalisiert, dass er runterschalten soll. Am Ende habe ich mit // wichtige Sachen aufgeschrieben, sodass sie meine Gedankengänge oder Variablen Deklaration besser verstehen können. Auch habe ich eine Javadoc Dokumentation generieren lassen, indem ich wichtige Sachen mit //* */ gekennzeichnet habe. Diese Dokumentationen finden sie im Anhang.

OBD:

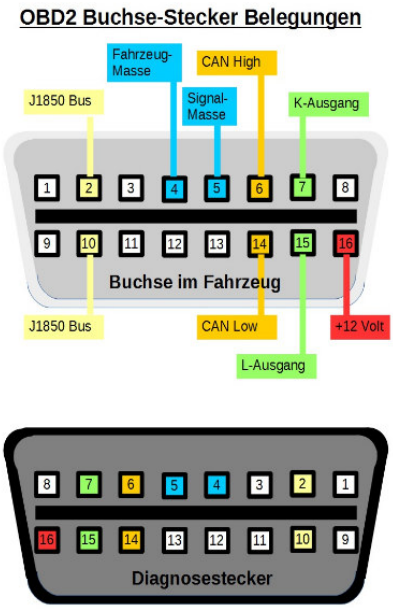
Ich erkläre in diesem Abschnitt die Funktionen und Möglichkeiten eines OBD-Auslesegerätes.

OBD bedeutet so viel wie On-Board-Diagnostik und ist ein standardisiertes System, das in Fahrzeugen verwendet wird, um verschiedene Daten ihrer Leistung und Daten des Autos zu überwachen und zu melden. OBD-Systeme wurden in den 1960er-Jahren von Automobilherstellern benutzt. Also ist die Technologie schon über 60 Jahre alt und wurde mit der Zeit gut weiterentwickelt. Es ist dafür gedacht, um Motorprobleme herauszufinden und zu beheben beziehungsweise überschreiben. Es spielt heute eine wichtige Rolle bei der Fahrzeugwartung und der Emissionskontrolle. Viele Tuner benutzen OBD-Geräte, um ihre Emissionswerte zu verändern, damit sie eine bessere Umweltplakette bekommen. Dies braucht aber gute Kenntnisse über Automechanik und Programmieren. Einerseits ist es illegal, aber auch kann man extrem viel Schaden am Auto anrichten, wenn man was vermasselt bei dem Überschreiben.

Die Geschichte von OBD startet mit der Anfangszeit der computergesteuerten Motorkontrollsysteme. In den 1980er-Jahren wurden elektronische Motorsteuerungen immer geläufiger, und es gab eine große Nachfrage für solche OBD-Geräte, da die Technik immer komplexer wurde und Motorprobleme waren ab jetzt nicht nur mechanisch. Deshalb bestand ein Bedarf an einer standardisierten Methode zur Diagnose von Motorproblemen. Das führte zur Einführung von OBD-I-Systemen, die zwischen den Herstellern immer unterschiedlich war. Ein gutes Beispiel ist BMW, welche schon immer eine eigene Technologie hatten, um ihre Autos auszulesen. Zwar konnte man auch mit den herkömmlichen Geräten ein paar Fehler beheben, aber um das volle Potenzial aus den BMW-Autos herauszubekommen, braucht man die von BMW hergestellte Software. Das erschwerte es den unabhängigen Werkstätten und

Technikern, an verschiedenen Fahrzeugmodellen zu arbeiten. Um dieses Problem zu lösen, schrieb die California Air Resources Board (CARB) ab 1991 die Verwendung eines standardisierten OBD-Systems für alle in Kalifornien verkauften Fahrzeuge vor. Dieses standardisierte System, bekannt als OBD-II, nutze ich auf für mein Projekt. Es war ein Meilenstein und wurde zum Grundstein für moderne OBD-Systeme, die weltweit verwendet werden. OBD-II führte einen gemeinsamen Satz von Diagnosefehlercodes (DTCs), welche man im Internet ganz einfach recherchieren kann. Auch brachte es einen standardisierten Stecker zur Datenabfrage ein, welche bis heute in den meisten Autos verbaut sind. Davon ausgeschlossen sind bestimmte Modelle und sehr alte Autos wie zum Beispiel der Scirocco gt2.

An dem unteren Bild sind die wichtigsten Pins der OBD-Stecker gekennzeichnet.



OBD-II-Systeme nutzen verschiedene Sensoren und Komponenten, um die Leistung des Motors und der Abgasreinigungssysteme zu überwachen. Diese Sensoren

erfassen Daten zu Parametern wie Motordrehzahl, Fahrzeuggeschwindigkeit, Kühlmitteltemperatur, Sauerstoffgehalt im Abgas und mehr. Das OBD-II-System überwacht dauerhaft diese Werte und speichert diese festgestellten Fehler oder Unregelmäßigkeiten als DTCs. Mein Programm speichert diese Fehler nicht, sondern warnt dich Live beim Fahren. Als dann die Neuzeit kam und Smartphones immer wichtiger für unser Leben wurde, wurde die Technologie des OBD-Bluetooth-Adapter vorgestellt. Diese Technik wurde immer beliebter bei den normalen alltäglichen Fahrern, da man sie nur einmal einstecken musste und sie sogar beim Fahren nutzen kann. Ein OBD-Bluetooth-Adapter ist ein kleines Gerät, das in die OBD-II-Buchse eines Fahrzeugs gesteckt wird und drahtlos über Bluetooth eine Verbindung zu einem Smartphone oder Tablet herstellt. Doch muss man auch beachten, dass zwar jetzt jeder es verwenden kann, aber davon abgeraten wird. Es braucht Wissen über das Auto ohne Technologie. Dass Lesen von Werten, kann ein jeder tun. Aber überschreiben von Daten oder löschen von Fehlercodes sollten von Profis gemacht werden. OBD-Bluetooth-Adapter arbeiten, indem sie eine Bluetooth-Verbindung zwischen dem OBD-II-System des Fahrzeugs und einer mobilen App herstellen. Der Adapter fungiert als Brücke und überträgt Daten von den Sensoren des Fahrzeugs auf das Mobilgerät. Dadurch können wir dann auf eine Vielzahl von Informationen zugreifen, einschließlich Motordrehzahl, Fahrzeuggeschwindigkeit, Kraftstoffverbrauch, Emissionsdaten und mehr. Für die Genauere Befehle werde ich hier eine Tabelle einfügen, auf der man schauen kann, welche auslese Möglichkeiten existieren. Die markierten Zeilen Habe ich verwendet. Rechts ist das, was der Programmierer in sein Programm schreiben muss, um die richtigen Werte herauszubekommen. Diese Tabelle ist um einiges länger, aber diese Werte braucht der normale Nutzer nicht. Das Wichtigste steht in der Tabelle 01. Die Tabellen sind nummeriert. Also wenn du Engine Speed haben willst, muss du nicht nur die 0C an den OBD-Stecker schicken, sondern auch die Tabellen Bezeichnung. In diesem Fall „010C“.

Service / Mode (hex)	Description
01	Show current data
02	Show freeze frame data
03	Show stored Diagnostic Trouble Codes
04	Clear Diagnostic Trouble Codes and stored values
05	Test results, oxygen sensor monitoring (non CAN only)
06	Test results, other component/system monitoring (Test results, oxygen sensor monitoring for CAN only)
07	Show pending Diagnostic Trouble Codes (detected during current or last driving cycle)
08	Control operation of on-board component/system
09	Request vehicle information
0A	Permanent Diagnostic Trouble Codes (DTCs) (Cleared DTCs)

PIDs (hex)	PID (Dec)	Data bytes returned	Description	Min value	Max value	Units	Formula[a]
00	0	4	PIDs supported [$01 - $20]				Bit encoded [A7..D0] == [PID $01..PID $20] See below
01	1	4	Monitor status since DTCs cleared. (Includes malfunction indicator lamp (MIL), status and number of DTCs, components tests, DTC readiness checks)				Bit encoded. See below
02	2	2	DTC that caused freeze frame to be stored.				Decoded as in service 3
03	3	2	Fuel system status				Bit encoded. See below
04	4	1	Calculated engine load	0	100	%	$\frac{100}{255}A$ (or $\frac{A}{2.55}$)
05	5	1	Engine coolant temperature	-40	215	°C	$A - 40$
06	6	1	Short term fuel trim—Bank 1	-100 (Reduce Fuel: Too Rich)	99.2 (Add Fuel: Too Lean)	%	$\frac{100}{128}A - 100$ (or $\frac{A}{1.28} - 100$)
07	7	1	Long term fuel trim—Bank 1				
08	8	1	Short term fuel trim—Bank 2				
09	9	1	Long term fuel trim—Bank 2				
0A	10	1	Fuel pressure (gauge pressure)	0	765	kPa	$3A$
0B	11	1	Intake manifold absolute pressure	0	255	kPa	A
0C	12	2	Engine speed	0	16,383.75	rpm	$\frac{256A + B}{4}$
0D	13	1	Vehicle speed	0	255	km/h	A

2ter Test:

