

Martin Pyka

Simon Kesting

Mikrocontrollersysteme WS19/20

Prof. Dr.-Ing. Karl-Ragmar Riemschneider

Fahrzeug-Inforamtions-Display

Analoger Tachometer, Digitaler Tageskilometerzähler, Bewegungsrichtung

Inhaltsverzeichnis

[Einleitung 2](#_Toc29217817)

[1. Planung 3](#_Toc29217818)

[1.1 Laboraufteilung 3](#_Toc29217819)

[1.2 Zeitplan und Arbeitsaufteilung im Team 4](#_Toc29217820)

[2. Konzept 5](#_Toc29217821)

[2.1 Bewegungsrichtung – und Raddrehzahl – Messung 5](#_Toc29217822)

[2.1.1 Messung der Bewegungsrichtung 6](#_Toc29217823)

[2.1.2 Messung der Raddrehzahl 6](#_Toc29217824)

[2.2 Aufteilung des Grafik-Display 7](#_Toc29217825)

[2.3 Test und Realisierung 8](#_Toc29217826)

[3. Programm 9](#_Toc29217827)

[3.1 Initialisierungen 13](#_Toc29217828)

[3.2 Interrupt-handler 15](#_Toc29217829)

[3.3 LC-Display 18](#_Toc29217830)

[3.3.1 Touch 19](#_Toc29217831)

[3.3.2 Display 20](#_Toc29217832)

[4. Ergebnisse 27](#_Toc29217833)

[4.1 Planung 27](#_Toc29217834)

[4.2 Konzept 27](#_Toc29217835)

[4.3 Programm 27](#_Toc29217836)

[Fazit 28](#_Toc29217837)

[Tabellenverzeichnis 29](#_Toc29217838)

[Abbildungsverzeichnis 29](#_Toc29217839)

[Quellenverzeichnis 29](#_Toc29217840)

# Einleitung

Im Rahmen der Vorlesung zu Mikrocontroller-Systeme wurde die Projektaufgabe Fahrzeug-Informations-Display bearbeitet.

Die Aufgabe besteht darin, ein analoges Tachometer zur Geschwindigkeitsanzeige, einen digitalen Tageskilometerzähler und die Ausgabe der Bewegungsrichtung zu entwickeln.

Die Entwicklung besteht in dem Entwurf, der Programmierung und dem Test mithilfe der vorgegebenen Rechnersysteme und Peripherie. [1]

Im Labor wurde das Programm mit der Entwicklungssoftware Code Composer Studio CCS von Texas Instruments in der Programmiersprache C realisiert.

Die Tiva Evaluation Boards mit dem Mikrocontroller TM4C1294 wurden zur Ansteuerung des Touch-Screen-Controllers XPT2046 und einem Wheel Sensor-Modul verwendet. [2]

# 1. Planung

Zu Beginn wurde das Projekt zeitlich und inhaltlich geplant. Dazu wurde ein Projektplan mit den einzelnen Arbeitsschritten erstellt.

## 1.1 Laboraufteilung

Zunächst wurde in dem Team festgehalten, wie die Labortermine genutzt werden, um eine grobe Einteilung festzulegen.

Grundlage

* Modulare Aufteilung der Projektdateien
* Wiederholte Absprachen vor den einzelnen Terminen

Termin 1

* Rechnersysteme und Peripherie untersuchen
* Wheel Sensor -> Signalanalyse, Versuche
* Konzept Anpassung

Termin 2

* Display: grundlegende Funktionen, sinnvolle Darstellungen
* Schnittstellen Definition
* Interrupt handler zur zeitlichen Abstimmung der Signale

Termin 3

* Wheel Sensor -> S1 und S2 analysieren
* Interrupt handler
* Anzeige der analysierten Signale

Termin 4

- Systemtest (gesamt)

Grundlegend waren wiederholte Absprachen vor den einzelnen Terminen geplant, um die Labortermine effektiv nutzen zu können.

