**Henning Groß, Beuth Hochschule für Technik Berlin. Studiengang Technische Informatik Bachelor, 749992**

WS2012

**Bachelorarbeit**

**Einrichtung zur Erfassung und Visualisierung von Drehzahl, Geschwindigkeit und Abgastemperatur klassischer Motorroller**

**Henning Groß**

Inhaltsverzeichnis

[1. Selbstständigkeitserklärung 4](#_Toc341536656)

[2. Copyright 5](#_Toc341536657)

[3. Einleitung 6](#_Toc341536658)

[4. Aufgabenstellung, fachliche Anforderungen/ Rahmen- und Umgebungsbedingungen 8](#_Toc341536659)

[4.1. Messeinrichtung 8](#_Toc341536660)

[4.1.1. Wahl einer Plattform zur Erfassung der Messdaten 8](#_Toc341536661)

[4.1.2. Entwicklung der Messschaltungen, Erfassen der Signale/Messdaten 8](#_Toc341536662)

[4.1.2.1. Drehzahl 8](#_Toc341536663)

[4.1.2.2. Weg/Zeit (Tacho) 8](#_Toc341536664)

[4.1.2.3. Abgastemperatur 9](#_Toc341536665)

[4.2. Auswertung und Darstellung 9](#_Toc341536666)

[4.2.1. Wahl einer Plattform zur Auswertung und Darstellung der Messdaten 9](#_Toc341536667)

[4.2.2. Entwicklung der Algorithmen zur Verarbeitung der Messdaten 9](#_Toc341536668)

[4.2.3. Visualisierung der Messdaten 10](#_Toc341536669)

[4.3. Realisierung der Kommunikation zwischen Mess- und Auswertungseinrichtung 10](#_Toc341536670)

[4.3.1. Wahl eines Übertragungsverfahrens 10](#_Toc341536671)

[4.3.2. Entwicklung eines Datenformates und Protokolls 10](#_Toc341536672)

[5. Lösungsansätze 11](#_Toc341536673)

[5.1. Messeinrichtung 11](#_Toc341536674)

[5.1.1. Drehzahl 11](#_Toc341536675)

[5.1.1.1. Hall-Effekt 11](#_Toc341536676)

[5.1.1.2. Induktion am Zündkabel 13](#_Toc341536677)

[5.1.1.3. Lichtmaschine 14](#_Toc341536678)

[5.1.2. Tachosignal 16](#_Toc341536679)

[5.1.2.1. Reedkontakt 16](#_Toc341536680)

[5.1.2.2. Rechnerisch 17](#_Toc341536681)

[5.1.2.3. Hall-Effekt 17](#_Toc341536682)

[5.1.3. Abgastemperatur 18](#_Toc341536683)

[5.1.3.1. Umsetzung mit Analogschaltung 19](#_Toc341536684)

[5.1.3.2. IC 19](#_Toc341536685)

[5.2. Signalverarbeitung 21](#_Toc341536686)

[5.2.1. Drehzahl / Tacho 21](#_Toc341536687)

[5.2.1.1. Beispiele 21](#_Toc341536688)

[5.2.2. Abgastemperatur 22](#_Toc341536689)

[5.3. Datenübertragung 23](#_Toc341536690)

[5.3.1. Seriell über Kabel 23](#_Toc341536691)

[5.3.2. Can-Bus 24](#_Toc341536692)

[5.3.3. Bluetooth 25](#_Toc341536693)

[5.4. Auswertung und Darstellung 25](#_Toc341536694)

[5.4.1. Mikrocontroller mit LCD-Display 25](#_Toc341536695)

[5.4.2. Android 29](#_Toc341536696)

[6. Anforderungsdefinition, Systementwurf und Realisierung 30](#_Toc341536697)

[6.1. Systemübersicht und Modulbeschreibungen 30](#_Toc341536698)

[6.2. Datenpaket 31](#_Toc341536699)

[6.3. Messsystem 31](#_Toc341536700)

[6.3.1. Plattform 31](#_Toc341536701)

[6.3.2. Schaltpläne 34](#_Toc341536702)

[6.3.2.1. Gesamtschaltplan Messschaltung 34](#_Toc341536703)

[6.3.2.2. Spannungsregelung 35](#_Toc341536704)

[6.3.2.3. Drehzahlerfassung 36](#_Toc341536705)

[6.3.2.4. Auswertung Thermoelement (Abgastemperatur) 37](#_Toc341536706)

[6.3.2.5. Erfassung Tachosignal 37](#_Toc341536707)

[6.3.3. Detaillösungen Software Messsystem 38](#_Toc341536708)

[6.3.3.1. Schiebealgorithmus EGT/Max6675 38](#_Toc341536709)

[6.3.3.2. Paketserialisierung 39](#_Toc341536710)

[6.4. Einrichtung zur Auswertung und Visualisierung 39](#_Toc341536711)

[6.4.1. Klassendiagramme 39](#_Toc341536712)

[6.4.1.1. Paket de.gaffa.vescom.vesdroid 40](#_Toc341536713)

[6.4.1.2. Paket de.gaffa.vescom.vesdroid.bluetoothmanagement 41](#_Toc341536714)

[6.4.1.3. Paket de.gaffa.vescom.vesdroid.packet 42](#_Toc341536715)

[6.4.1.4. Android SDK: BluetoothChat 43](#_Toc341536716)

[6.5. Paketserialisierung/Deserialisierung 44](#_Toc341536717)

[6.5.1. Serialisierung in der Messeinheit 45](#_Toc341536718)

[6.5.2. Deserialisierung in der Visualisierungseinheit 46](#_Toc341536719)

[7. Zusammenfassung und Ausblick 48](#_Toc341536720)

[8. Abbildungsverzeichnis 49](#_Toc341536721)

[9. Quellenverzeichnis 50](#_Toc341536722)

[!!!!!!!!!!!!!!!!!!!!TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! 51](#_Toc341536723)

# Selbstständigkeitserklärung

Henning Groß, 749992, Studiengang: Technische Informatik BA, Fachbereich VI

**Erklärung zur Abschlussarbeit**

Ich versichere, dass ich meine Abschlussarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Berlin,

Unterschrift

# Copyright

Die vorliegende Arbeit steht unter Creative Commons-Lizenz (Attribution-Non-Commercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0)). Sie dürfen sie kopieren, veröffentlichen, übertragen und auch in Auszügen nutzen so lang der ursprüngliche Autor und die Arbeit im Ergebnis kenntlich gemacht und in keiner Weise der Eindruck erweckt wird, es handele sich um das Ergebnis Ihrer Arbeit. Sie dürfen den Inhalt der Arbeit nur für nichtkommerzielle Zwecke verwenden. Wenn Sie diese Arbeit – auch in Auszügen – nutzen, erweitern oder verändern so muss das Ergebnis unter der gleichen Lizenz veröffentlicht werden. In keinem Fall darf die weitere Nutzung erweitert oder eingeschränkt werden.

Der dieser Arbeit beigefügte Quelltext steht zum Teil unter General Public License (GPL, <http://www.gnu.org/copyleft/gpl.html>, deutsche Fassung: <http://www.gnu.de/documents/gpl-3.0.de.html>) und zum Teil unter Apache License (<http://www.apache.org/licenses/LICENSE-2.0>). Quellcode ist in den Dateien mit Lizenzhinweisen versehen. Solche Dateien die keine Lizenzhinweise enthalten stehen unter GPL. Nutzung der Quellen, auch in Auszügen, ist nur unter Beachtung der Lizenzbedingungen gestattet.

# Einleitung

Klassische Motorroller der Firma Piaggio (Vespa) erfreuen sich unter anderem aufgrund der möglichen Leistungsoptimierungen großer Beliebtheit. Wettbewerbsfahrzeuge des Typs V50N werden im Rennsport von 1,7PS auf bis zu über 40PS Motorleistung (Hinterrad) bei Drehzahlen von bis zu 15.000 Umdrehungen/Minute optimiert.

Die verwendeten Auspuffanlagen der Zweitaktmotoren führen durch Gegendruck in einem bestimmten Resonanzbereich (Drehzahl) zu erheblichem Leistungszuwachs. Außerhalb des Resonanzbandes liegt deutlich weniger Leistung an. Siehe Diagramm ab ca. 6250 Umdrehungen/Minute.

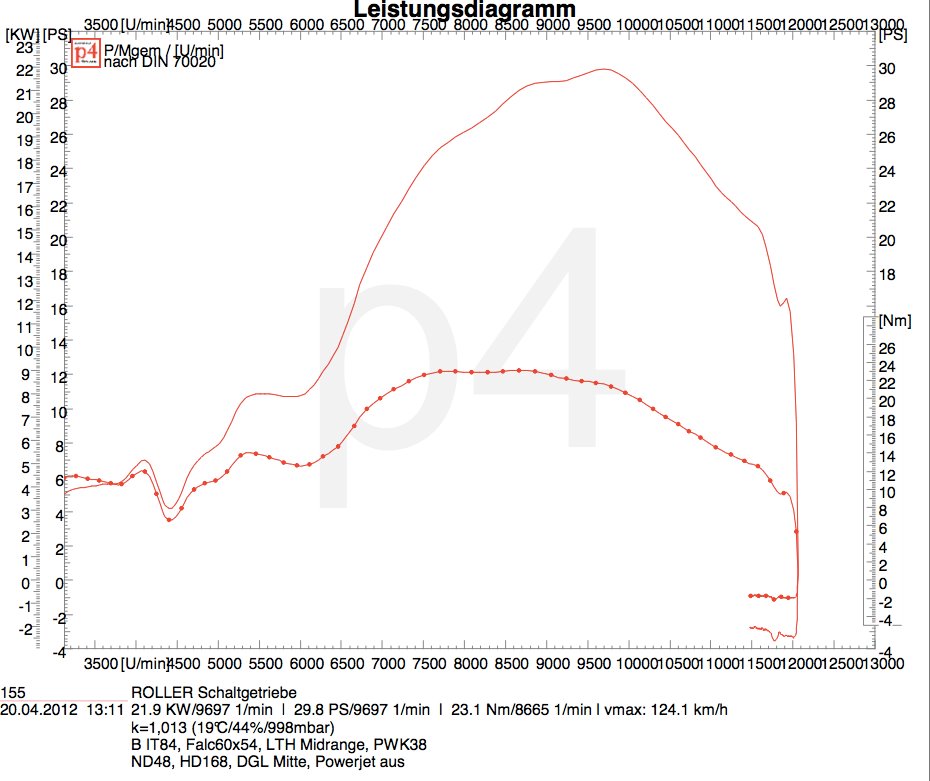


Abbildung ‑ - Leistungsdiagramm des Testfahrzeugs dieser Arbeit

Hier ist zu erkennen, dass vor Resonanzdrehzahl bei etwa 6250 Umdrehungen/Minute maximal 12 PS anliegen. Innerhalb eines Anstiegs der Drehzahl um ca. 750 Umdrehungen/Minute verdoppelt sich die Leistung auf 24 PS und steigt bis 29,8 PS.

Diese Charakteristik bedingt hohe Anforderungen an den Fahrer bei Schalt- und Startvorgängen. Diese können durch die Anpassung der Drehzahl effektiv optimiert werden. Die Optimierung besteht daraus, bei Startvorgängen den Motor beim Einkuppeln auf einer Drehzahl zu halten bei der der Motor ein möglichst hohes Drehmoment abgibt. Bei Schaltvorgängen muss unter Berücksichtigung der Getriebeübersetzung ein Zeitpunkt gewählt werden, bei dem im nächsten Gang eine Motordrehzahl erreicht wird zu der möglichst viel Leistung abgegeben wird sodass das Leistungsband optimal ausgenutzt wird.

Das Vornehmen solcher Anpassungen ist ohne Einrichtung zur Erfassung der Motordrehzahl schwierig.