Der zweite Test verlief auch sehr erfolgreich. Ich konnte ihnen Per Video zeigen, dass mein Programm funktioniert und dass alle Features, welche ich vorhin erklärt habe, auch so funktionieren wie sie sollen. Diese Videos werden auch in den Anhang gepackt. Das Programm ist leistungsstark gewesen und hat nur einen minimalen Lack, wenn man die Drehzahlen sehr schnell sehr viel erhöht. Der Quellcode sah ordentlich und leicht zu verstehen aus. Das Programm wurde an mehreren verschiedenen Autos erfolgreich ausgetestet. An unseren 2 Skodas (Skoda Citigo, Skoda Roomster), an einem BMW e92 und an einem Peugeot 106. Es wurde auch auf längeren Strecken getestet. Die Strecke verlief zwischen Waldshut nach Lörrach nach Freiburg nach Titisee Neustadt und letztendlich wieder nach Waldshut zurück. Es sind recht genau 3 Stunden und 210,5 km Strecke. In der Zeit musste das Handy alle 10 min kurz angetippt werden, damit es nicht ausschaltet. Das liegt aber an meinem Smartphone, da ich nicht einstellen kann, dass mein Bildschirm dauerhaft an bleibt. Das Programm hat sich als sehr stromsparend rausgestellt. Es verbraucht 40 % meines Akkus nach 3 Stunden Display On Zeit. Wenn man rückwärtsfährt, zeigt mir das Programm eine Geschwindigkeit an, welche aber als eine positive integer ausgegeben wird. Leider kann das Programm nicht feststellen, ob das Auto rückwärts fährt. In wenigen Fällen hat das Programm nicht beim ersten Anlauf funktioniert, sondern musste dann am Handy neugestartet werden.

2ter Rückblick:

Ich habe das Programm fertig und rückblickend bin ich doch sehr froh, dass ich gerade dieses Projekt ausgewählt habe. Ich habe viel dazu gelernt, und zwar in informatischer als auch Auto mechanischen Teil. Ich habe viel über die Sockets gelernt und wie man jene richtig konfiguriert. Ich sage, dass ich dadurch im Generellen besser in Java programmieren geworden bin. Ich habe mit diesem Projekt 2 Sachen unter einen Hut gebracht, die mir gefallen. Es hat zwar oft viel Nerven gebraucht und mich strapaziert und dennoch hat es mir sehr Spaß gemacht zusehen, dass mein Programm funktioniert und genauso geworden ist, wie ich es wollte. Ich habe meinen eigenen Zeitplan fasst, perfekt eingehalten und meinem erst erstellten Plan sehr nahe eingehalten. Ich habe bestimmt was dazu gelernt, was mir in der Zukunft Weiterhelfen

wird. Ich will, dass Programm auch noch außerhalb des Projektes Weiterentwickeln und schauen, was ich da noch alles rausholen kann.

Schwierigkeiten 2:

Das Programmieren des Bluetooth-Socket hat sich als sehr schwierig herausgestellt und hat sehr lange zum Recherchieren gebraucht. Als ich mich in die das Thema eingelesen habe, waren im Internet viele verwirrende Beiträge, welche es schwerer gemacht haben, an die eigentliche Lösung heranzukommen. Aber nachdem man es verstanden hat, schien es doch alles sehr logisch.

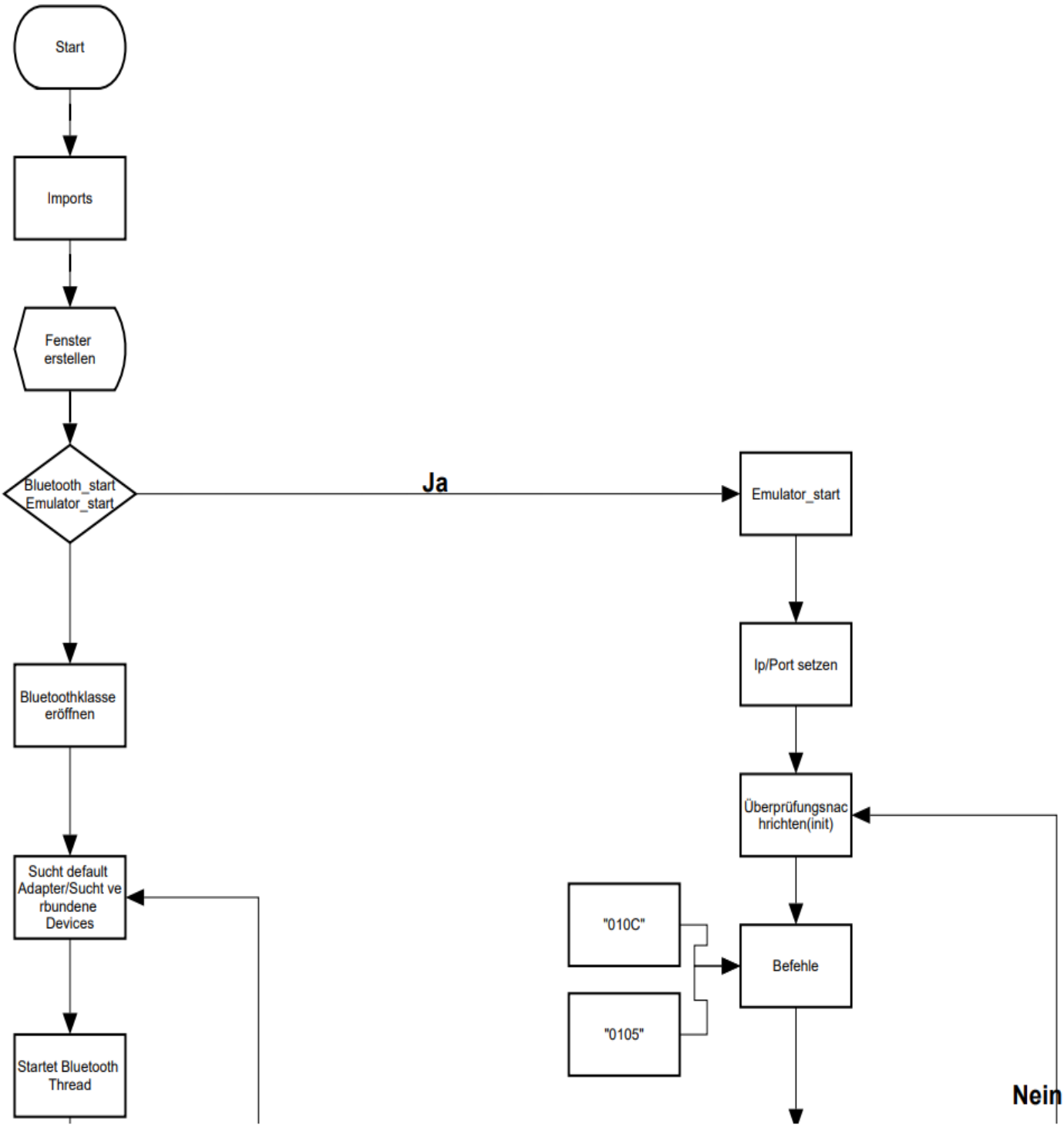
Ein weiteres Problem ist, wenn ich das Auto ausschalte, zeigt mir meine App verschiedene Werte an. Zum Beispiel Wasser und Lufttemperatur -30 Grad und Drehzahl und Geschwindigkeit eine 1. Dabei sollten an allen stellen 0 anzeigen, wenn ich das Auto ausschalte. Ich habe mich bis zum return 0 herangetastet und habe herausgefunden, dass das Programm das return 0 erfasst, aber nicht zurückgibt. Die -30 Grad, kann ich mir nur dadurch erklären, dass es irgendwie mit der -40 was zu tun hat (Siehe Wiki Beitrag OBD-Codes).

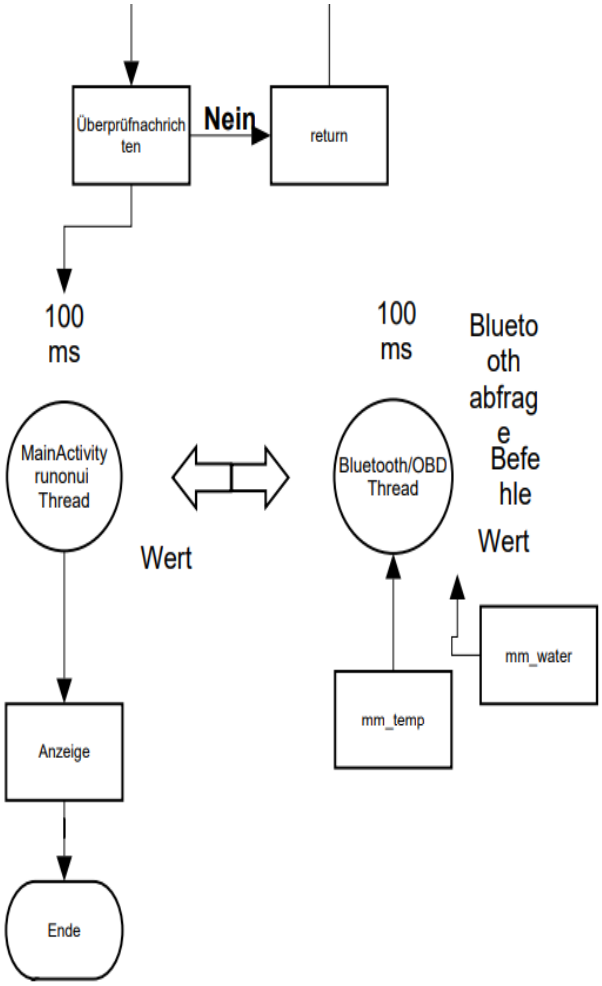
Wie oben erwähnt habe ich ab einem bestimmten Punkt im Keller gearbeitet. Dies war meist unangenehm, da es nicht warm ist, dort kein richtiger Stuhl ist und man sich dort nicht richtig konzentrieren konnte. Aber weil ich den Plan eigentlich schon davor hatte, war es im Keller eigentlich nur noch stupides Abtippen.