## 1.2 Zeitplan und Arbeitsaufteilung im Team

Der geschätzte Zeitbedarf zu jedem Arbeitsschritt und die Zuordnung zu den Teammitgliedern wurde in einer Excel-Tabelle dokumentiert.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Week |  | 42 | 42 | 43 | 43 | 44 | 44 | 45 | 45 | 46 | 46 | 47 | 47 | 48 | 48 | 49 | 49 |
|  |  | Plan | Real | Plan | Real | Plan | Real | Plan | Real | Plan | Real | Plan | Real | Plan | Real | Plan | Real |
| Concept/Planning | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | Simon | 4 | **4** | 1 | **1** | 1 | **0** | 2 | **4** | 1 | **1** | 1 | **0** | 2 | **1** | 1 | **1** |
|  | Martin | 3 | **3** | 1 | **1** | 1 | **0** | 2 | **5** | 1 | **1** | 1 | **0** | 2 | **1** | 1 | **1** |
| Display |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | Simon | 2 | **2** | 4 | **5** | 2 | **1** |  |  | 2 | **2** |  |  |  |  |  |  |
|  | Martin | 2 | **2** | 3 | **4** | 3 | **2** |  |  | 3 | **2** |  |  |  |  |  |  |
| Wheel Sensor-Modul (Interrupts) | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | Simon |  |  |  |  | 4 | **3** | 2 | **5** | 6 | **4** | 2 | **2** | 4 | **4** |  |  |
|  | Martin |  |  |  |  | 6 | **4** | 2 | **8** | 6 | **4** | 2 | **1** | 5 | **4** |  |  |
| Test/Verification | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | Simon |  |  | 1 | **1** | 1 | **0** | 2 | **4** | 1 | **0** | 0 | **1** | 1 | **1** | 3 | **4** |
|  | Martin |  |  | 1 | **1** | 1 | **0** | 2 | **4** | 1 | **0** | 0 | **1** | 1 | **1** | 3 | **5** |
| Documentation | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | Simon |  |  | 1 | **1** | 1 | **0** | 1 | **2** | 1 | 0 | 1 | **2** | 0 | **1** | 1 | **1** |
|  | Martin |  |  | 1 | **1** | 1 | **0** | 1 | **1** | 1 | 1 | 1 | **2** | 1 | **1** | 1 | **1** |
| Presentation | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | Simon |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | Martin |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Sum | Plan | 11 |  | 13 |  | 21 |  | 14 |  | 23 |  | 8 |  | 16 |  | 10 |  |
| Sum | **Ist** |  | **11** |  | **15** |  | **10** |  | **33** |  | **15** |  | **9** |  | **14** |  | **13** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Week |  | 50 | 50 | 51 | 51 | 52 | 52 | 1 | 1 | Simon |  | Martin |  | Total |  |
|  |  | Plan | Real | Plan | Real | Plan | Real | Plan | Real | Plan | Real | Plan | Real | Plan | Real |
| Concept/Planning | |  |  |  |  |  |  |  |  |  |  |  |  | 29 | 28 |
|  | Simon | 1 | **0** | 1 | **0** | 0 | **1** |  |  | 15 | **13** |  |  |  |  |
|  | Martin | 1 | **2** | 1 | **0** | 0 | **1** |  |  |  |  | 14 | **15** |  |  |
| Display |  |  |  |  |  |  |  |  |  |  |  |  |  | 21 | 20 |
|  | Simon |  |  |  |  |  |  |  |  | 10 | **10** |  |  |  |  |
|  | Martin |  |  |  |  |  |  |  |  |  |  | 11 | **10** |  |  |
| Wheel Sensor-Modul (Interrupts) | |  |  |  |  |  |  |  |  |  |  |  |  | 39 | 45 |
|  | Simon | 0 | **1** |  |  | 0 | **2** |  |  | 18 | **21** |  |  |  |  |
|  | Martin | 0 | **1** |  |  | 0 | **2** |  |  |  |  | 21 | **24** |  |  |
| Test/Verification | |  |  |  |  |  |  |  |  |  |  |  |  | 23 | 33 |
|  | Simon | 3 | **2** | 0 | **2** |  |  | 0 | **1** | 12 | **16** |  |  |  |  |
|  | Martin | 2 | **1** | 0 | **3** |  |  | 0 | **1** |  |  | 11 | **17** |  |  |
| Documentation | |  |  |  |  |  |  |  |  |  |  |  |  | 30 | 30 |
|  | Simon | 1 | **1** | 3 | **2** | 5 | **6** | 1 | **0** | 16 | **16** |  |  |  |  |
|  | Martin | 0 | **1** | 3 | **2** | 3 | **2** | 1 | **2** |  |  | 14 | **14** |  |  |
| Presentation | |  |  |  |  |  |  |  |  |  |  |  |  | 10 | 12 |
|  | Simon |  |  | 2 | **2** | 2 | **2** | 1 | **2** | 5 | **6** |  |  |  |  |
|  | Martin |  |  | 2 | **2** | 2 | **2** | 1 | **2** |  |  | 5 | **6** |  |  |
| Sum | Plan | 8 |  | 12 |  | 12 |  | 4 |  | 76 |  | 76 |  | 152 |  |
| Sum | **Ist** |  | **9** |  | **13** |  | **18** |  | **8** |  | **82** |  | **86** |  | **168** |