Die Abgastemperatur lässt Rückschlüsse auf das Verbrennungsluftverhältnis zu. Niedrige Abgastemperatur deutet darauf hin, dass die Gemischeinstellung zu fett ist. Dies führt neben höherem Verbrauch zu direkten und indirekten Leistungseinbußen. So verrußt das Kolbendach und die Abgasanlage und werden Zündkerzen unbrauchbar. Eine hohe Temperatur deutet auf zu magere Vergasereinstellung, ungünstige Wahl des Zündzeitpunkte oder Undichtigkeiten des Systems (z.B. Zylinderkopf, Ansaugtrakt, Kurbelwellendichtring) hin und ist ein wichtiges Frühwarnzeichen bevor es zu thermischen Schäden wie dem Verklemmen des Kolbens oder Durchbrennen des Kolbendachs kommt. Die bei Viertaktmotoren übliche Ermittlung des Verbrennungsluftverhältnisses mit Hilfe von Lambda-Sonden ist beim Zweitaktmotor nicht möglich da die Sonden vom Öl im Abgasstrom mechanisch beschädigt werden. Somit ist die Überwachung der Abgastemperatur das effektivste Mittel zum Abstimmen und zur Überwachung von leistungsstarken Zweitaktmotoren.

Aufgrund des Alters der Fahrzeuge und der Anforderungen aufgrund des ursprünglichen Leistungsniveaus werden diese Fahrzeugdaten allerdings nicht erfasst und dargestellt. Bei Importmodellen fehlt häufig gar eine Einrichtung zur Erfassung der Geschwindigkeit. Auch existieren auf dem Markt keine Geräte welche die drei relevanten Informationen Drehzahl, Abgastemperatur und Geschwindigkeit übersichtlich und kombiniert darstellen. In Folge sind an Rennfahrzeugen häufig mehrere Messeinrichtungen untergebracht was die Übersicht des Fahrers einschränkt.

Die zentrale Darstellung überwachter Messwerte bringt den Fahrer in die Lage, sein Fahrzeug ideal einzustellen und zu betreiben. Die Entwicklung einer solchen Einrichtung ist Ziel dieser Arbeit.

Die Erfassung der Daten soll mit einem Mikrocontroller mit Hilfe eines im Verlauf der Arbeit zu entwickelndem Softwareprogramms (C) und entsprechenden Messschaltungen erfolgen. Zur Erfassung der Drehzahl, des Tachosignals und der Abgastemperatur sind geeignete Sensorschaltungen zu entwickeln deren Ausgangssignale vom Mikrocontroller erfasst und verarbeitet werden.

Die erfassten Daten müssen ausgewertet und dargestellt werden.

# Aufgabenstellung, fachliche Anforderungen/ Rahmen- und Umgebungsbedingungen

## Messeinrichtung

### Wahl einer Plattform zur Erfassung der Messdaten

Die Erfassung der Messdaten erfordert eine Echtzeitumgebung. Die erfassten und gewandelten physikalischen Größen sollen digitalisiert und übertragen werden können. Die Plattform muss über verschiedene digitale und analoge Anschlussmöglichkeiten (Analog-Digital-Wandler, Interrupt-Pins, Timer mit externem Trigger etc.) sowie Kommunikationsschnittstellen verfügen.

### Entwicklung der Messschaltungen, Erfassen der Signale/Messdaten

#### Drehzahl

Die Motordrehzahl muss über einen geeigneten Sensor erfasst und in ein zu der Messeinrichtung kompatibles Signal gewandelt werden. Die Verarbeitung des Drehzahlsignales in der Messeinrichtung ist zu gestalten.

##### Erwarteter Wertebereich

Die minimale Standgasdrehzahl des Motors (der Kurbelwelle) beträgt ca. 800 Umdrehungen/Minute. Drehzahlen bis 15000 Umdrehungen/Minute sind zu erwarten.

##### Aktualisierungsintervall

Die Drehzahlanzeige soll mindestens 2 Mal pro Sekunde aktualisiert werden.

#### Weg/Zeit (Tacho)

Der zurückgelegte Weg muss über einen geeigneten Sensor erfasst und in ein zu der Messeinrichtung kompatibles Signal gewandelt werden. Die Verarbeitung des Tachosignals in der Messeinrichtung ist zu gestalten.

##### Erwarteter Wertebereich

Die Drehrichtung der Räder soll zur Ermittlung der Geschwindigkeit keine Rolle spielen. Der Minimalwert entspricht daher 0 (Stillstand). Die maximal erfassbare Geschwindigkeit soll 200km/h betragen.

##### Aktualisierungsintervall

Der Reifenumfang des Testfahrzeugs beträgt 1330mm. Die Geschwindigkeitsanzeige ist bei hohen Geschwindigkeiten mindestens 2 Mal pro Sekunde zu aktualisieren.

#### Abgastemperatur

Die Abgastemperatur muss über einen bezüglich Temperaturbereich und Trägheit geeigneten Sensor erfasst und in ein zu der Messeinrichtung kompatibles Signal gewandelt werden. Die Verarbeitung des Signales der Abgastemperatur in der Messeinrichtung ist zu gestalten.

##### Erwarteter Wertebereich

Die Abgastemperatur bei laufendem Motor ist mit 200°-750° Celsius anzunehmen. Die Höchsttemperatur kann kurzfristig bis zu 900° Celsius betragen.

##### Aktualisierungsintervall

Die Abgastemperatur soll ein Mal pro Sekunde ermittelt und übertragen werden.

## Auswertung und Darstellung

### Wahl einer Plattform zur Auswertung und Darstellung der Messdaten

Zur Verarbeitung und Darstellung der Messdaten ist eine geeignete Plattform zu wählen. Anforderungen sind die Unterstützung eines zum Messsystem kompatiblen Übertragungsverfahrens sowie die  Möglichkeit, Messdaten übersichtlich zu visualisieren (Display, bevorzugt Grafisch).

### Entwicklung der Algorithmen zur Verarbeitung der Messdaten

Die Verarbeitung der von der Messeinrichtung übermittelten Werte der Drehzahl, des Tachosignals und der Abgastemperatur ist zu gestalten.

### Visualisierung der Messdaten

Die ermittelten Werte der Drehzahl, Geschwindigkeit und der Abgastemperatur sollen dargestellt werden.

## Realisierung der Kommunikation zwischen Mess- und Auswertungseinrichtung

### Wahl eines Übertragungsverfahrens

Es ist ein Übertragungsverfahren zu wählen welches von der Mess- und Auswerteeinrichtung unterstützt wird. Weiter sollte das Übertragungsprotokoll Fehlerkorrektur unterstützen. Es ist bevorzugt ein drahtloses Übertragungsverfahren zu wählen da die Verlegung von Leitungen aufgrund der beengten Verhältnisse am Fahrzeug problematisch ist. Die elektrische Trennung schützt die Einrichtung zur Visualisierung außerdem vor Schäden und verhindert die Einstreuung von Störungen.

### Entwicklung eines Datenformates und Protokolls

Für die Übertragung der Messdaten ist ein Nachrichtenformat zu spezifizieren. Start-, End- und Trennzeichen sind in Form eines einfachen Protokolls zu spezifizieren.

# Lösungsansätze

## Messeinrichtung

### Drehzahl

#### Hall-Effekt

Der Hall-Effekt beschreibt eine Spannung die in einem stromdurchflossenen Leiter innerhalb eines Magnetfeldes auftritt. Dabei kommt es aufgrund der Kraft die das Magnetfeld auf die bewegte elektrische Ladung auswirkt (Lorentzkraft) zu einer Ladungstrennung welche eine messbare Spannung erzeugt (Hall-Spannung) [1]. Dies wird in der folgenden Grafik verdeutlicht.

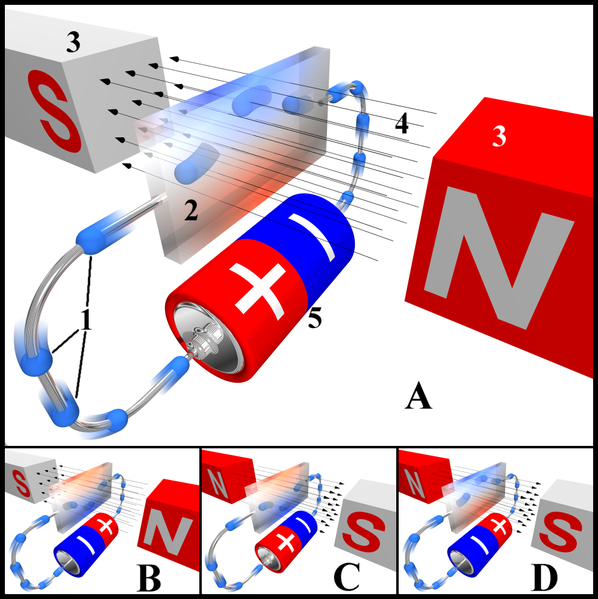


Abbildung ‑ - Hall-Effekt, Quelle: Wikipedia [1]

Es gibt auf dem Markt einige integrierte Schaltkreise welche den Hall-Effekt nutzen. Hier wird die gemessene Spannung verstärkt. Ein Schmitt-Trigger sorgt für die binäre Ansteuerung eines Transistors der für einen TTL-Pegel am Ausgang sorgt.

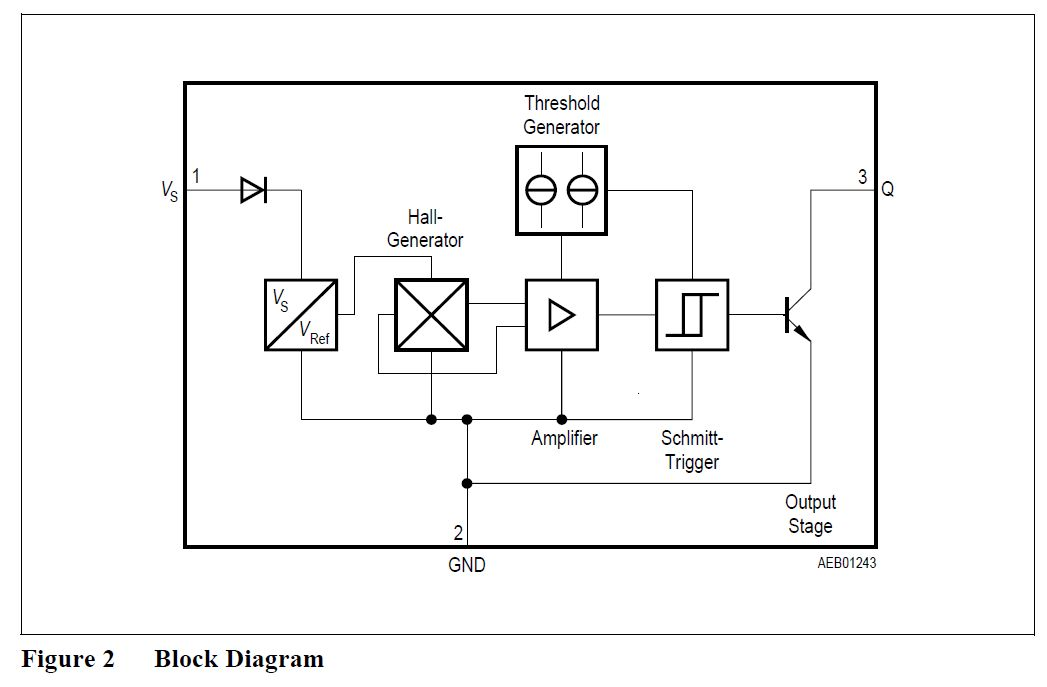


Abbildung ‑ - Blockdiagramm Hall-IC, Quelle: Infineon Datenblatt TLE49x5L [2]

Je nach Ausführung (Unipolar/Bipolar, Switch/Latch) finden sich leichte Abweichungen in der Ausführung. Durch die Art der Realisierung der Vespa-Lichtmaschine, bei der Spulen auf dem Stator durch Dauermagnete auf dem Rotor zur Induktion von Spannung genutzt werden, ist es möglich, einen solchen IC auf dem Stator zu befestigen um die Drehzahl zu erfassen.

Von Vorteil ist, dass die ICs ein sauberes binäres Signal mit TTL-Pegel liefern und keine weitere (analoge) Verarbeitung des Signals vor Einspeisung in den Mikrocontroller nötig ist.