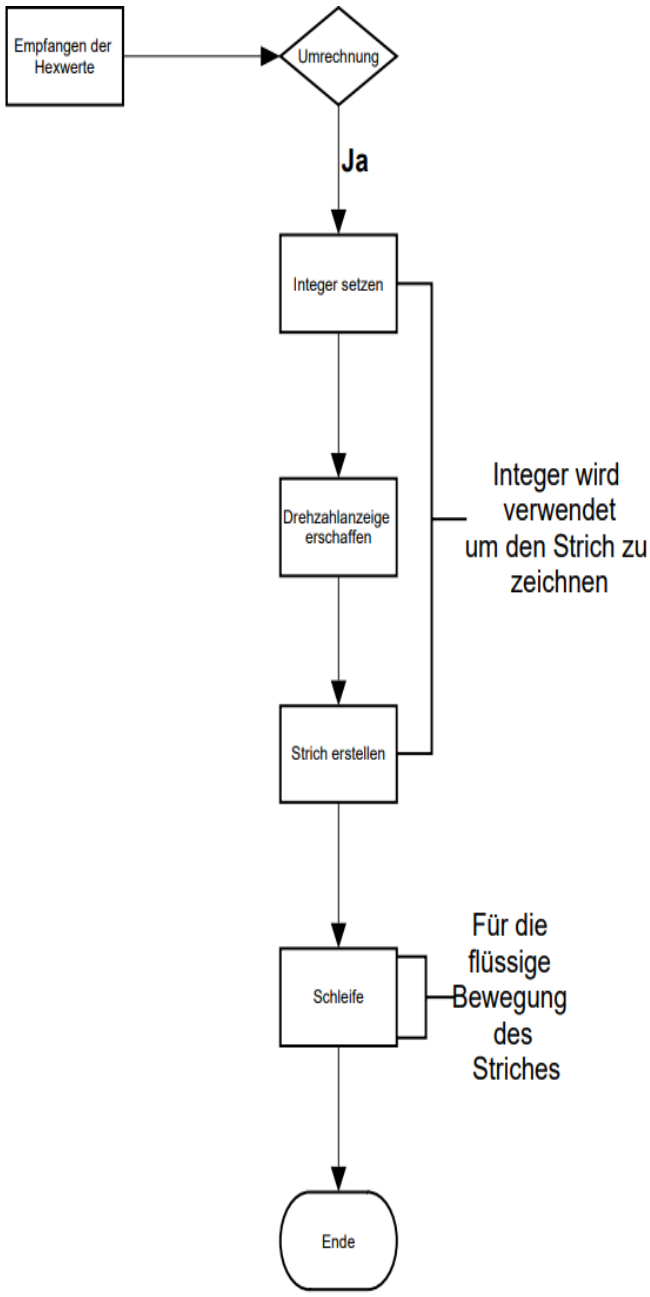
Ich habe an einem Punkt ihne[redacted]esagt, dass ich ein Hoch und runterschalt Assistenten Programmieren werde. Nach ihrer Aussage hätte ich es mit Kurven Programmieren sollen, sodass er durch Kurvenschneidungen erkennen kann, ob der Fahrer Hoch oder Runterschalten muss. Ich habe einen Assistenten auf Logik gebaut, aber den von ihnen vorgeschlagene Lösungsvorschlag konnte ich nicht Programmieren

Der letzte Punkt ist, dass ich viele Ideen hatte, welche ich Programmieren wollte, welche aber schier unmöglich sind für einen Anfänger. Dennoch muss man sagen, dass es ja auch ein Projekt für die Oberstufe ist, und dafür bin ich sehr zufrieden mit meinem Projekt.

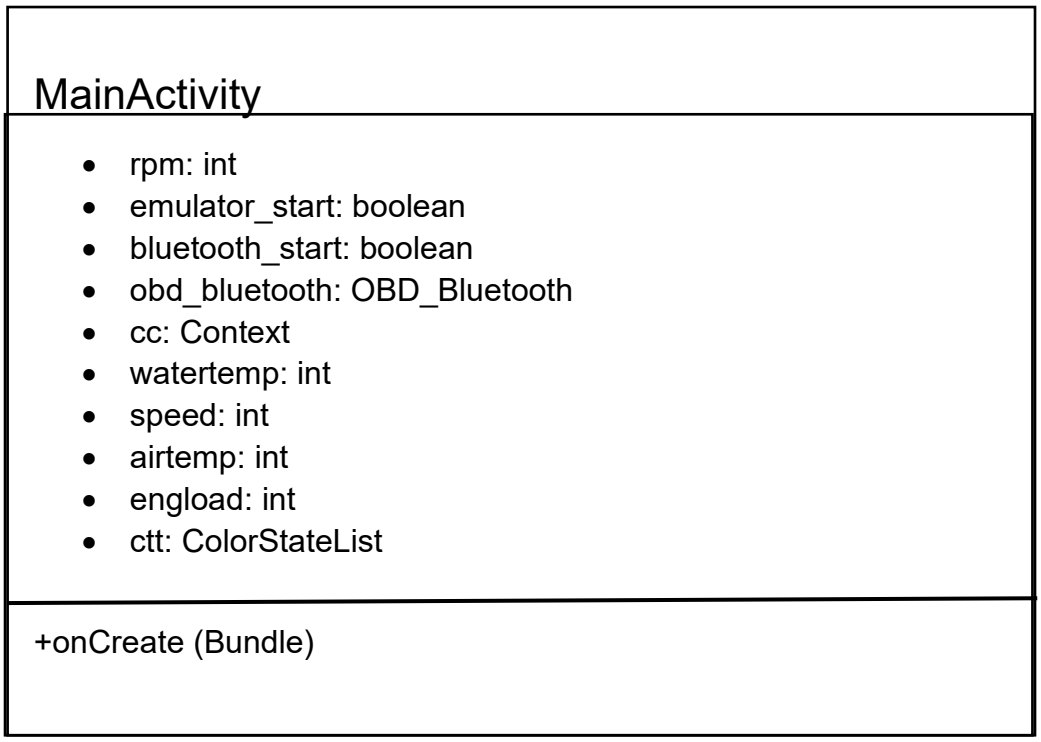
PAP-Zeichnung

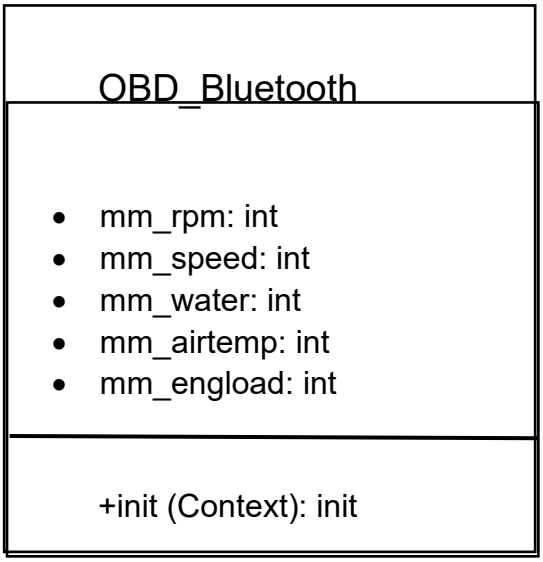


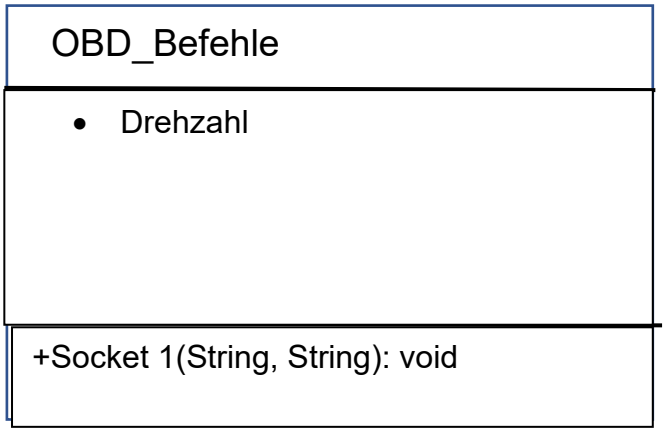

Überprüfnachrichten
Nein
return
100 ms
100 ms
Blueto oth abfrag e Befe hle
MainActivity runonui Thread
Wert
Bluetooth/OBD Thread
Wert
mm_water
mm_temp
Anzeige
Ende