Tabelle : Excel-Tabelle mit Zeitplan und Arbeitsaufteilung

# 2. Konzept

Die Beschreibung des Konzeptes wird in Abschnitte unterteilt, welche sich an der Aufgabenstellung orientieren.

## 2.1 Bewegungsrichtung – und Raddrehzahl – Messung

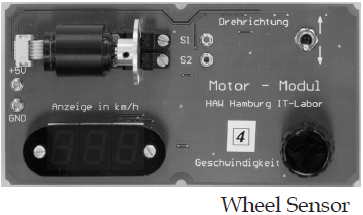
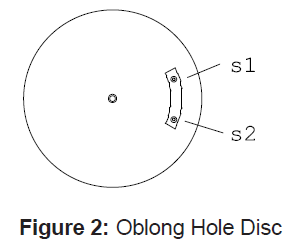


Abbildung : Peripheriebaustein - Rad Sensor und Signale s1, s2

[1]

Das „Wheel Sensor“ – Modul enthält einen Motor, welcher eine Scheibe mit zwei Löchern antreibt. Die Drehgeschwindigkeit und Drehrichtung kann verändert werden.

Es werden zwei kurze High-Signale (s1 und s2) bei jeder Umdrehung ausgegeben.

Aus der Reihenfolge der High-Impulse lässt sich die Bewegungsrichtung ermitteln.

Die Signale s1 und s2 müssen logisch erfasst werden.

Mit dem Oszilloskop wurde gemessen, um das Zeitverhalten zu analysieren.

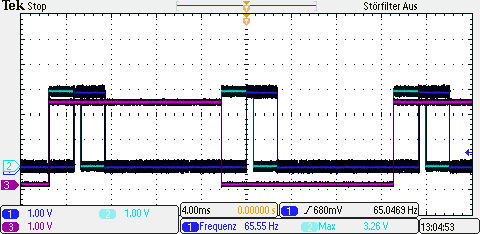


Abbildung : Oszilloskop, Messung des Zeitverhaltens der Sensorsignale

Es ist zu erkennen, dass in einem kurzen Moment beide Sensorsignale s1 und s2 auf high sind. Die Drehrichtung entscheidet darüber, welches Signal zuerst auf high geht.

### 2.1.1 Messung der Bewegungsrichtung

Variante 1:

Die High-Signale von s1 und s2 werden von der Interrupt Service Routine einzeln gezählt. In der ISR wird die Drehrichtung gesetzt. Durch das einzelne zählen von s1 und s2 kann die Drehrichtung erkannt werden. Wenn der Zähler von s1 größer als der Zähler von s2 ist kann man von einer Vorwärtsdrehung ausgehen. Wenn der Zähler von s1 kleiner als der Zähler von s2 ist kann man von einer Rückwärtsdrehung ausgehen. Wenn beide Zähler gleich groß sind bleibt man in der aktuellen Richtung. Die ISR beider Signale sind in der Lage die Drehrichtung zu verändern. Da pro Umdrehung das erste Signal um einen größer ist als das zweite Signal, setzt das erste Signal die Drehrichtung. Das Zweite erkannte Signal hat den gleichen Wert wie sein Vorgänger und ändert nichts an der Richtung.

Variante 2:

Es wird geprüft, welche Zustände bei einem anliegenden Signal bestehen. Der Interrupt welcher zuerst aktiviert wird, prüft den Zustand des jeweils anderen. Die Reihenfolge der Signale ist entscheidend, welche Richtung erkannt wird.

Eine Darstellung in Form einer reduzierten „sate-machine“ ist möglich. Die Zustände, dass beide Sensor-Signale high sind sowie der darauffolgende Zustand, dass das erste Signal von High auf Low wechselt, werden nicht benötigt. Die steigenden Flanken der Signale werden betrachtet.