Nachteilig ist die Modifikation am Stator. Der Rotor muss entfernt werden, die zuverlässige Befestigung der Schaltung am Stator ist wichtig. Eine zusätzliche Leitung muss gelegt werden.



Abbildung ‑ - Hall-IC-Schaltung auf Stator

#### Induktion am Zündkabel

Die Vespa-Zündung ist eine Kondensatorentladungszündung. Hier wird durch die Zündspule auf dem Stator ein Kondensator geladen und zum Zündzeitpunkt mit Hilfe eines Thyristors entladen. Hierbei entstehen Spannungen von bis zu 40kVolt an der Zündkerze.

Einige Wicklungen Kupferkabel um das Zündkabel führen zu messbarer Induktionsspannung. Da jedoch viele Parameter wie die tatsächliche Zündenergie und das Umfeld sowie der Kerzenwiderstand stark variieren ist die Anzahl der nötigen Wicklungen nicht einfach zu ermitteln und der Aufwand zur Aufarbeitung des Signals erheblich. Systeme wie Leistungsprüfstände (z.B. P4 der Firma Amerschläger) oder Stroboskoplampen zum Einstellen von Zündzeitpunkten arbeiten häufig mit Abnahme über das Zündkabel. Es ist folglich ein gangbarer wenn auch komplexer Ansatz.



Abbildung ‑ - Induktionsmessung am Zündkabel

#### Lichtmaschine

Die Vespa-Lichtmaschine liefert eine Wechselspannung von überlagerten Wellen (mehrere Erzeugerspulen). Die Frequenz ist proportional abhängig zur Drehzahl da der Stator direkt auf der Kurbelwelle befestigt ist. Üblich sind zwei Spulen und vier Magnete mit alternierender Polarität sodass pro Umdrehung jede Spule zwei Perioden durchläuft.

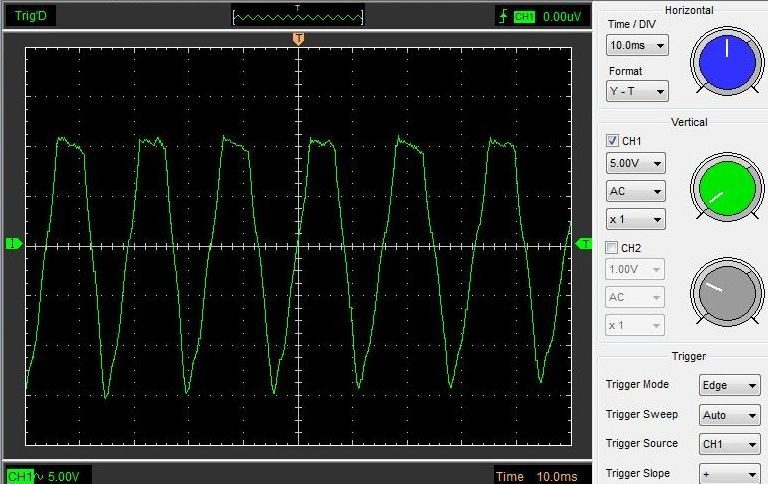


Abbildung ‑ - Lichtmaschinensignal Vespa

Durch Beschaltung mit einer Diode kann das Signal auf die positiven Halbwellen begrenzt werden.



Abbildung ‑ - Lichtmaschine mit Diode

Ein Tiefpassfilter glättet das Signal zusätzlich. Die Spannung wird über einen Spannungsteiler reduziert, eine Z-Diode schützt vor Überspannungen.

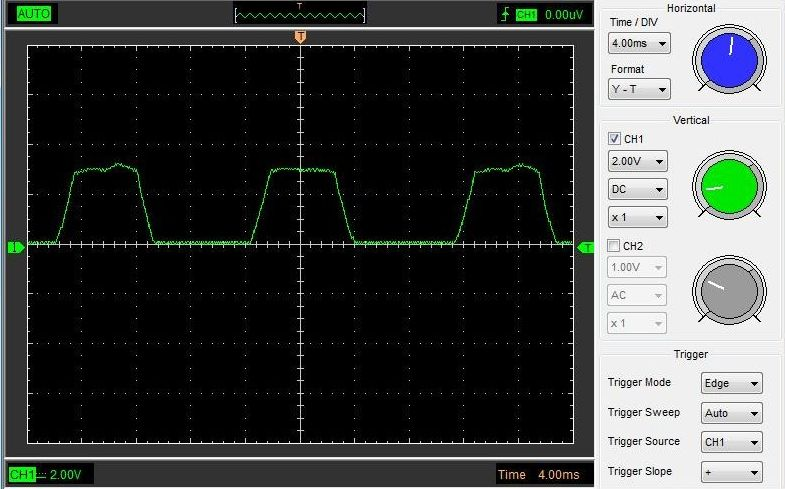


Abbildung ‑ - Lichtmaschinensignal mit Diode, Filter und Spannungsbegrenzung

In dieser Form kann das Signal direkt vom Mikrocontroller verarbeitet werden, wenn dieser (wie die verschiedenen Ausführungen der Atmel Atmega-Produktreihe) über Schmitt-Trigger an den digitalen Eingängen verfügt.

### Tachosignal

#### Reedkontakt

Eine gängige Variante der Ermittlung der aktuellen Geschwindigkeit bzw. der zurückgelegten Distanz ist das Anbringen eines Reedkontakts (Reed-Relais) an der Gabel und eines Magnetes an der Felge. Beim Reed-Relais befindet sich ein Federkontakt innerhalb eines gasgefüllten Kolbens (Korrosionsschutz). Der Kontakt schließt wenn er in ein Magnetfeld gebracht wird. Durch den Dauermagneten an der Felge wird er ausgelöst. Problematisch hierbei ist, dass Reed-Kontakte eine relative hohe Prellneigung haben und das Signal so noch entprellt werden muss. Ein weiterer Nachteil ist, dass das Reed-Relais ein mechanisches Bauteil ist und somit Verschleiß unterliegt.

#### Rechnerisch

Die Motordrehzahl lässt sich über die Getriebeübersetzung und den Radumfang in die Geschwindigkeit umrechnen. Umgekehrt erfassen Leistungsprüfstände wie der P4 von Amerschläger so die Motordrehzahl über die Winkelveränderung der Prüfstandsrolle nachdem die Übersetzung über eine Drehzahlmessung eingemessen wurde. Rutschende oder gezogene Kupplungen verfälschen das Ergebnis jedoch erheblich. Zusätzlich weichen die Getriebeübersetzungen verschiedener Roller sehr stark voneinander ab. Auch der Reifendruck und die Beladung des Fahrzeuges haben bei dieser Methode Einfluss auf das Messergebnis.

#### Hall-Effekt

Der als Ansatz zu Erfassung der Drehzahl beschriebene Hall-Effekt kann auch für das Tachosignal genutzt werden. Wie im Ansatz mit Reedkontakt beschrieben werden hierbei Umdrehungen des Rades gemessen. Die Hall-IC-Schaltung wird wird ebenfalls an der Gabel angebracht, ein Dauermagnet am Reifen. Als einzig notwendiger Parameter muss der Radumfang bekannt sein. Gegenüber dem Reed-Relais weisen Hall-ICs weder Prellverhalten noch mechanischen Verschleiß auf. Das Ausgangssignal liegt in einer vom Mikrocontroller direkt verarbeitbaren Form vor (TTL (5V) bzw. Low Voltage TTL (3V3) je nach Ausführung).



Abbildung ‑ - Tachosensor-Prototyp befestigt mit Kabelbinder. Dauermagnet an Felge

### Abgastemperatur

Aufgrund des Temperaturbereichs bietet sich als Sensor ein Typ-K Thermoelement an. Themoelemente wandeln Wärme in elektrische Energie (vgl. Wikipedia [3]). Hierbei wird der Seebeck-Effekt genutzt welcher die auftretende Spannung in einem Stromkreis zwischen zwei verschiedenen elektrischen Leitern bei einer Temperaturdifferenz der Kontaktstellen beschreibt. Die Spannung kann in Abhängigkeit der Materialkonstanten in eine Temperaturdifferenz zwischen Messstelle und Vergleichsstelle gewandelt werden. Da es sich nur um einige Mikrovolt handelt werden hohe Anforderungen an die Genauigkeit der Messeinrichtung gestellt.

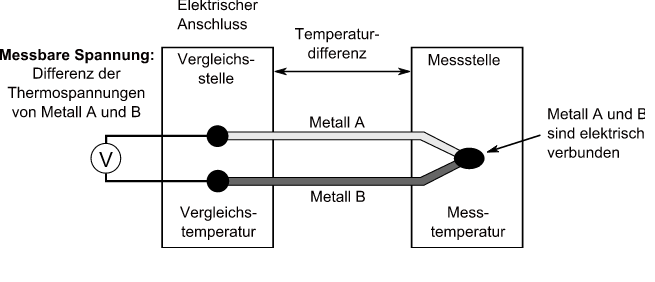


Abbildung ‑ - Thermoelement, Quelle: Wikipedia [3]

#### Umsetzung mit Analogschaltung

Die induzierte Spannung muss verstärkt werden bevor sie sinnvoll gewandelt werden kann. Problematisch ist die je nach Unterbringung erhebliche Leitungslänge zwischen Messschaltung und Verstärkerschaltung/A-D-Wandler. Hier müsste im Zweifel eine Verstärkung im Umfeld der Messstelle umgesetzt werden während die übrige Schaltung im Umfeld des Mikrocontrollers unterzubringen ist. Weiter hat eine weitere Temperaturmessung der Umgebungstemperatur als Vergleichsmessung zu erfolgen.

#### IC

Am Markt gibt es integrierte Schaltkreise zur Auswertung von Thermoelementen. Einige realisieren ausschließlich die Verstärkung sodass eine durch Mikrocontroller sinnvoll messbare Spannung entsteht, andere messen und verrechnen bereits die Vergleichstemperatur. Der MAX6675 von Maxim liefert den Messwert seriell als Zahlenwert mit der Basis 2 (binär). Er kümmert sich also um Verstärkung, Referenz und Digitalisierung.

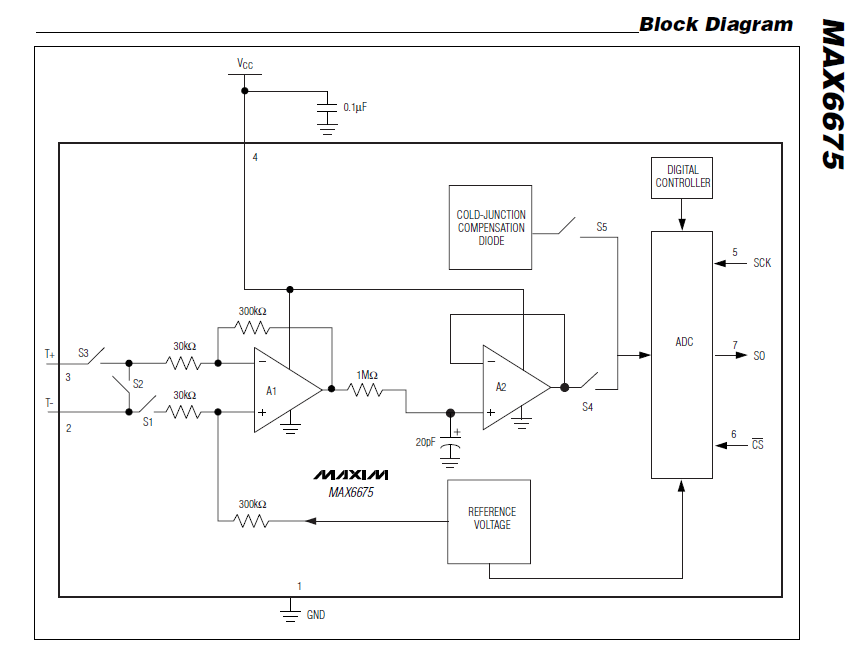


Abbildung ‑ - Blockdiagramm MAX6675, Quelle: Maxim-Datenblatt MAX6675 [4]

Die Ausgabe erfolgt über eine serielle Verbindung.