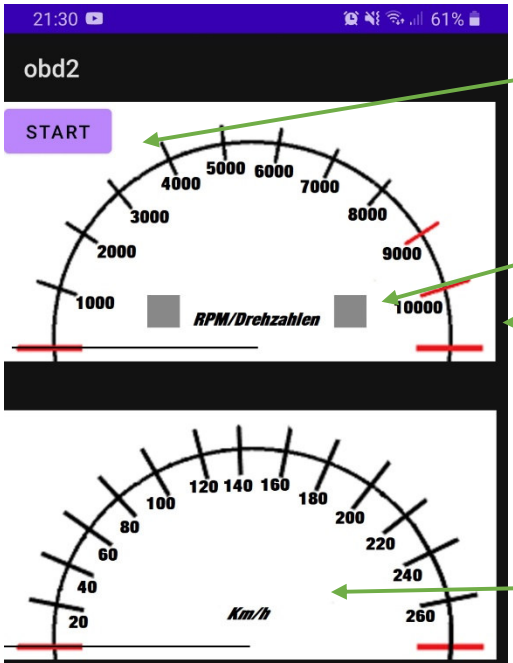

Empfangen der Hexwerte
Umrechnung
Ja
Integer setzen
Drehzahlanzeige erschaffen
Integer wird verwendet um den Strich zu zeichnen
Strich erstellen
Schleife
Für die flüssige Bewegung des Striches
Ende

UML-Diagramm:







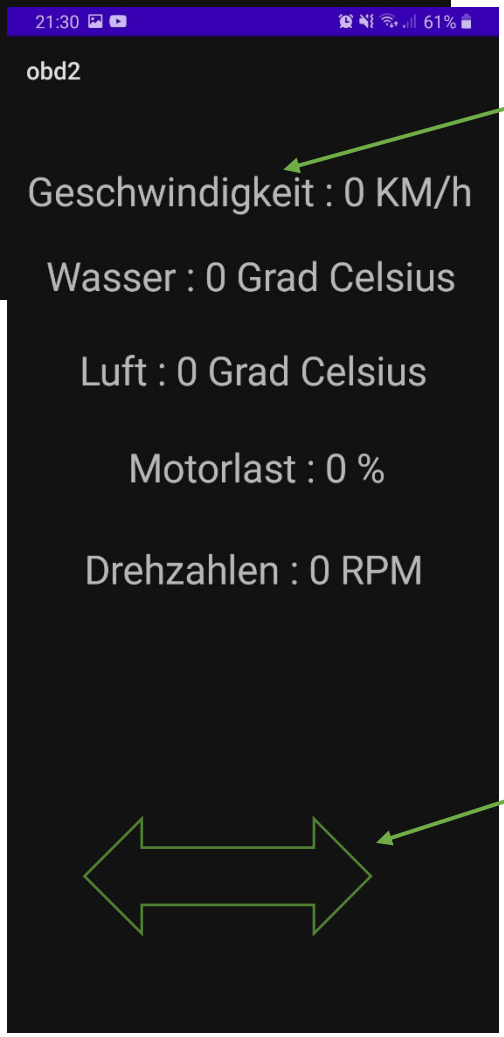


Not-Start Button

Hoch Runterschalt Assistent

Drehzahlanzeige

Geschwindigkeitsanzeige



Diese Zeilen werden Rot oder Grün (Erläuterung siehe oben)

Links oder rechts Swipen

Quellenverzeichnis:

https://www.oreilly.com/library/view/java-swing-2nd/0596004087/ch04s02.html

https://www.tutorialspoint.com/java/lang/math_cos.htm

https://javabeginners.de/Ein-_und_Ausgabe/Scanner.php

https://stackoverflow.com/questions/24104313/how-do-i-make-a-delay-in-java

https://stackoverflow.com/questions/57449238/calling-repaint-from-a-thread

https://android.googlesource.com/platform/packages/services/Car/+/refs/heads/master/obd2-lib/src/com/android/car/obd2/Obd2Connection.java

https://github.com/Ircama/ELM327-emulator

https://www.sparkfun.com/datasheets/Widgets/ELM327_AT_Commands.pdf

https://python-obd.readthedocs.io/en/latest/Command%20Tables/

https://en.wikipedia.org/wiki/OBD-II_PIDs

https://android.googlesource.com/platform/frameworks/base/+/56a2301/core/java/android/bluetooth

Eigenständigkeitserklärung

Hiermit bestätige ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken (dazu zählen auch Internetquellen) entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht

[redacted]

(Datum, Unterschrift)

Package com.example.obd2

Class MainActivity

java.lang.Object
  android.content.Context
    android.content.ContextWrapper
      android.view.ContextThemeWrapper
        android.app.Activity
          androidx.core.app.ComponentActivity
            androidx.activity.ComponentActivity
              androidx.fragment.app.FragmentActivity
                androidx.appcompat.app.AppCompatActivity
                  com.example.obd2.MainActivity

All Implemented Interfaces:

android.content.ComponentCallbacks, android.content.ComponentCallbacks2, android.view.KeyEvent.Callback, android.view.LayoutInflater.Factory, android.view.LayoutInflater.Factory2, android.view.View.OnCreateContextMenuListener, android.view.Window.Callback, androidx.activity.contextaware.ContextAware, androidx.activity.OnBackPressedDispatcherOwner, androidx.activity.result.ActivityResultCaller, androidx.activity.result.ActivityResultRegistryOwner, androidx.appcompat.app.ActionBarDrawerToggle.DelegateProvider, androidx.appcompat.app.AppCompatCallback, androidx.core.app.ActivityCompat.OnRequestPermissionsResultCallback, androidx.core.app.ActivityCompat.RequestPermissionsRequestCodeValidator, androidx.core.app.OnMultiWindowModeChangedProvider, androidx.core.app.OnNewIntentProvider, androidx.core.app.OnPictureInPictureModeChangedProvider, androidx.core.app.TaskStackBuilder.SupportParentable, androidx.core.content.OnConfigurationChangedProvider, androidx.core.content.OnTrimMemoryProvider, androidx.core.view.KeyEventDispatcher.Component, androidx.core.view.MenuHost, androidx.lifecycle.HasDefaultViewModelProviderFactory, androidx.lifecycle.LifecycleOwner, androidx.lifecycle.ViewModelStoreOwner, androidx.savedstate.SavedStateRegistryOwner

---

```
public class MainActivity
extends androidx.appcompat.app.AppCompatActivity
```

Mainaktivität mit runuithread

lädt Drehzahlanzeige (Bild)

ruft die Klassen auf und zeichnet die Tachonadel

Auswahl per ELM Emulator oder Bluetooth

Field Summary

Fields

Modifier and Type	Field	Description
static boolean	bluetooth_start	
static android.content.Context	cc	
static int	dstart	
static boolean	emulator_start	
static java.lang.String	ipp	
static OBD_Bluetooth	obd_bluetooth	

enforceCallingUriPermission, enforcePermission, enforceUriPermission, enforceUriPermission, fileList, getApplicationContext, getApplicationInfo, getAttributionSource, getAttributionTag, getBaseContext, getCacheDir, getClassLoader, getCodeCacheDir, getContentResolver, getDatabasePath, getDataDir, getDir, getDisplay, getExternalCacheDir, getExternalCacheDirs, getExternalFilesDir, getExternalFilesDirs, getExternalMediaDirs, getFilesDir, getFileStreamPath, getMainExecutor, getMainLooper, getNoBackupFilesDir, getObbDir, getObbDirs, getOpPackageName, getPackageCodePath, getPackageManager, getPackageName, getPackageResourcePath, getParams, getSharedPreferences, getSystemServiceName, grantUriPermission, isDeviceProtectedStorage, isRestricted, isUiContext, moveDatabaseFrom, moveSharedPreferencesFrom, openFileInput, openFileOutput, openOrCreateDatabase, openOrCreateDatabase, registerReceiver, registerReceiver, registerReceiver, registerReceiver, revokeUriPermission, revokeUriPermission, sendBroadcast, sendBroadcast, sendBroadcastAsUser, sendBroadcastAsUser, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcastAsUser, startForegroundService, startInstrumentation, startService, stopService, unbindService, unregisterReceiver, updateServiceGroup

Methods inherited from class android.content.Context

getColor, getColorStateList, getDrawable, getString, getString, getSystemService, getText, obtainStyledAttributes, obtainStyledAttributes, obtainStyledAttributes, obtainStyledAttributes, registerComponentCallbacks, sendBroadcastWithMultiplePermissions, unregisterComponentCallbacks