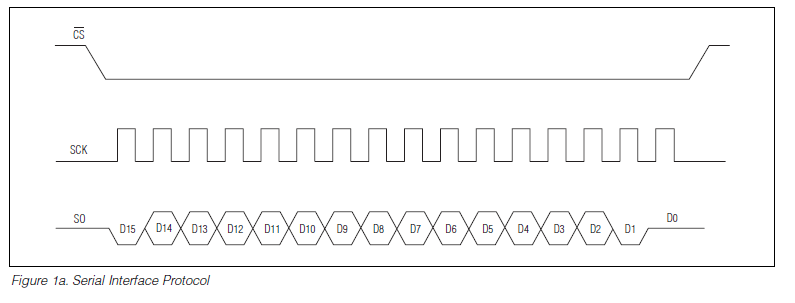


Abbildung ‑ - Protokoll MAX6675, Quelle: Maxim-Datenblatt MAX6675 [4]

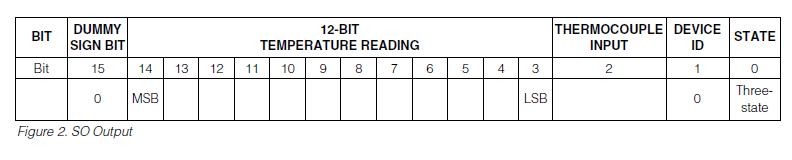


Abbildung ‑ - Datenframe/Bits MAX6675, Quelle: Maxim-Datenblatt MAX6675 [4]

## Signalverarbeitung

### Drehzahl / Tacho

Mikrocontroller der Atmega-Familie haben Eingänge für externe Interrupts. So kann wahlweise die positive oder negative (oder beides) Flanke zur Auslösung eines Interrupts gewählt werden. Ein integrierter Schmitt-Trigger sorgt für eine saubere Schaltschwelle. So kann, wird gleichzeitig ein Timer zur Erfassung einer „Systemzeit“ genutzt, bei jeder erfassten Flanke die Zeitdifferenz (in Takten) zur letzten Messung erfasst werden. Außerhalb der Interruptroutine kann diese dann weiter verarbeitet (an die Auswerteeinheit übermittelt) werden. Innerhalb der Auswerteeinheit muss der zum Systemtakt relative Wert anschließend in die entsprechende Einheit gewandelt werden.

#### Beispiele

CLK = Systemtakt = 4\*10^6 Hz  
CLKDIV = Prescaler des Timers der Systemzeit = 256  
TPS = Ticks pro Sekunde des Systemzeit-Timers = CLK / CLKDIV = 15625 Hz  
T\_DIFF = Gemessene Takte zwischen zwei Flanken der Lichtmaschine bzw. zwischen zwei Flanken des Tachosensors  
SPR = Signale pro Umdrehung (zB. Zündung: zwei Signale aufgrund mehrerer Lichtspulen, Tacho: ein Signal aufgrund eines Magneten am Rad)  
U = Umfang des Rades mit dem Tachosensor in cm

##### Drehzahl in Umdrehungen pro Minute

RPM = TPS/T\_DIFF/SPR\*60

##### Geschwindigkeit in Kilometern pro Stunde

V = TPS/T\_DIFF/SPR\*3600\*U/10^5

Der Initialzustand bei Einschalten des Systems ist bei der Implementierung zu berücksichtigen und zu behandeln (ansonsten je nach Realisierung der Berechnung Auftreten von Division durch 0 möglich).

### Abgastemperatur

Abbildung 5.12 zeigt den Aufbau eines Datenpaketes, Abbildung 5.11 die Kommunikation und Bitfolge. Für den Einsatzzweck reicht eine Auflösung von ~10 Grad Celsius. Zur genaueren Darstellung ist eine Abstufung bis auf 1 Grad Celsius wünschenswert. Die 12 Bit der Übertragung jedoch lösen mit 0,25 Grad Celsius auf. Die für unsere Temperaturauswertung relevanten Bits werden daher auf die zehn Bits 5-14 definiert. Die Abwicklung der Kommunikation ist wie folgt zu realisieren:



Abbildung ‑ - Ablaufdiagramm Kommunikation mit Max6675

Zunächst wird CS low gesetzt und der IC dadurch vom Mess- in den Kommunikationsmodus versetzt. Als nächstes wird durch das Setzen von SCK auf High das erste (irrelevante) Bit abgefragt. SCK geht wieder Low. Nun kommen die 10 relevanten Bits in der Reihenfolge MSB->LSB. Diese werden jeweils durch Setzen von SCK auf High, Auslesen und Einshiften des Bits auf der Leitung S0 und setzen von SCK auf Low in das Wertregister des Mikrocontrollers addiert. Zum Schluss wird CS wieder High gesetzt und der IC somit wieder in den Mess-Modus versetzt.   
Das Diagramm verzichtet zur besseren Übersicht auf die Initialisierung der Wertvariablen sowie auf Initialisierung und Manipulation der Durchlaufvariablen.

Bei der Implementierung ist darauf zu achten, dass der Max6675 keine so schnellen Schaltzeiten/Taktungen unterstützt wie Atmega-Mikrocontroller. Nach Flankenwechseln und zwischen einzelnen Messungen sind ggfs. Delays einzubauen.

## Datenübertragung

Die Datenübertragung zwischen der Mikrocontroller-Messeinrichtung und dem System zur Auswertung und Visualisierung ist zu realisieren.

### Seriell über Kabel

Da sowohl die Mikrocontroller der Atmega-Familie als auch das in der Evaluation zur Visualisierung getestete Android-Smartphone über integrierte UART-Schnittstellen verfügen ist die serielle Kommunikation über Kabel naheliegend. Leider sind die für die Kommunikation mit dem HTC-Smartphone benötigten Stecker teuer (da proprietär), kaum verfügbar und schwer zu löten. Weiter wird ein „gerootetes“ Gerät (Gerät mit administrativem Zugriff) benötigt, da der Zugriff auf die serielle Schnittstelle nicht vorgesehen ist. Zu guter Letzt verfügt nicht jedes Android-Gerät überhaupt über eine serielle UART-Schnittstelle (oder diese ist nur schwer im Gehäuseinneren zugänglich). Dennoch wurde dieser Ansatz als erster evaluiert. In Abbildung 5.14 ist die erfolgreiche Kommunikation zu sehen.

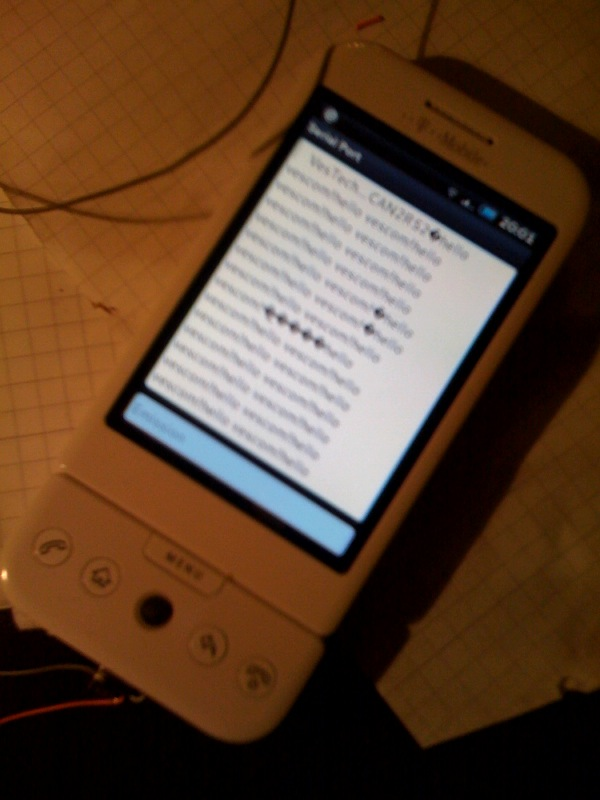


Abbildung ‑ -Mikrocontroller kommuniziert über serielle Kabelverbindung mit Android-Mobiltelefon

Ein weiteres Problem im ersten Versuch waren die ebenfalls in der Abbildung erkennbaren Störungen die über die Leitung ins System kamen.

#### Can-Bus

Die Störanfälligkeit und andere Nachteile der seriellen Kommunikation haben zu der Evaluation der Übertragung der Daten via Can-Bus geführt. Diese Kommunikationsmöglichkeit wurde als gangbar vor allem zwischen zwei Mikroncontrollern mit einigem Abstand (zum Beispiel Messsystem im Motorbereich und Visualisierungseinrichtung im Lenkerbereich) befunden. Wenn jedoch zum Beispiel Android-Geräte zur Visualisierung eingesetzt werden sollen bedarf es zusätzlich einer weiteren Übertragungseinrichtung da diese über keine Can-Schnittstelle verfügen. Des Weiteren ist der Schaltungsaufwand nicht unerheblich.

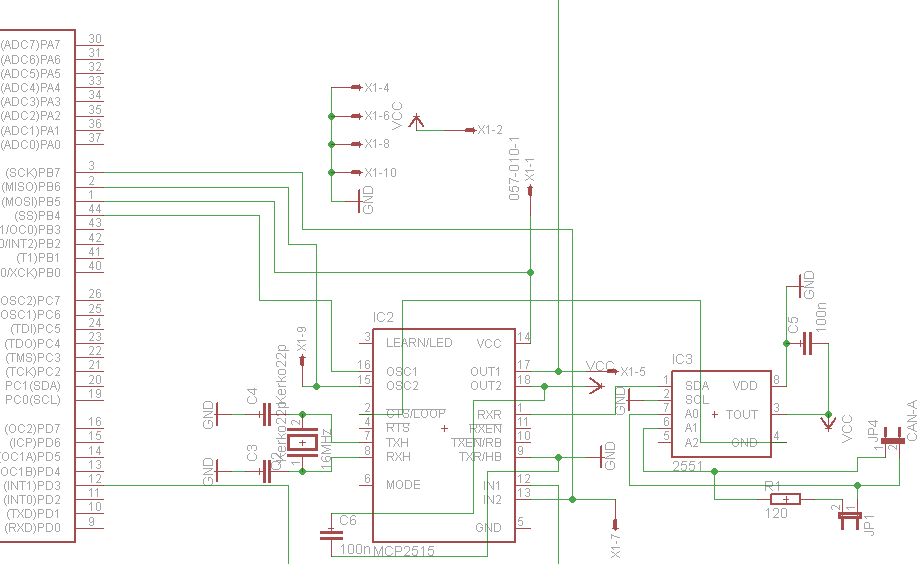


Abbildung ‑ - Auszug des Schaltplans für Kommunikation via Can-Bus

### Bluetooth

Alle gängigen Android-Geräte verfügen über Bluetooth-Hardware und die Nutzung als serielle Schnittstelle ist ohne Erlangung von Administrationsprivilegien möglich. Weiter gibt es fertige ICs welche über eine UART-Schnittstelle empfangene Daten per Bluetooth versenden. Durch die Einfachheit der Benutzung, die eingebaute Fehlerkorrektur, die entkoppelte Kommunikation (und hierdurch nicht vorhandene Gefahr der physikalischen Schädigung der Android-Geräte durch die Messschaltung) sowie die geringen Kosten stellt Bluetooth eine ideale Übertragungsmöglichkeit dar.

## Auswertung und Darstellung

### Mikrocontroller mit LCD-Display

Ein erster Prototyp verfolgte den Ansatz, ein LC-Display im Lenker unterzubringen und den Rest der Schaltung nahe des Motors.



Abbildung ‑ - Display eines Prototypen am Vespa-Lenker

Leider wurden hierbei Störungen durch die langen Datenleitungen eingestreut. Diese konnten auch durch Entstörwiderstände nicht vollständig eliminiert werden.



Abbildung ‑ - Display mit Störungen

Ein späterer Prototyp enthält einen Mikrocontroller, die CAN-Beschaltung, welche im vorangegangenen Kapitel erläutert wurde, sowie zwei Taster und das Display. Schaltungen für Drehzahl- und Tachoauswertung sind direkt integriert. Die Messschaltung für die Abgastemperatur sollte per CAN-Bus angekoppelt werden.

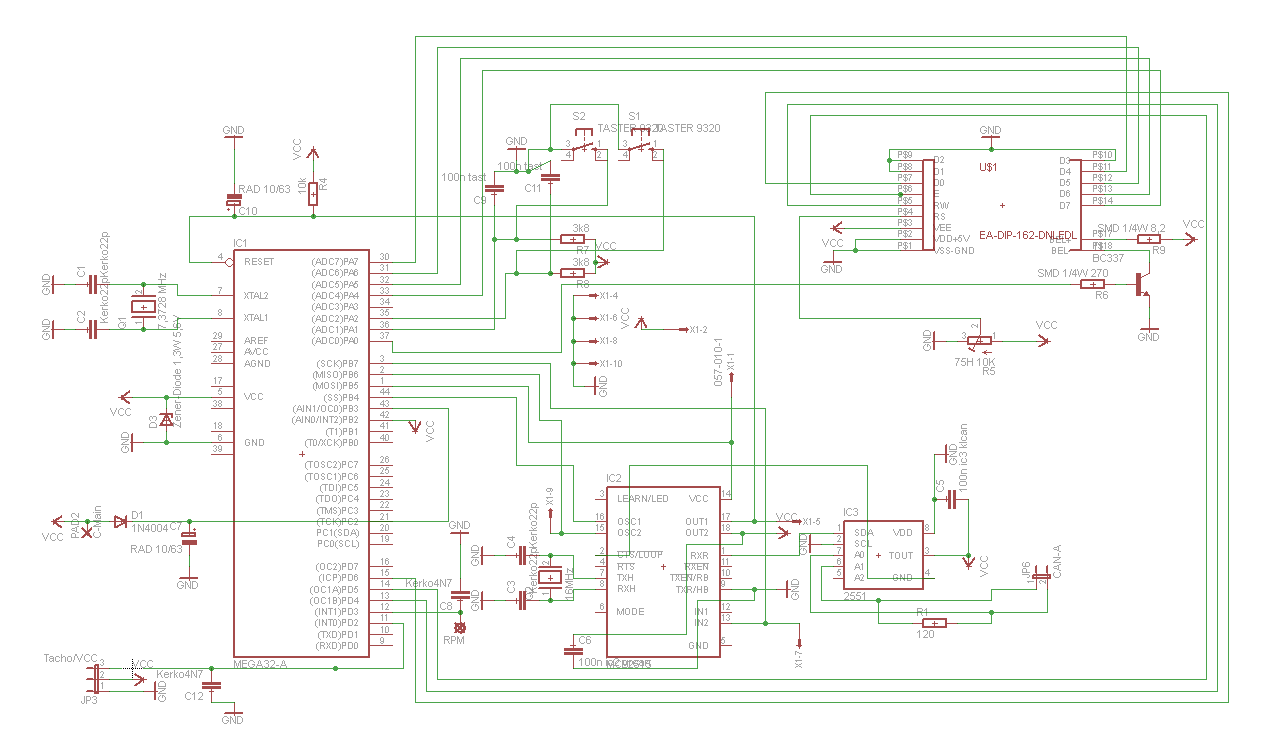


Abbildung ‑ - Schaltplan Prototyp integrierte Mess-, Auswerte- und Visualisierungseinheit

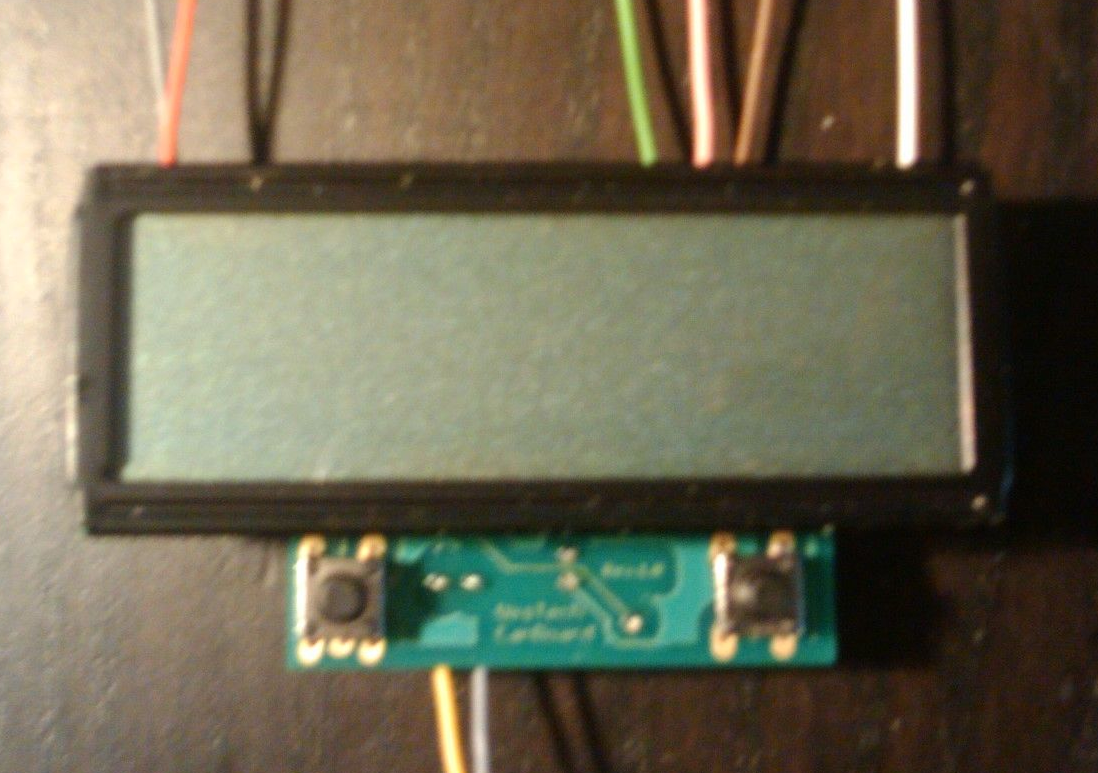


Abbildung ‑ - Prototyp integrierte Mess-, Auswerte- und Visualisierungseinheit

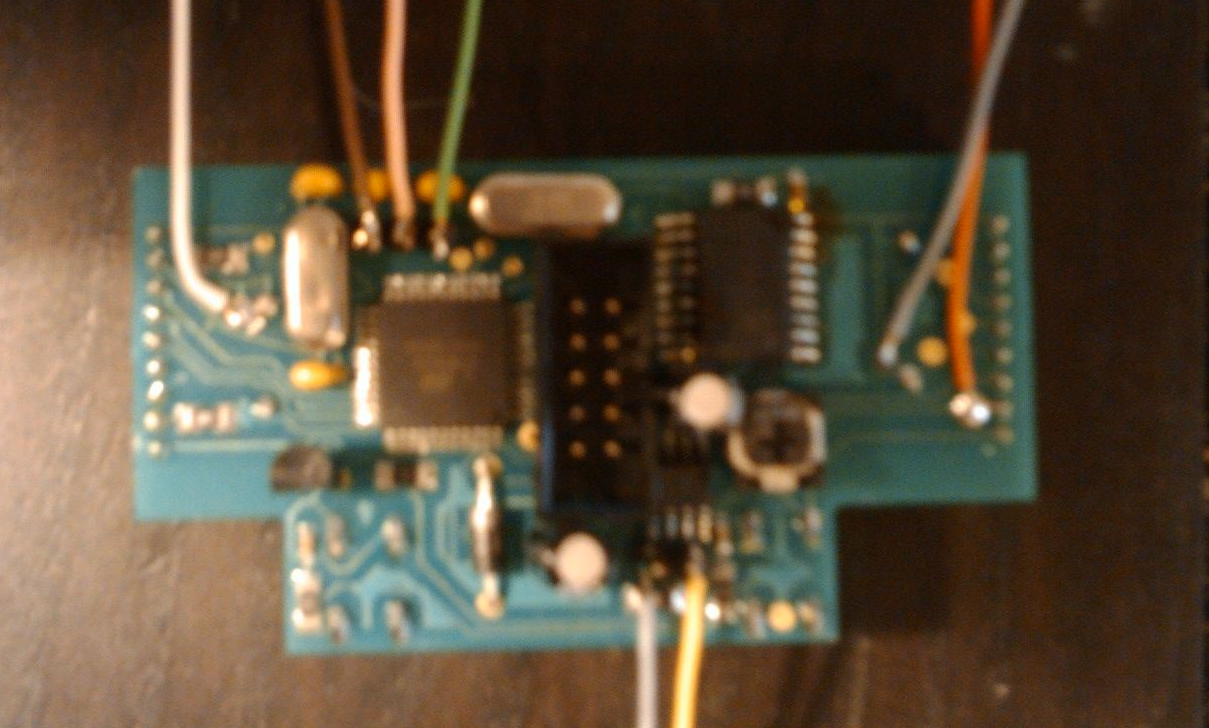


Abbildung ‑ - Prototyp integrierte Mess-, Auswerte- und Visualisierungseinheit

Dieser Weg erschien zunächst als Königslösung. Aufgrund von Platzproblemen im Vespa-Lenker und existierender beweglicher Teile wie der Schalt- und Gasrolle sowie des bei manchen Modellen vorhandenen Zündschlosses im Lenker ist die Unterbringung aber schwierig und es kam immer wieder zu Beschädigungen an den Prototypen. Auch drang regelmäßig Flüssigkeit in das System ein (die Taster mussten nutzbar bleiben um die Ansichten zu wechseln).

### Android

Der letzte evaluierte Lösungsweg ist die Nutzung eines Android-Gerätes. Der große Vorteil hierbei ist, dass die Auswertung und Visualisierung innerhalb eines viel leistungsfähigeren Umfeldes erfolgen kann. Bei Berechnungen ist nicht auf überlaufende Register zu achten, Visualisierung kann viel komfortabler als auf einem LC-Display erfolgen. Auch kann so der Fokus stärker auf die Entwicklung der Messschaltungen und Messalgorithmen gesetzt werden da der Aufwand im Bereich Auswertung/Darstellung sinkt.

# Anforderungsdefinition, Systementwurf und Realisierung

## Systemübersicht und Modulbeschreibungen

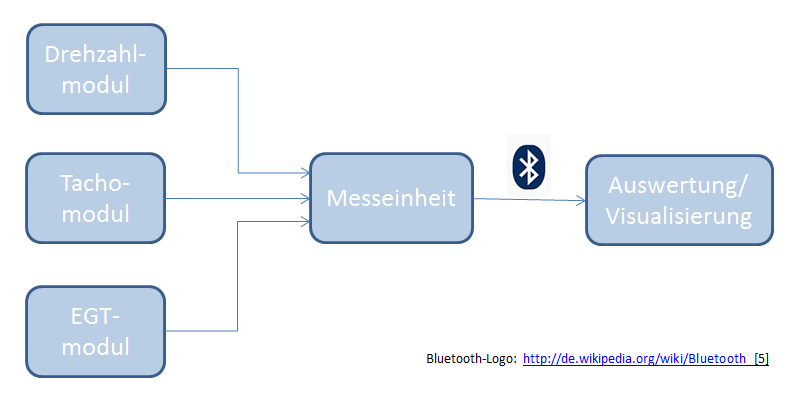


Abbildung ‑ - Systemübersicht

|  |  |  |  |
| --- | --- | --- | --- |
| **Modul** | **Zweck** | **Eingänge** | **Ausgänge** |
| Drehzahlmodul | Wandlung des Lichtmaschinensignals in ein zu der Messeinheit kompatibles Signal (annähernd Rechteck, direkt proportional zur Motordrehzahl) | - | Motorumdrehungen |
| Tachomodul | Erfassung der Radumdrehungen durch Hall-Effekt und Übertragung als zur Messeinheit kompatiblem Signals (Rechteck, entsprechend der Radumdrehungen) | - | Radumdrehungen |
| EGT-Modul | Wandlung des Ausgangssignals eines Typ-K Thermoelements (Spannung) in ein zur Messeinheit kompatiblen Signals (binär, serielle Übertragung) | - | Abgastemperatur |
| Messeinheit | Erfassen der Frequenzen/Zeitdeltas von Drehzahl- und Tachomodul sowie Verarbeitung der Daten vom EGT-Modul. Übertragung via Bluetooth | Motorumdrehungen Radumdrehungen Abgastemperatur | Datenpakete für Auswertung/ Visualisierung |
| Auswertung/ Visualisierung | Empfang der Datenpakete der Messeinheit via Bluetooth. Auswertung und Visualisierung der Messdaten. | Datenpakete der Messeinheit | - |

## Datenpaket

Das Datenpaket welches zwischen Mess- und Auswerte-/Visualisierungseinheit verschickt wird enthält folgende Daten

|  |  |
| --- | --- |
| **Feld** | **Beschreibung** |
| PACKET\_ID | Fortlaufende Paketnummer |
| RPM\_DIFF | Takte zwischen den letzten zwei Flanken am Lichtmaschinensensor |
| V\_DIFF | Takte zwischen den letzten zwei Flanken am Radumdrehungssensor |
| EGT | Abgastemperatur |

## Messsystem

Das Drehzahlmodul wird über den unter Lösungsansätze/Drehzahl/Lichtmaschine beschriebenen Ansatz realisiert.