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface androidx.core.app.ActivityCompat.RequestPermissionsRequestCodeValidator

validateRequestPermissionsRequestCode

Methods inherited from interface android.view.Window.Callback

onPointerCaptureChanged

Field Detail

dstart

public static int dstart

emulator_start

public static boolean emulator_start

bluetooth_start

```
public static boolean bluetooth_start
```

ipp

```
public static java.lang.String ipp
```

obd_bluetooth

```
public static OBD_Bluetooth obd_bluetooth
```

cc

```
public static android.content.Context cc
```

oil

```
public static int oil
```

water

```
public static int water
```

Constructor Detail

MainActivity

```
public MainActivity()
```

Package com.example.obd2

Class OBD_Befehle

java.lang.Object
    com.example.obd2.OBD_Befehle

```
public class OBD_Befehle
extends java.lang.Object
```

Socket Test mit Elm Emulator auf port 3[redacted]

Field Summary

Fields

Modifier and Type	Field	Description
static int	Drehzahl1	interne Drehzahl
static int[]	intArray	Mittelwert

Constructor Summary

Constructors

Constructor	Description
OBD_Befehle()	

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description
int	Socket1(java.lang.String ip, java.lang.String cmd)	

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

Drehzahl1

```
public static int Drehzahl1
```

interne Drehzahl

intArray

```
public static int[] intArray
```

Mittelwert

Constructor Detail

OBD_Befehle

```
public OBD_Befehle()
```

Method Detail

Socket1

```
public int Socket1(java.lang.String ip, java.lang.String cmd)
```

Package com.example.obd2

Class OBD_Bluetooth

java.lang.Object
    com.example.obd2.OBD_Bluetooth

```
public class OBD_Bluetooth
extends java.lang.Object
```

Bluetooth Klasse mit Thread
speichert die Anezigewerte in lokale Variablen , die dann vom Main UI Thread ausgelesen werden : runOnUiThread

Nested Class Summary

Nested Classes

Modifier and Type	Class	Description
class	OBD_Bluetooth.BThread	Btread Klasse - schickt und liest messages alle 100ms...

Field Summary

Fields

Modifier and Type	Field	Description
static android.bluetooth.BluetoothDevice	device	
static android.bluetooth.BluetoothAdapter	mBluetoothAdapter	interen Adapter
static int	mm_rpm	interne Drehzahl
static int	mm_water	interne Wassertemperatur
java.io.InputStream	mmInStream	lese-stream zu ELM Adapter
java.io.OutputStream	mmOutStream	write-stream zu ELM Adapter
static android.bluetooth.BluetoothSocket	mmSocket	

Constructor Summary

Constructors

Constructor	Description
`OBD_Bluetooth()`	

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description
int	`init(android.content.Context cc)`	Init Funktion
int	`readBsocket(java.lang.String cmd)`	readBsocket Funktion

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

mBluetoothAdapter

```
public static android.bluetooth.BluetoothAdapter mBluetoothAdapter
```

interen Adapter

device

```
public static android.bluetooth.BluetoothDevice device
```

mmSocket

```
public static android.bluetooth.BluetoothSocket mmSocket
```

mmInStream

```
public java.io.InputStream mmInStream
```

lese-stream zu ELM Adapter

mmOutStream

```
public java.io.OutputStream mmOutStream
```

write-stream zu ELM Adapter

mm_rpm

```
public static int mm_rpm
```

interne Drehzahl

mm_water

```
public static int mm_water
```

interne Wassertemperatur

Constructor Detail

OBD_Bluetooth

```
public OBD_Bluetooth()
```

Method Detail

init

```
public int init(android.content.Context cc)
```

Init Funktion

- sucht Default Adapter

- sucht verbundede Devices

- started den Bluetooth Thread

Parameters:

`cc` - Kontext für Anzeige UI Thread

Returns:

0:failed , 1:OK

readBsocket

```
public int readBsocket(java.lang.String cmd)
```

readBsocket Funktion

- sucht Default Adapter

- sucht verbundede Devices

Parameters:

`cmd` - OBD Kommando Beispiel 010C für Drehzahl

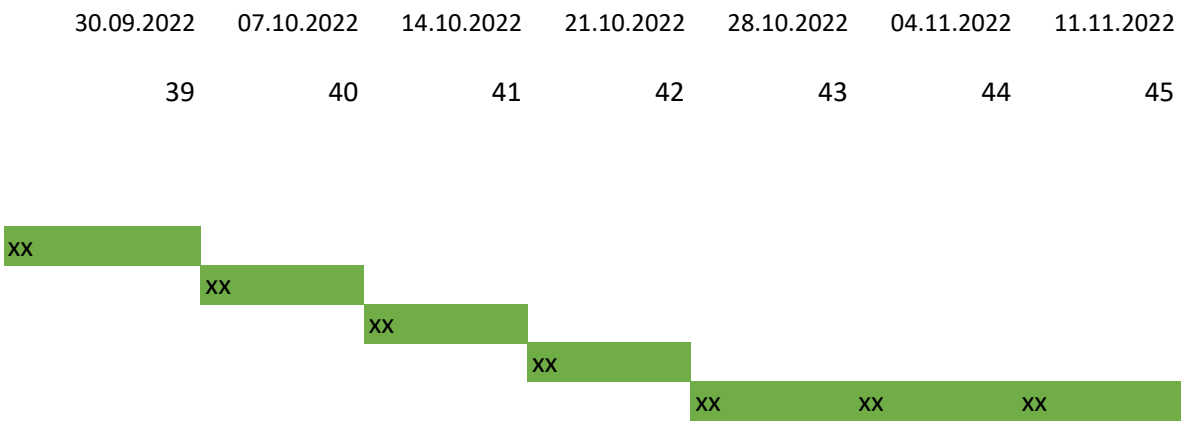
Returns:

1 :ok 0. failed

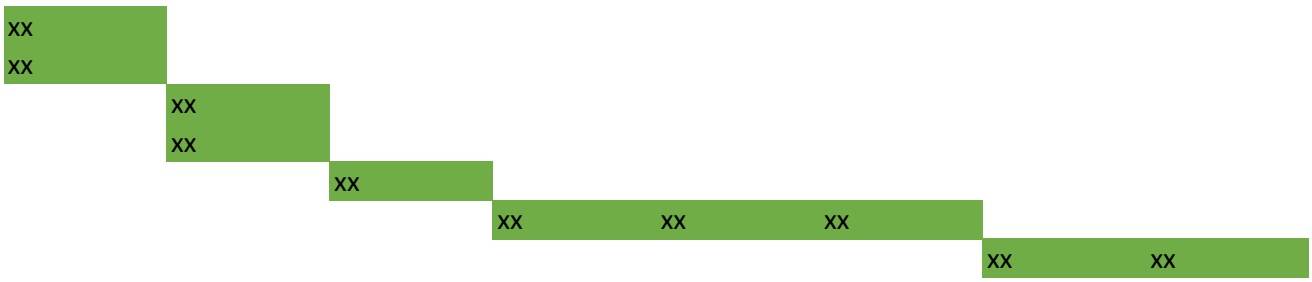
Seminarkurs

Projekttitel: OBD Anzeige

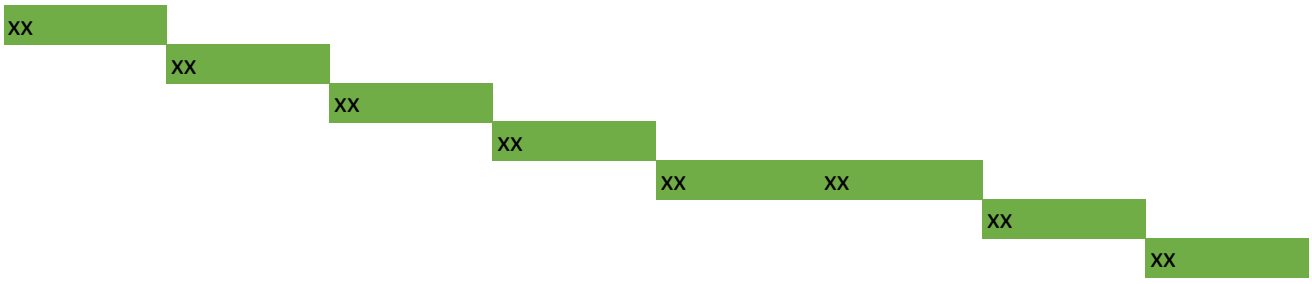
	Termine			23.09.2022
	Gemacht	Nicht Gemacht		
	KW			38
	Teilaufgaben:	**Starttermin**	**Ende**	
1	*Lehrerabsprache*			xx
2	*Ideensammlung erweitern*			xx
3	*Programmieren anfang*			
4	*Anzeige*			
5	*Drehzahlanzeige*			
6	*kmh anzeige*			
7	*evt. Erweiterung*			
8	*Testen evt. Am auto*			
9	*eine handy Datei draus machen*			
10	*optimieren*			
11	*Anzeige Testen*			
12	***Fehlersuche***			
13	*Ausbessern*			
14	2te meinung holen			
15	*Text schreiben*			
16	*Kontrolle*			
17	*Powerpoint*			
18	*Vortragbar machen*			
19	*Üben*			
20	*Mit KFZ meister sprechen*			
21	*Endkontrolle*			
22	***Abgabe***			
23				
24				
25				
26				
27				
28				
29				
30				
31				
32				
33				
34				
35				
36				
37				
38				
39				


30.09.2022
07.10.2022
14.10.2022
21.10.2022
28.10.2022
04.11.2022
11.11.2022
39
40
41
42
43
44
45
xx
xx
xx
xx
xx
xx
xx

18.11.2022	25.11.2022	02.12.2022	09.12.2022	16.12.2022	23.12.2022	30.12.2022	06.01.2023
46	47	48	49	50	51	52	53



13.01.2023	20.01.2023	27.01.2023	03.02.2023	10.02.2023	17.02.2023	24.02.2023	03.03.2023
54	55	56	57	58	59	60	61



10.03.2023	17.03.2023	24.03.2023	31.03.2023	07.04.2023	14.04.2023	21.04.2023	28.04.2023
62	63	64	65	66	67	68	69

05.05.2023	12.05.2023	19.05.2023	26.05.2023	02.06.2023	09.06.2023	16.06.2023	23.06.2023
70	71	72	73	74	75	76	77
		ABGABE					
	xx						

```java
package com.example.obd2;

import androidx.appcompat.app.AppCompatActivity;
import android.annotation.SuppressLint;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.res.ColorStateList;
import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.os.Bundle;
import android.os.StrictMode;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

import java.util.Timer;
import java.util.TimerTask;



/**
 * Mainaktivität mit runuithread
 * <p> lädt Drehzahlanzeige ( Bild)
 * <p>ruft die Klassen auf und zeichnet die Tachonadel
 * <p>Auswahl per ELM Emulator oder Bluetooth
 *
 * */
public class MainActivity extends AppCompatActivity {
    /**Gesetzter Wert für Tachonadel (FEST)(Im weiteren sind sie Variabel)*/
    public static int rpm=1000;
    /**Auswahl des Emulator startes*/
    public static boolean emulator_start=false;
    /**Auswahl des Bluetooth startes*/
    public static boolean bluetooth_start=true;
    /**Bluetooth object */
    public static OBD_Bluetooth obd_bluetooth=null;
    /**Mainactivity Context  */
    public static Context cc =null;
    /**Wassertemperatur -40 bis 215 Grad*/
    public static int watertemp=0;
    /**Geschwindigkeit von 0 bis 255 KMH*/
    public static int speed=0;
    /**Lufttemparatur von -40 bis 215 Grad*/
    public static int airtemp=0;
    /**Motorbelastung von 0 bis 100%*/
    public static int engload = 0;
    public static ColorStateList ctt=null;
    public static Paint mPaint;



//UI
    @SuppressLint("MissingInflatedId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll
().build();
        StrictMode.setThreadPolicy(policy);

        //Kontext Rückgreifpunkt
```

```
67          cc = this;
68
69          View view = findViewById(android.R.id.content).getRootView();
70          view.setOnTouchListener(new OnSwipeTouchListener(cc) {
71              @Override
72              public void onSwipeLeft() {
73                  // TerminalTest = Toast.makeText(cc, "left" , Toast.LENGTH_LONG).show();
74                  setContentView(R.layout.secondlayout);
75              }
76              public void onSwipeRight() {
77                  //TerminalTest Toast.makeText(cc, "right" , Toast.LENGTH_LONG).show();
78                  setContentView(R.layout.activity_main);
79              }
80          });
81
82          //Bluetooth start siehe oben. (wechsel zwischen Bluetooth und Computer)
83          if (bluetooth_start) {
84             obd_bluetooth = new OBD_Bluetooth();
85
86
87              int ret = obd_bluetooth.init(  cc);
88              //() -> OBD_Bluetooth.HandlerThread
89
90          }
91
92          //Verzögerung für flüssigere Ui
93          Timer timerObj = new Timer(true);
94          TimerTask timerTaskObj = new TimerTask() {
95              public void run() {
96                  runOnUiThread(new Runnable() {
97                      public void run() {
98
99
100                         //Zugriff auf das Emullierte Auto
101                         if (emulator_start) {
102                             OBD_Befehle obd_befehle = new OBD_Befehle();
103                             String ip= "[redacted]"; //Ip muss festgesetzt werden am
    Gerät des Autos-Emulator
104                             obd_befehle.Socket1(ip,"010C"); //Testbefehle 010C=
    Drehzahlen
105                             rpm = obd_befehle.Drehzahl1; //Siehe OBD Befehle
106
107                         }
108
109                         //Zugriff auf den Bluetooth Adapter
110                         if (bluetooth_start) {
111
112
113                             rpm = obd_bluetooth.mm_rpm; //Siehe OBD_Bluetooth
114                             speed = obd_bluetooth.mm_speed;
115                             watertemp = obd_bluetooth.mm_water;
116                             airtemp = obd_bluetooth.mm_airtmp;
117                             engload = obd_bluetooth.mm_engload;
118
119
120                         }
121
122                         double drehzahl = rpm / 61.11; // 11.000 rpm = 180grad   61.11=
    11.000/180
123                         double geschwindigkeit = speed / 1.55; // 280 Km/h = 180grad   1
    .55=280/180
124
125
126
127                         //2te seite
128                         TextView t6 = findViewById(R.id.id_hello6); //Textfelder auf der
     2ten Seite
```

```
129                         TextView t7 = findViewById(R.id.id_hello7);
130                         TextView t8 = findViewById(R.id.id_hello8);
131                         TextView t9 = findViewById(R.id.id_hello9);
132                         TextView t10 = findViewById(R.id.id_hello10);
133
134
135                         //zweite Seite Farben für Warnung des Motors
136                         if(t6!=null) {
137
138                             if (ctt == null) ctt = t6.getTextColors(); //init
139
140
141                             if (watertemp > 70) {
142                                 t7.setTextColor(Color.GREEN);
143                             }
144                             if (watertemp > 110) {
145                                 t7.setTextColor(Color.RED);
146                             } else {t7.setTextColor(ctt);}
147
148                             if (airtemp > 50) {
149                                 t8.setTextColor(Color.RED);
150                             } else {t8.setTextColor(ctt);}
151
152                             if (engload > 90) {
153                                 t9.setTextColor(Color.RED);
154                             } else {t9.setTextColor(ctt);}
155
156                             if (rpm > 3500) {
157                                 t10.setTextColor(Color.RED);
158                             }else {t10.setTextColor(ctt);}
159
160                             t6.setText("Geschwindigkeit : " + speed + " KM/h ");
161                             t7.setText("Wasser : " + watertemp + " Grad Celsius ");
162                             t8.setText("Luft : " + airtemp + " Grad Celsius ");
163                             t9.setText("Motorlast : " + engload + " % ");
164                             t10.setText("Drehzahlen : " + rpm + " RPM ");
165
166                             t6.setTextSize(30);
167                             t7.setTextSize(30);
168                             t8.setTextSize(30);
169                             t9.setTextSize(30);
170                             t10.setTextSize(30);
171                         }
172                         else
173                         {
174
175
176                             //Geschwindigkeitsanzeige
177                             ImageView va = findViewById(R.id.imageView2);
178                             Log.d("DEBUG", "" + va.getMeasuredWidth() + ", " + va.
    getMeasuredHeight()); //Zentrierung der Tachonadel durch die Erkennung der Mitte des
    Lichtes
179
180                             int wi = va.getMeasuredWidth();
181                             int he = va.getMeasuredHeight();
182                             double radi = he;
183                             int dx = (int) (radi * Math.cos(Math.toRadians(drehzahl)));
184                             int dy = (int) (radi * Math.sin(Math.toRadians(drehzahl)));
185
186                             if (wi > 0) {
187
188
189                                 Bitmap b = Bitmap.createBitmap(va.getMeasuredWidth(), va
    .getMeasuredHeight(), Bitmap.Config.ARGB_8888);
190                                 Canvas c = new Canvas(b);
191                                 Paint paint = new Paint();
192                                 paint.setColor(Color.BLACK);
```

```
193                                 paint.setStyle(Paint.Style.STROKE);
194                                 paint.setStrokeWidth(8);
195                                 paint.setAntiAlias(true);
196                                 //c.drawLine(200, 200, 500, 500, paint);
197                                 c.drawLine(wi / 2 - dx, he - dy, wi / 2, he, paint);
198
199                                 va.setImageBitmap(b);
200                                 va.postInvalidate();
201
202                                     ImageView va1 = findViewById(R.id.imageView4);
203
204                                     Log.d("DEBUG", "" + va1.getMeasuredWidth() + ", " +
    va1.getMeasuredHeight());
205
206
207                                     int wi1 = va1.getMeasuredWidth();
208                                     int he1 = va1.getMeasuredHeight();
209                                     double radi1 = he1;
210                                     int dx1 = (int) (radi1 * Math.cos(Math.toRadians(
    geschwindigkeit)));
211                                     int dy1 = (int) (radi1 * Math.sin(Math.toRadians(
    geschwindigkeit)));
212
213                                     if (wi1 > 0) {
214
215
216                                         Bitmap b1 = Bitmap.createBitmap(va1.
    getMeasuredWidth(), va1.getMeasuredHeight(), Bitmap.Config.ARGB_8888);
217                                         Canvas c1 = new Canvas(b1);
218                                         Paint paint1 = new Paint();
219                                         paint1.setColor(Color.BLACK);
220                                         paint1.setStyle(Paint.Style.STROKE);
221                                         paint1.setStrokeWidth(8);
222                                         paint1.setAntiAlias(true);
223                                         c1.drawLine(wi1 / 2 - dx1, he1 - dy1, wi1 / 2,
    he1, paint1);
224
225                                         va1.setImageBitmap(b1);
226                                         va1.postInvalidate();
227
228
229
230                                         ImageView va2 = findViewById(R.id.imageView2);
231
232                                         int wi2 = va2.getMeasuredWidth();
233                                         int he2 = va2.getMeasuredHeight();
234                                         double radi2 = he2;
235
236                                         if (wi2 > 0) {
237
238
239                                             //Bitmap b2 = Bitmap.createBitmap(va2.
    getMeasuredWidth(), va2.getMeasuredHeight(), Bitmap.Config.ARGB_8888);
240                                             Canvas c2 = new Canvas(b);
241                                             Paint paint2 = new Paint();
242                                             if (rpm <= 2000 && engload >= 85){
243                                                 paint2.setColor(Color.RED);
244                                             }else {
245                                                 paint2.setColor(Color.GRAY);
246                                             }
247                                             paint2.setStyle(Paint.Style.STROKE);
248                                             paint2.setStrokeWidth(70);
249                                             paint2.setAntiAlias(true);
250                                             c2.drawLine(290, 450, 360, 450, paint2);
251
252
253                                             //va2.setImageBitmap(b1);
```

```
254                                         //va2.postInvalidate();
255
256
257                                         Canvas c3 = new Canvas(b);
258                                         Paint paint3 = new Paint();
259                                         if (rpm >= 2500 && engload >= 85 || rpm >=
    2500 && engload <= 35){
260                                             paint3.setColor(Color.GREEN);
261                                         }else {
262                                             paint3.setColor(Color.GRAY);
263                                         }
264                                         paint3.setStyle(Paint.Style.STROKE);
265                                         paint3.setStrokeWidth(70);
266                                         paint3.setAntiAlias(true);
267                                         c3.drawLine(690, 450, 760, 450, paint3);
268                                     }
269                                 }
270
271                             }
272                         }
273                     }//run
274                 });//perform your action here
275             };
276         };
277         timerObj.schedule(timerTaskObj, 0, 100);
278
279
280
281         Button button = (Button) findViewById(R.id.B_Start); // Not start
282         button.setOnClickListener(new View.OnClickListener() {
283             public void onClick(View v) {
284                 // Do something in response to button click
285                 AlertDialog alertDialog = new AlertDialog.Builder(MainActivity.this).
    create();
286                 alertDialog.setTitle("Alert");
287                 alertDialog.setButton(AlertDialog.BUTTON_NEUTRAL, "OK",
288                         new DialogInterface.OnClickListener() {
289                             public void onClick(DialogInterface dialog, int which) {
290                                 dialog.dismiss();
291                             }
292                         });
293
294                 if (bluetooth_start) { //Not start
295                     int ret = obd_bluetooth.init(cc);
296                 }
297
298
299                 Log.d("Info", "This is my message"); //Terminal check
300
301
302                 if (emulator_start) {
303                     OBD_Befehle obd_befehle = new OBD_Befehle();
304                     String ip= "[redacted]";
305                     obd_befehle.Socket1(ip,"010C");
306                     rpm = obd_befehle.Drehzahl1;
307                 }
308
309             }
310
311         });
312
313     }
314
315 }//oncreate
316
317
```