Das Tachomodul wird über den unter Lösungsansätze/Tachosignal/Hall-Effekt beschriebenen Ansatz realisiert.

Die Abgastemperatur wird über den unter Lösungsansätze/Abgastemperatur/IC beschriebenen Ansatz mit Hilfe des Maxim ICs Max6675 realisiert.

Angeschlossen werden die Messschaltungen an einen Mikrocontroller der Atmega-Familie der über den USART mit einem Bluetooth-Modul kommuniziert.

### Plattform

Das Entwicklungsboard Bluecontroller ([www.bluecontroller.com](http://www.bluecontroller.com)) wird als Basis genutzt. Es enthält das BTM-222 Bluetooth-Modul, einen 3,3V Spannungsregler für das Bluetooth-Modul (welches auf 3,3 statt 5V arbeitet), einen Atmega-Mikrocontroller, den nötigen Levelshifter zwischen Bluetooth-Modul und Mikrocontroller, Status-LEDs, zwei Taster und eine Chipantenne. Bei sehr kleiner Bauform und günstigem Preis reduziert das Board somit Entwicklungsarbeit und Kosten bei der Umsetzung.

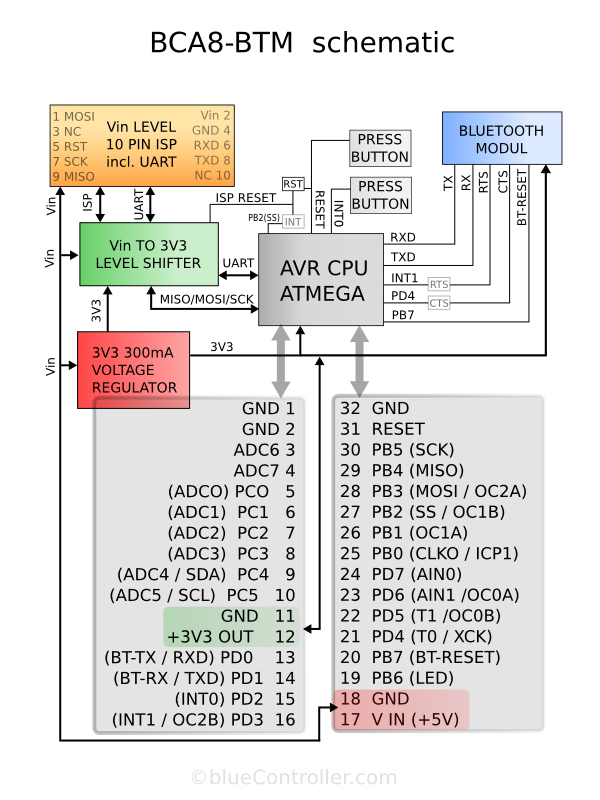


Abbildung ‑ -Blockschaltbild Bluecontroller, Quelle: www.bluecontroller.com

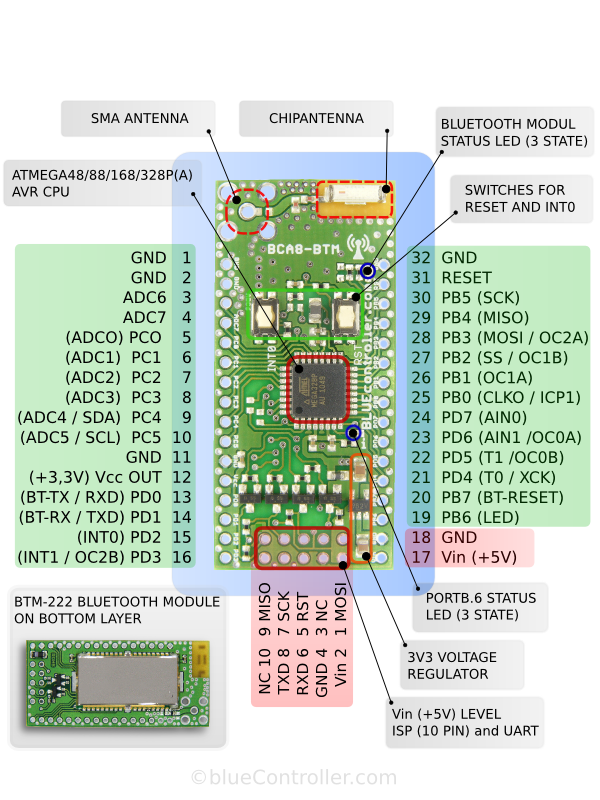


Abbildung ‑ - Bluecontroller, Bordlayout mit Legende, Quelle: www.bluecontroller.com

### Schaltpläne

#### Gesamtschaltplan Messschaltung

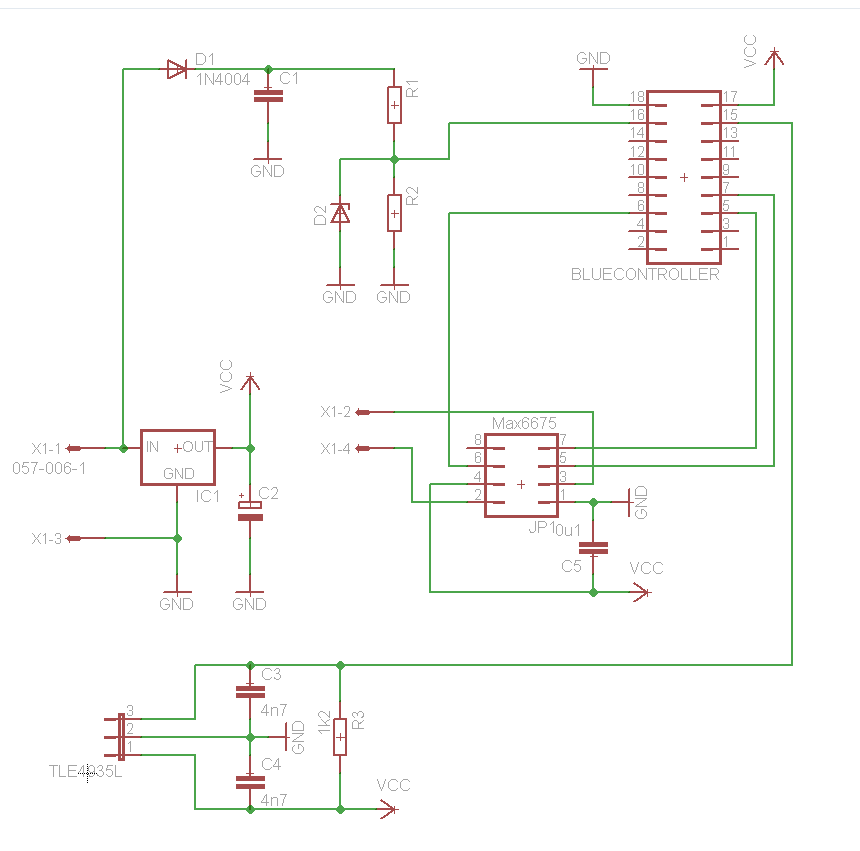


Abbildung ‑ - Gesamtschaltplan Messsystem

Der Gesamtschaltplan zeigt links mittig um IC1 die Schaltung zur Regelung der Eingangsspannung, davon ausgehend neben VCC ein Abgriff zur Schaltung zur Drehzahlerfassung um C1. Unten links um TLE4935L die Beschaltung des Hall-ICs zur Erfassung des Tachosignals. Weiter ist der Max6675 zur Erfassung der Abgastemperatur sowie der Bluecontroller beschaltet.

#### Spannungsregelung

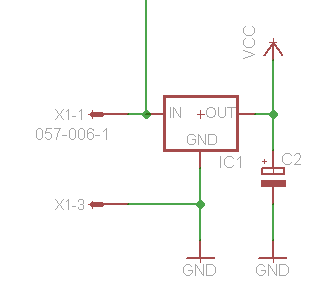


Abbildung ‑ - Schaltplan Spannungsregelung Messsystem

Die Schaltung zur Regelung der Spannung enthält einen IC zur Regelung der Bordspannung (~12V AC) auf 5VDC. Wird ein Festspannungsregler statt einem kombinierten Regler verwendet ist die Gleichrichtung durch Brückengleichrichter oder Diode zu realisieren. Weiter im Bild ist ein Kondensator zur Stabilisierung der Ausgangsspannung.

#### Drehzahlerfassung

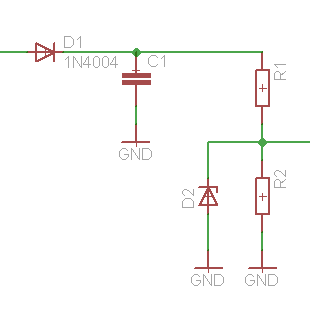


Abbildung ‑ - Schaltplan Drehzahlerfassung Messsystem

Die Messschaltung zur Erfassung der Drehzahl besteht aus einer Diode D1 zur Beschränkung auf die positive Halbwelle, einem Kondensator C1 zur Glättung und in Folge einem Spannungsteiler zur Begrenzung der Amplitude und einer Zenerdiode D2 zum Schutz des Eingangs am Mikrocontroller. Die Dimensionierung des Spannungsteilers muss derart erfolgen, dass bei schwankender Bordspannung die Schaltschwelle der Schmitt-Trigger im Mikrocontroller nicht unterschritten, die maximale Eingangsspannung jedoch nicht überschritten wird. Weiter muss der Widerstand R1 derart ausgelegt werden dass der Strom durch D2 bei Spannungsspitzen/hoher Bordspannung D2 nicht schädigt.

#### Auswertung Thermoelement (Abgastemperatur)

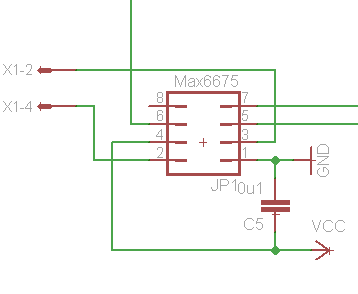


Abbildung ‑ - Schaltplan Beschaltung Max6675

Die Beschaltung des Maxim-ICs Max6675 zur Auswertung von Typ K Thermoelementen gemäß Datenblatt.

#### Erfassung Tachosignal

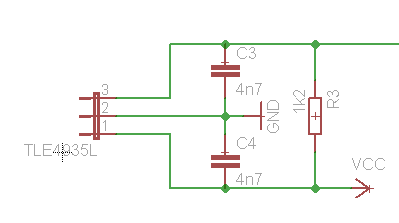


Abbildung ‑ - Schaltplan Beschaltung TLE49x5

Die Beschaltung des Hall-ICs TLE4935L von Infineon zur Erfassung des Tachosignals gemäß Datenblatt.

### Detaillösungen Software Messsystem

#### Schiebealgorithmus EGT/Max6675

Der hier als Codeauszug aufgeführte Algorithmus implementiert den unter Lösungsansätze/Signalverarbeitung/Abgastemperatur modellierten Algorithmus.