[redacted]obd2\OBD_Bluetooth.java

```java
package com.example.obd2;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.widget.Toast;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.nio.charset.StandardCharsets;
import java.util.Set;
import java.util.UUID;
import android.widget.Toast;

/**
 * Bluetooth Klasse mit Thread
 * <br> speichert die Anezigewerte in lokale Variablen , die dann vom
 * <br> Main UI Thread ausgelesen werden : runOnUiThread
 * */
public class OBD_Bluetooth {
    /**interner Adapter*/
    public static BluetoothAdapter BlueAdap = null;
    /**Adapter erkennung verbindung*/
    public static BluetoothDevice device = null;
    /**Absoluter erkennungscode*/
    private static final UUID MY_UUID = UUID.fromString("[redacted]");
    /**Verbinden zum bluetooth Socket*/
    public static BluetoothSocket mmSocket = null;
    /**lese-stream zu ELM Adapter*/
    public InputStream mmInStream = null;
    /**write-stream zu ELM Adapter*/
    public OutputStream mmOutStream = null;
    /**interne Drehzahl*/
    public static int mm_rpm = 0;
    /**interne Geschwindigkeit*/
    public static int mm_speed = 0;
    /**interne Wassertemperatur*/
    public static int mm_water = 0;
    /**interne Lufttemparatur*/
    public static int mm_airtmp = 0;
    /**interne Motorbelastung*/
    public static int mm_engload = 0;

    /**
    * Init Funktion
     * <p> - sucht Default Adapter
     * <p> - sucht verbundede Devices
     * <p> - started den Bluetooth Thread
     * @param cc Kontext für Anzeige UI Thread
     * @return 0: failed , 1: OK
     * */
    public int init(Context cc) {
        BlueAdap = BluetoothAdapter.getDefaultAdapter();

        if (BlueAdap == null) {
            Toast.makeText(cc, "Bluetooth is not available", Toast.LENGTH_LONG).show();
//Textbalken fürs Handy zum erkennen kleiner Fehler (Bluetooth=aus) Diese Nachricht
            return 0;
        }