/\*\*

\* communciates with max6675 to determine egt

\* all delay-calls are because the max6675 does not support timings as fast as the mega

\*/

uint16\_t get\_egt(){

uint16\_t temp=0;

//cs low for data

EGT\_PORT &= ~(1 << EGT\_CS);

\_delay\_ms(EGT\_DELAY);

//clk high for irrelevant bit

EGT\_PORT |= (1 << EGT\_CLK);

\_delay\_ms(EGT\_DELAY);

//clk low

EGT\_PORT &= ~(1 << EGT\_CLK);

\_delay\_ms(EGT\_DELAY);

//now go through relevant bits and shift them in

for(int i=0; i<10; i++){

//clk high for next bit

EGT\_PORT |= (1 << EGT\_CLK);

\_delay\_ms(EGT\_DELAY);

//shift in bit

temp = temp << (EGT\_PIN & (1 << EGT\_DATA));

//clk low

EGT\_PORT &= ~(1 << EGT\_CLK);

\_delay\_ms(EGT\_DELAY);

}

return temp;

}

#### Paketserialisierung

Die Serialisierung und Deserialisierung von Datenpaketen von Messsystem und Visualisierungseinrichtung ist übersichtlich im Abschnitt „Paketserialisierung/Deserialisierung“ zusammengefasst.

## Einrichtung zur Auswertung und Visualisierung

### Klassendiagramme

Im folgendem wird der softwaretechnische Systementwurf für die Erstellung der Android-Software zum Empfang, der Auswertung und Visualisierung der Messdaten mit Hilfe von Klassendiagrammen beschrieben. Vertiefend wird das Studium der Programmquellen aus der Anlage oder dem beigefügtem Datenträger empfohlen. Begonnen wird mit einer Übersicht. In Folge werden die einzelnen Pakete beschrieben.

Zunächst das Klassendiagramm aller im Rahmen dieser Arbeit zu erstellenden Klassen.

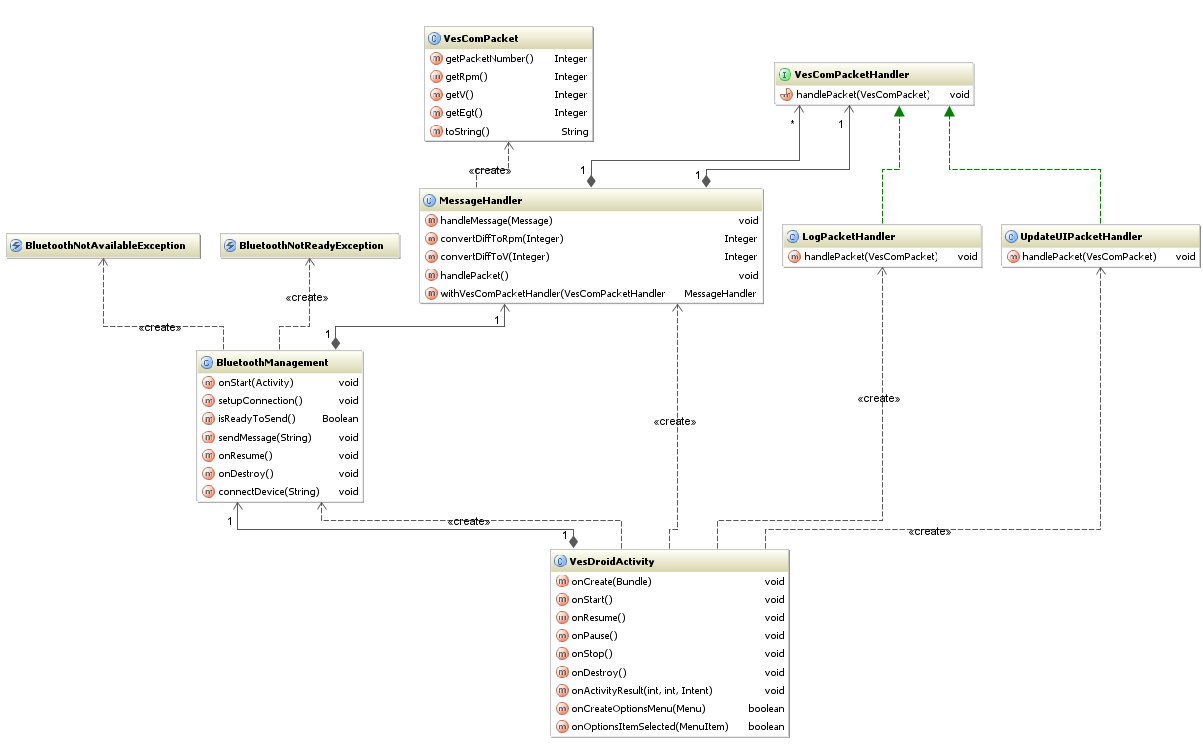


Abbildung ‑ - Klassendiagramm Visualisierungseinheit

#### Paket de.gaffa.vescom.vesdroid

Das Paket *de.gaffa.vescom.vesdroid* ist das „Basispaket“ der Anwendung. Es enthält die Pakete *exception*, *packet* und *bluetoothmanagement* deren Klassen in den Folgeabschnitten beschrieben werden sowie die Klasse *VesDroidActivity* die die Android-Activity zur Visualisierung der Daten enthält und alle anderen Objekte initialisiert/instantiiert.

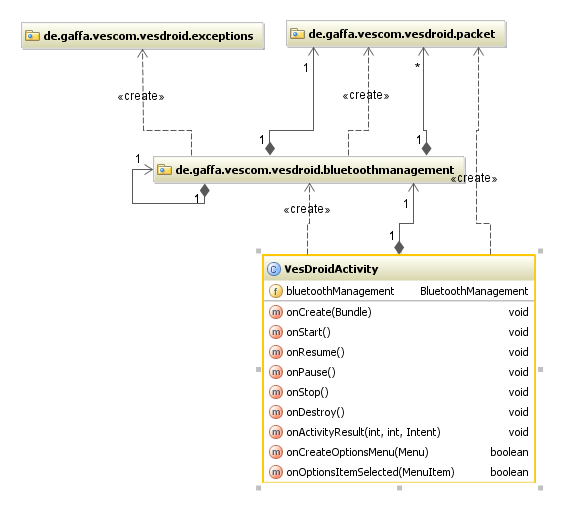


Abbildung ‑ - Klassendiagramm Paket de.gaffa.vescom.vesdroid

#### Paket de.gaffa.vescom.vesdroid.bluetoothmanagement

Das Paket *de.gaffa.vescom.vesdroid.bluetoothmanagement* ist für die Bluetooth-Kommunikation zuständig. Es enthält die Klassen *Bluetoothmanagement* und *MessageHandler*. Die Aufgabe von *Bluetoothmanagement* ist die Verwaltung des Bluetooth-Adapters sowie das Erstellen und Pflegen von Verbindungen zu Geräten (dem Messsystem). *Messagehandler* ist dafür zuständig, die seriell erhaltenen Daten in ein vereinbartes Format (*VesComPacket*) zu parsen und registrierte Handler vom Typ *VesComPacketHandler* mit der weiteren Verarbeitung zu beauftragen. Hier erkennen wir das Observer-Pattern. Beliebige Klassen die das entsprechende Interface implementieren können als Handler durch den Aufruf von *withVesComPacketHandler()* registriert werden.

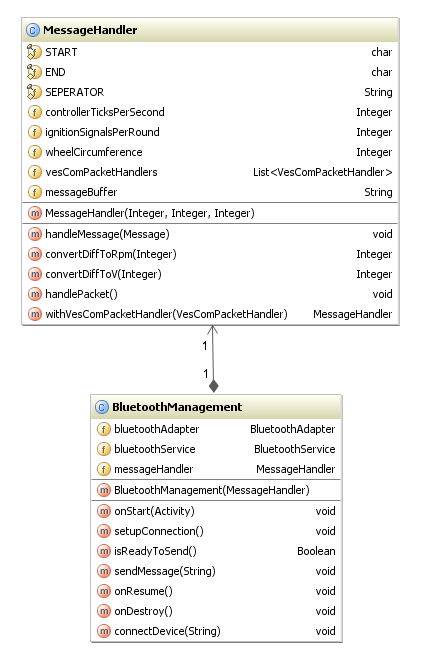


Abbildung ‑ – Klassendiagramm Paket de.gaffa.vescom.vesdroid.bluetoothmanagement

#### Paket de.gaffa.vescom.vesdroid.packet

Das Paket *de.gaffa.vescom.vesdroid.packet* enthält das Business-Objekt VesComPacket, das Interface welches Handler implementieren müssen, welche ein solches bei Empfang verarbeiten wollen sowie die Implementierungen UpdateUIPacketHandler, welche sich um die Aktualisierung der Benutzeroberfläche (also der eigentlichen Visualisierung) kümmert, und LogPacketHandler welche die empfangenen Daten ins Systemlog schreibt.

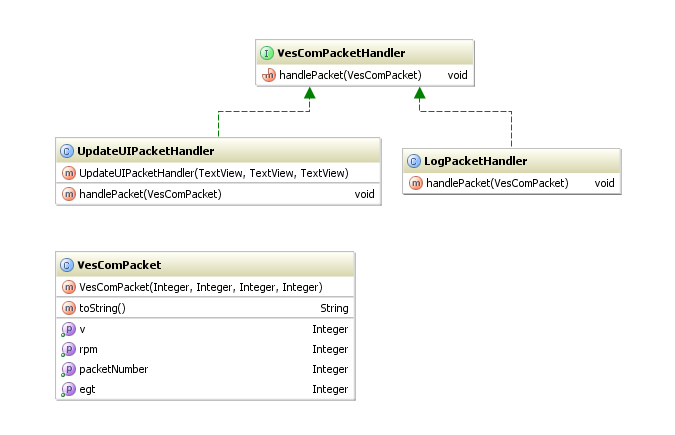


Abbildung ‑ - Klassendiagramm Paket de.gaffa.vescom.vesdroid.packet

#### Android SDK: BluetoothChat

Das im Android SDK enthaltene Beispielprogramm Bluetoothchat wurde als Basis für die Kommunikationsimplementierung in der Visualisierungseinrichtung genutzt. So wurde die Klasse DeviceListActivity zur Auswahl eines gekoppelten Bluetoothgerätes verwendet und auch die Threadimplementierung zum Verbindungsmanagements des BluetoothSerivce wurde mit Modifikationen übernommen.

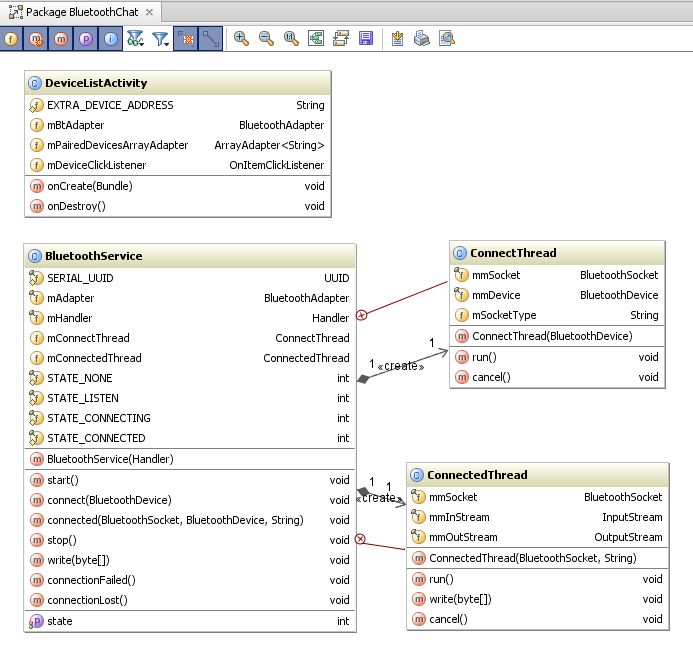


Abbildung ‑ - Klassendiagramm Android SDK: BluetoothChat

## Paketserialisierung/Deserialisierung

Zur Kommunikation zwischen der Messeinrichtung und der Einrichtung zur Verarbeitung und Visualisierung der Messdaten wurde ein Datenpaket vereinbart (siehe „Anforderungen, Systementwurf und Realisierung/Datenpaket“). Dieses Paket muss auf Seite der Messeinrichtung seriell via Bluetooth übertragen und auf Seite der Einrichtung zur Verarbeitung und Visualisierung deserialisiert werden damit es weiter verarbeitet werden kann. Im Folgenden werden die Lösungen für dieses Problem anhand von Codebeispielen erläutert.