```

```
66
67          if (!BlueAdap.isEnabled()) {
68              Intent enableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
69              Toast.makeText(cc, "Bluetooth is not available", Toast.LENGTH_LONG).show();
70              return 0;
71
72          }
73
74
75          Set<BluetoothDevice> all_devices = BlueAdap.getBondedDevices();
76          if (all_devices.size() > 0) {
77              for (BluetoothDevice currentDevice : all_devices) {
78                  Log.d("DEBUG", "" + currentDevice.getName());
79
80                  if (currentDevice.getName().toLowerCase().endsWith("obdii")) { // Name
    des Bluetooth Gerät
81                      BluetoothSocket tmp = null;
82
83                      String mac = currentDevice.getAddress();
84                      Toast.makeText(   cc, "obd found" + mac, Toast.LENGTH_LONG).show();
85                      device = BlueAdap.getRemoteDevice(mac);
86                      try {
87                          tmp = device.createRfcommSocketToServiceRecord(MY_UUID);
88                      } catch (IOException e) {
89                          Log.e("ERR", "Socket Type: create() failed", e);
90                      }
91                      mmSocket = tmp;
92                      if (tmp == null)
93                          Toast.makeText(cc, "obd create nok" + mac, Toast.LENGTH_LONG).
    show();
94                  }
95              }
96          }
97
98          if (mmSocket != null) {
99              // Stellt verbindung zum Bluetooth Socket
100             try {
101                 mmSocket.connect();
102             } catch (IOException e) {
103
104                 try {
105                     mmSocket.close();
106                     Log.d("ERR", " close() socket during connection failure", e);
107                 } catch (IOException e2) {
108                     Log.d("ERR", "unable to close() socket during connection failure",
    e2);
109                 }
110             ;
111                 return 0;
112             }
113             Toast.makeText(cc, "obd connect ok", Toast.LENGTH_LONG).show();
114         } else {
115             Toast.makeText(cc, "obd connect not ok", Toast.LENGTH_LONG).show();
116
117             return 0;
118
119         }
120
121         InputStream tmpIn = null;
122         OutputStream tmpOut = null;
123         if (mmSocket != null) {
124
125             try {
126                 tmpIn = mmSocket.getInputStream();
127                 tmpOut = mmSocket.getOutputStream();
128             } catch (IOException e) {
129                 Log.d("ERR", "temp sockets not created", e);
```

```
130             }
131
132             mmInStream = tmpIn;
133             mmOutStream = tmpOut;
134             if (mmInStream != null && mmOutStream != null)
135                 Toast.makeText(cc, "obd io ok", Toast.LENGTH_LONG).show();
136         }
137
138         Thread b = new BThread();
139         b.start();
140         return 1;
141     }//init
142
143     /**
144      * Btread Klasse
145      * <br> - schickt und liest messages alle 100ms...
146      *
147      * */
148     public class BThread extends Thread {
149
150         public void run() {
151
152             while(true) {
153                 mm_rpm = readBsocket("010C"); // Obd befehle für Abfrage der Werte
154                 mm_speed = readBsocket("010D");
155                 mm_water = readBsocket("0105");
156                 mm_airtmp = readBsocket("010F");
157                 mm_engload = readBsocket("0104");
158                 //mm_rpm = 1000; Testwert
159                 Log.d("ERR", "This is my msg----------------------------" );
160                 try {
161                     sleep(100); // 1 von 2 GUIs
162                 } catch (InterruptedException e) {
163                     e.printStackTrace();
164                 }
165             }
166
167         }
168
169     }
170
171
172     /**
173      * readBsocket Funktion
174      * <p> - sucht Default Adapter
175      * <p> - sucht verbundede Devices
176      * @param cmd OBD Kommando Beispiel 010C für Drehzahl
177      * @return 1: ok , 0: failed
178      * */
179     public int readBsocket(String cmd) {
180
181         if(mmOutStream==null){return 0;}
182
183         try {
184             mmOutStream.write(cmd.getBytes(StandardCharsets.UTF_8));
185         } catch (IOException e) {
186             //e.printStackTrace();
187             Log.d("ERR", "This is my err"+e);
188             mm_airtmp=0;
189             mm_engload=0;
190             mm_rpm=0;
191             mm_speed=0;
192             mm_water=0;
193         }
194
195         String s,msg="";
196         while (true) {
```

```
197             try {
198                 byte[] buffer = new byte[1];
199                 int bytes = mmInStream.read(buffer, 0, buffer.length);
200                 s = new String(buffer);
201                 for (int i = 0; i < s.length(); i++) {
202                     char x = s.charAt(i);
203                     msg = msg + x;
204                     if (x == 0x3e) {  // Ist das größer Zeichen <
205                         //mHandler.obtainMessage(OBDActivity.MESSAGE_READ, buffer.length
    , -1, msg).sendToTarget();
206                         //msg=""; Debugging Nachrichten
207                         Log.d("ERR", "This is my msg" + msg);
208 
209                         //rpm 010C41 0C 0F 9C  < Testwert
210                         // oil 010541 05 6A Testwert
211                         if (msg.length() > 14 && msg.contains("010C") && !msg.contains("
    SEARCHING")) { //rpm 14= länge der Hexzahl
212                             msg = msg.substring(10, 15).trim().replace(" ", ""); //
    Kürzt die Hex auf das nötigste herunter
213                             //Toast.makeText(this, "obd:"+msg, Toast.LENGTH_LONG).show
    ();
214                             return Integer.decode("0x" + msg) / 4; // Werte welche
    Wichtig für das Problem in der Ui ist
215                         }
216                         if (msg.length() > 11 && msg.contains("010D")) {  //speed
217                             msg = msg.substring(10, 12).trim().replace(" ", "");
218                             //Toast.makeText(this, "obd:"+msg, Toast.LENGTH_LONG).show
    ();
219                             return Integer.decode("0x" + msg);
220                         }
221                         if (msg.length() > 11 && msg.contains("0105")) {  //
    watertemperatur
222                             msg = msg.substring(10, 12).trim().replace(" ", "");
223                             //Toast.makeText(this, "obd:"+msg, Toast.LENGTH_LONG).show
    ();
224                             return Integer.decode("0x" + msg) - 40;
225                         }
226 
227                         if (msg.length() > 11 && msg.contains("010F")) {  //air
    temperatur
228                             msg = msg.substring(10, 12).trim().replace(" ", "");
229                             //Toast.makeText(this, "obd:"+msg, Toast.LENGTH_LONG).show
    ();
230                             return Integer.decode("0x" + msg) - 40;
231                         }
232 
233                         if (msg.length() > 11 && msg.contains("0104")) {  //engineload
234                             msg = msg.substring(10, 12).trim().replace(" ", "");
235                             //Toast.makeText(this, "obd:"+msg, Toast.LENGTH_LONG).show
    ();
236                             return Integer.decode("0x" + msg) * 100 / 255;
237                         }
238 
239                         return 1;
240                     }
241                 }
242 
243             } catch(IOException e){
244                 mmOutStream = null;
245                 Log.d("ERR", "This is my msg err" + e);
246                 mm_airtmp=0;
247                 mm_engload=0;
248                 mm_rpm=0;
249                 mm_speed=0;
250                 mm_water=0;
251                 break;
252             }
```

```
253         }//while
254         return 0; // Problem
255     }
256
257 }
258
```

```java
package com.example.obd2;

import android.util.Log;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.UnknownHostException;


/**Socket Test mit Elm Emulator auf port 35000*/
public class OBD_Befehle {
    /** interne Drehzahl*/
    public static int Drehzahl1=0 ;
    /** Mittelwert*/
    public static int[] intArray = new int[100];


    // Code welchen man in dem Emulator eingeben muss um z.B. die Drehzahlen einzustellen
//emulator.answer['RPM'] = '<exec>ECU_R_ADDR_E + " 04 41 0C %.4X" % int(4 * 500)</exec><
writeln />'
//https://github.com/Ircama/ELM327-emulator
//c:\elm\elm :  elm -n 35000

    /**
     * Socket funktion für den python Emulator auf port 35000
     * <p>bildet mittelwert aus 10 werten und speichert es in Drehzahl1
     * @param ip: IP Adresse des Emulators
     * @param cmd : OBD Komanndo , Beispiel 010C für Drehzahl
     * @return 1: ok , 0: failed
     * */
    public int Socket1(String ip,String cmd) {
        String text1 = "";

        try {

            Log.i("Info", "start obd:"+cmd);


            Socket s = new Socket(ip, 35000); // Port für Zugriff auf Laptop
            if(!s.isConnected()){
                Log.i("Info", "start obd out:"+cmd);
                return 0;
            }
            PrintWriter out =   new PrintWriter(s.getOutputStream(), true);
            InputStreamReader bs = new InputStreamReader(s.getInputStream());
            out.println(cmd);
            BufferedReader reader = new BufferedReader(bs);

            String line;
            while ((line = reader.readLine()).length() != 0) {
                Log.i("Info", "start obd out:"+cmd);
                text1 += line;

                //für tests mit powershell reader
                if(line.length()==21) {break;}

            }
            Log.i("Info", "stop obd"+text1);
            reader.close();
            s.close();

        }
        catch (UnknownHostException ex) {

```

```
67              Log.i("Info", "This is my message1 "+ ex.getMessage());
68          }
69          catch (IOException ex) {
70
71              Log.i("Info", "This is my message2 "+ ex.getMessage());
72
73              if (Drehzahl1 > 0) {Drehzahl1 = 0;         return 1;}
74          }
75
76          if(text1.length()>17 && text1.contains("0142")) {
77              System.out.print(text1);
78          }
79              if(text1.length()>17 && text1.contains("010C")) {
80              text1 = text1.substring(17);
81
82
83              int Drehzahl2 = Integer.decode("0x" + text1) / 4;
84
85                  //010C7E8 04 41 0D 3E80
86                  //0100441 0D 3E80
87                  //010C41 0C 14 5F
88
89
90
91              // mittelwert aus 10 werten.
92              if (Drehzahl2 != Drehzahl1) {
93
94                  int sum = 0;
95                  //mittelwert...
96                  for (int i = 0; i < 9; i++) {
97                      intArray[i] = intArray[i + 1];
98                      intArray[9] = Drehzahl2;
99                  }
100                 for (int i = 0; i < 10; i++) {
101                     sum += intArray[i];
102                 }
103                 Drehzahl1 = sum / 10;
104
105                 Log.i("Info", text1 + ":" + Drehzahl2 + "\n");
106
107                 return 1;
108             }
109         }
110         return 0;
111
112     }
113
114 }
115
```

```java
package com.example.obd2;

import android.content.Context;
import android.view.GestureDetector;
import android.view.GestureDetector.SimpleOnGestureListener;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;

/**
 * Erkennt links und Recht Swipes und kann dadurch die zwei Ui wechseln.
 */
public class OnSwipeTouchListener implements OnTouchListener {

    private final GestureDetector gestureDetector;

    public OnSwipeTouchListener(Context context) {
        gestureDetector = new GestureDetector(context, new GestureListener());
    }

    public void onSwipeLeft() {
    }

    public void onSwipeRight() {
    }

    public boolean onTouch(View v, MotionEvent event) {
        return gestureDetector.onTouchEvent(event);
    }

    private final class GestureListener extends SimpleOnGestureListener {

        private static final int SWIPE_DISTANCE_THRESHOLD = 100;
        private static final int SWIPE_VELOCITY_THRESHOLD = 100;

        @Override
        public boolean onDown(MotionEvent e) {
            return true;
        }

        @Override
        public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY) {
            float distanceX = e2.getX() - e1.getX();
            float distanceY = e2.getY() - e1.getY();
            if (Math.abs(distanceX) > Math.abs(distanceY) && Math.abs(distanceX) > SWIPE_DISTANCE_THRESHOLD && Math.abs(velocityX) > SWIPE_VELOCITY_THRESHOLD) {
                if (distanceX > 0)
                    onSwipeRight();
                else
                    onSwipeLeft();
                return true;
            }
            return false;
        }
    }
}
```