### Serialisierung in der Messeinheit

Der Beginn des Pakets wird durch einen Stern (Ascii 42, „\*“) markiert. Es folgt die aktuelle Paketnummer, wiederum gefolgt von einem Komma (Ascii 44, „,“). Nun werden noch Drehzahlzeitdifferenz, Tachozeitdifferenz und Abgastemperatur hinzugefügt. Jeweils getrennt durch ein Komma. Das Ende des Pakets wird durch eine Raute (Ascii 35, „#“) markiert. Die so aufgebaute Zeichenkette wird dann über die serielle Schnittstelle an das Bluetooth-Modul übermittelt welches sie überträgt.

/\*\*

\* sends a vescompacket via uart from given values

\*/

void sendPacket(int rpm\_diff, int v\_diff, int egt){

char message[29];

int pointer = 0;

message[pointer] = '\*';

pointer++;

pointer += addIntToString(message, pointer, getPacketNumber());

message[pointer] = ',';

pointer++;

pointer += addIntToString(message, pointer, rpm\_diff);

message[pointer] = ',';

pointer++;

pointer += addIntToString(message, pointer, v\_diff);

message[pointer] = ',';

pointer++;

pointer += addIntToString(message, pointer, egt);

message[pointer] = '#';

message[pointer+1] = '\0';

uart\_puts(message);

}

Die hier genutzte Funktion addIntToString() wandelt einen Integer-Wert in ein Character-Array (String) und fügt sie dem per Parameter übergebenen String hinzu. Die Funktion getPacketNumber() ist dafür zuständig, einen Paketzähler zu inkrementieren und somit stets die nächsthöhere Paketnummer zur Verfügung zu stellen sodass eine fortlaufende Paketnummerierung gewährleistet wird.

### Deserialisierung in der Visualisierungseinheit

Im MessageHandler wird ein Buffer vorgehalten. Dieser wird beim Empfang des Startzeichens („\*“) neu initialisiert. Darauf folgende Zeichen werden bis zum Empfang des Endzeichens („#“) in den Puffer geschrieben.

public void handleMessage(Message msg) {

// handle message received event

if (msg.what == Properties.MESSAGE\_READ) {

byte[] readBuf = (byte[]) msg.obj;

// construct a string from the valid bytes in the buffer

String readMessage = new String(readBuf, 0, msg.arg1);

for (char c : readMessage.toCharArray()) {

if (c == START) {

messageBuffer = new String();

} else if (c == END) {

handlePacket();

} else if (messageBuffer != null) {

messageBuffer += c;

}

}

}

}

Beim Empfang des Endzeichens („#“) wird die Methode handlePacket() aufgerufen welche die Deserialisierung vornimmt. Hierbei wird der Puffer am Seperator (“,”) geteilt. Sind danach vier Werte vorhanden wird von einem gültigen Paket ausgegangen und die empfangenen Daten werden verarbeitet und in eine VescomPacket-Instanz überführt.

private void handlePacket() {

// create value-array from string by splitting at the separator-character

String[] fields = messageBuffer.split(SEPARATOR);

try {

if (fields.length != 4) {

// message seems to be invalid as 4 values are expected

throw new Exception("Message contained more or less than 4 fields");

}

/\*\*

\* assign values

\*/

Integer packetNumber = Integer.parseInt(fields[0]);

Integer rpm = convertDiffToRpm(Integer.parseInt(fields[1]));

Integer v = convertDiffToV(Integer.parseInt(fields[2]));

Integer egt = Integer.parseInt(fields[3]);

// create packet with values

VesComPacket packet = new VesComPacket(packetNumber, rpm, v, egt);

}

} catch (Throwable tr) {

// log all exceptions that occur

Log.e("VesDroid", "Fatal. A Message could not be converted to a VesDroid-Packet.", tr);

}

}

So ist die Visualisierung der nunmehr aufgearbeiteten Daten möglich. Zum tieferen Verständnis der Implementierung wird das Studium der beigefügten Quellen empfohlen.

# Zusammenfassung und Ausblick

Bei der Lösung der Aufgabenstellung hat sich die Arbeit mit Signalen aus der „realen Welt“ als schwieriger als erwartet erwiesen. Störungsumfelder und nicht erwartete Signalschwankungen wirken sich gravierend aus. Schnell führt eine Rechenoperation auf einen fehlerhaft erfassten Wert zum Controllerabsturz oder wird ein Eingangspin oder ein ganzer Mikrocontroller zerstört. Solche unvorhergesehenen Ereignisse haben die Verteilung des zeitlichen Aufwandes bei Bearbeitung der Arbeit in erheblichem Maße gegen die Erwartung/Planung beeinflusst sodass der Aufwand der praktische Umsetzung deutlich über der Zeitplanung lag. Dennoch sind alle Maßgeblichen Kriterien erfüllt und das Projekt kann als erfolgreich bewertet werden. Zu einer Markt- bzw. Serienreife fehlen noch Konfigurationsfeatures (zum Beispiel des Reifenumfangs, der Anzahl der Magnete am Reifen, der Anzahl der Spulen/Magnete der Lichtmaschine). Auch hat das minimale Benutzerinterface der Visualisierungseinrichtung Prototypcharakter. Ereignisse wie Stillstand des Motors oder des Fahrzeuges und damit einhergehende Manipulation des Drehzahl- bzw. Tachowertes auf 0 wurde nicht implementiert.

Theoretisch mögliche Features wie das Errechnen einer Leistungskurve aus den erfassten Daten würde ein Datenübertragungsprotokoll erfordern welches mit dem Verlust von Paketen umgehen kann (zB. erneutes Senden der Messeinrichtung bis Bestätigung der Visualisierungseinheit).

Weitere Features wie Warnung bei Überschreiten einer konfigurierbaren Abgastemperatur, Speichern und Visualisieren der zurückgelegten Distanz, Tageskilometerzähler sind denkbar. Das Projekt wird nach Abschluss der Arbeit vom Autor der Community zur Verfügung gestellt und dort weiter entwickelt.

# Abbildungsverzeichnis

[Abbildung 3‑1 - Leistungsdiagramm des Testfahrzeugs dieser Arbeit 6](#_Toc341536548)

[Abbildung 5‑1 - Hall-Effekt, Quelle: Wikipedia [1] 11](#_Toc341536549)

[Abbildung 5‑2 - Blockdiagramm Hall-IC, Quelle: Infineon Datenblatt TLE49x5L [2] 12](#_Toc341536550)

[Abbildung 5‑3 - Hall-IC-Schaltung auf Stator 13](#_Toc341536551)

[Abbildung 5‑4 - Induktionsmessung am Zündkabel 14](#_Toc341536552)

[Abbildung 5‑5 - Lichtmaschinensignal Vespa 15](#_Toc341536553)

[Abbildung 5‑6 - Lichtmaschine mit Diode 15](#_Toc341536554)

[Abbildung 5‑7 - Lichtmaschinensignal mit Diode, Filter und Spannungsbegrenzung 16](#_Toc341536555)

[Abbildung 5‑8 - Tachosensor-Prototyp befestigt mit Kabelbinder. Dauermagnet an Felge 18](#_Toc341536556)

[Abbildung 5‑9 - Thermoelement, Quelle: Wikipedia [3] 19](#_Toc341536557)

[Abbildung 5‑10 - Blockdiagramm MAX6675, Quelle: Maxim-Datenblatt MAX6675 [4] 20](#_Toc341536558)

[Abbildung 5‑11 - Protokoll MAX6675, Quelle: Maxim-Datenblatt MAX6675 [4] 20](#_Toc341536559)

[Abbildung 5‑12 - Datenframe/Bits MAX6675, Quelle: Maxim-Datenblatt MAX6675 [4] 21](#_Toc341536560)

[Abbildung 5‑13 - Ablaufdiagramm Kommunikation mit Max6675 22](#_Toc341536561)

[Abbildung 5‑14 -Mikrocontroller kommuniziert über serielle Kabelverbindung mit Android-Mobiltelefon 24](#_Toc341536562)

[Abbildung 5‑15 - Auszug des Schaltplans für Kommunikation via Can-Bus 25](#_Toc341536563)

[Abbildung 5‑16 - Display eines Prototypen am Vespa-Lenker 26](#_Toc341536564)

[Abbildung 5‑17 - Display mit Störungen 26](#_Toc341536565)

[Abbildung 5‑18 - Schaltplan Prototyp integrierte Mess-, Auswerte- und Visualisierungseinheit 27](#_Toc341536566)

[Abbildung 5‑19 - Prototyp integrierte Mess-, Auswerte- und Visualisierungseinheit 28](#_Toc341536567)

[Abbildung 5‑20 - Prototyp integrierte Mess-, Auswerte- und Visualisierungseinheit 28](#_Toc341536568)

[Abbildung 6‑1 - Systemübersicht 30](#_Toc341536569)

[Abbildung 6‑2 -Blockschaltbild Bluecontroller, Quelle: www.bluecontroller.com 32](#_Toc341536570)

[Abbildung 6‑3 - Bluecontroller, Bordlayout mit Legende, Quelle: www.bluecontroller.com 33](#_Toc341536571)

[Abbildung 6‑4 - Gesamtschaltplan Messsystem 34](#_Toc341536572)

[Abbildung 6‑5 - Schaltplan Spannungsregelung Messsystem 35](#_Toc341536573)

[Abbildung 6‑6 - Schaltplan Drehzahlerfassung Messsystem 36](#_Toc341536574)

[Abbildung 6‑7 - Schaltplan Beschaltung Max6675 37](#_Toc341536575)

[Abbildung 6‑8 - Schaltplan Beschaltung TLE49x5 37](#_Toc341536576)

[Abbildung 6‑9 - Klassendiagramm Visualisierungseinheit 40](#_Toc341536577)

[Abbildung 6‑10 - Klassendiagramm Paket de.gaffa.vescom.vesdroid 41](#_Toc341536578)

[Abbildung 6‑11 – Klassendiagramm Paket de.gaffa.vescom.vesdroid.bluetoothmanagement 42](#_Toc341536579)

[Abbildung 6‑12 - Klassendiagramm Paket de.gaffa.vescom.vesdroid.packet 43](#_Toc341536580)

[Abbildung 6‑13 - Klassendiagramm Android SDK: BluetoothChat 44](#_Toc341536581)

# Quellenverzeichnis

[1] Hall-Effekt   
<http://de.wikipedia.org/wiki/Hall-Effekt>

[2] Infineon Datenblatt TLE49x5 Hall-ICs  
<http://www.infineon.com/dgdl/Data_Sheet_TLE49x5L_Family_V1_5.pdf?folderId=db3a304412b407950112b408e8c90004&fileId=db3a304412b407950112b4090265005b>

[3] Thermoelement  
<http://de.wikipedia.org/wiki/Thermoelement>

[4] Maxim Datenblatt MAX6675 Typ-K Thermoelement ICs  
<http://datasheets.maxim-ic.com/en/ds/MAX6675.pdf>

[5] Bluetooth-Logo  
<http://de.wikipedia.org/wiki/Bluetooth>

# !!!!!!!!!!!!!!!!!!!!TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

USB-Stick mit Anhang (Source, Schaltplan, Layout) dazulegen

Anhang (Code, Schaltpläne, Layouts) ausgedruckt/gebunden dazulegen

Anhang Selbstständigkeitserklärung dazulegen <http://fb6.beuth-hochschule.de/pub/attach/Formulare/selbstaendig-verfasst-erklaerung.pdf>

Unterzeichnen Selbstständigkeitserklärung in den Arbeiten