### 2.1.2 Messung der Raddrehzahl

Variante 1:

Für die Raddrehzahl Erkennung werden die High-Signale von s1 und s2 von der Interrupt Service Routine gezählt. Mit einem Timer werden zu fest definierten Zeiten der Zählerwert der Signale s1 oder s2 abgespeichert. Mit dem festen Zeitabstand und der Anzahl der Umdrehungen kann die Drehzahl/sec berechnet werden.

Drehzahlt/s = (Zählerwert/Timer\_Zeit) \* (Wert, damit aus Timer\_zeit \* Wert = 1s/1000ms)

Geschwindigkeit = Radumfang(1m) \* Drehzahl/s

## 2.2 Aufteilung des Grafik-Display

Der „Start Up“- Bildschirm wird direkt nach der Initialisierung in das Grafik-Display geladen.

#### Start Up

Abbildung : Display - Start Up

Die Balkenfarbe und die Kasten Farbe für die Fahrtrichtung passen sich an die Fahrtrichtung an: Vorwärts (grün), Rückwärts (gelb).

#### Vorwärts

V

123,47km

Abbildung : Display - Vorwärts

Durch Berührung lässt sich die Tageskilometerzähler-Anzeige auf die digitale Geschwindigkeits-Anzeige ändern.

#### Rückwärts

128,31 km/h

R

Abbildung : Display - Rückwärts

## 2.3 Test und Realisierung

Es ist die Frage zu beantworten, ob es sinnvoll sei das Fahrzeug-Informations-Display während der Testphase auf dem PC-Bildschirm auszugeben.

Es wäre sinnvoll für Arbeiten außerhalb des Labors, allerdings könnte sich das Display in dem Labor anders verhalten. Das Ziel ist es, das System vollständig im Labor zu testen. Mit vorhandener Hardware wäre die Darstellung auf dem PC nicht nötig.

Es wird ein Interrupt-Handler benötigt. Die Analyse der Wheel-Sensor Signale S1 und S2 müssen im Zweiten Termin mithilfe von Interrupts und Timern analysiert werden. Interrupts sind sinnvoll, da die Signale der Drehzahlsensoren zeitlich stark variieren. Die Polling variante führt dazu, dass Signale möglicherweise nicht erkannt werden. Da aktiv auf die Signale gewartet werden muss, kann man keine anderen Prozesse ausführen.

Die Interrupts werden von den Sensoren über die Pins direkt getriggert.

Die Interrupt Handler zählen die Umdrehungsanzahl.

Grundlegende Aufgaben in Bezug auf die Realisierung sind eine übersichtliche, hierarchische Codierung und die Dokumentation der Ergebnisse. Mithilfe von Git findet eine Versionierung statt.

Zur Realisierung des Displays ergeben sich folgende Aufgaben:

* Bibliothek für „Buchstaben und Zahlen zeichnen“
* Segmentdarstellung
* Wechsel von Tageskilometer zu Digitale Geschwindigkeit durch Touch
* Schnittstelle zur Verfügung stellen
* Start Up Bild einmalig starten und einzelne Segmente überschreiben
* Nur das zu ändernde Feld löschen und neu beschreiben

Zur Realisierung der Drehzahlmessung ergeben sich folgende Aufgaben:

* Drehrichtung erkennen (s1 u s2)
* Pins Initialisieren (AF)
* Interrupts initialisieren (Pins)
* I-Handler schreiben
* Zählen (Overflow abfangen)
* Drehrichtung Setzen (Vergleich der Zähler)
* Definition von Vorwärts (V) und Rückwärts (R)
* Drehzahl berechnen (m/s)

*unsigned int detect\_frequency (unsigned int s1/s2)*

* Timer initialisieren (one shot, ISR, highest prio)
* Zählerwert von s1/s2 holen nach bestimmter Zeitdauer
* Berechnung der Drehzahl anhand von Umdrehungen pro fester Timerzeit
* Zähler der s1 und s2 Signale zurücksetzten
* Timer neustarten
* Drehgeschwindigkeit berechnen (s1/s2)

*unsigned int detect\_speed (unsigned int s1/s2)*

* Berechnung der Geschwindigkeit anhand von Umdrehung/s und Umfang des Rads.
* Testen von Interrupt und Display Kommunikation (zeitliche Abstimmung)

# 3. Programm

Im Folgenden soll die aktuelle Programmversion beschrieben werden. Dabei stehen die Programmstruktur und die Umsetzung des Echtzeitproblems mit Hilfe von Interrupt-Handlern im Vordergrund.

Grundlegend wurde bei der Codierung auf die Übersichtlichkeit und Hierarchie geachtet.

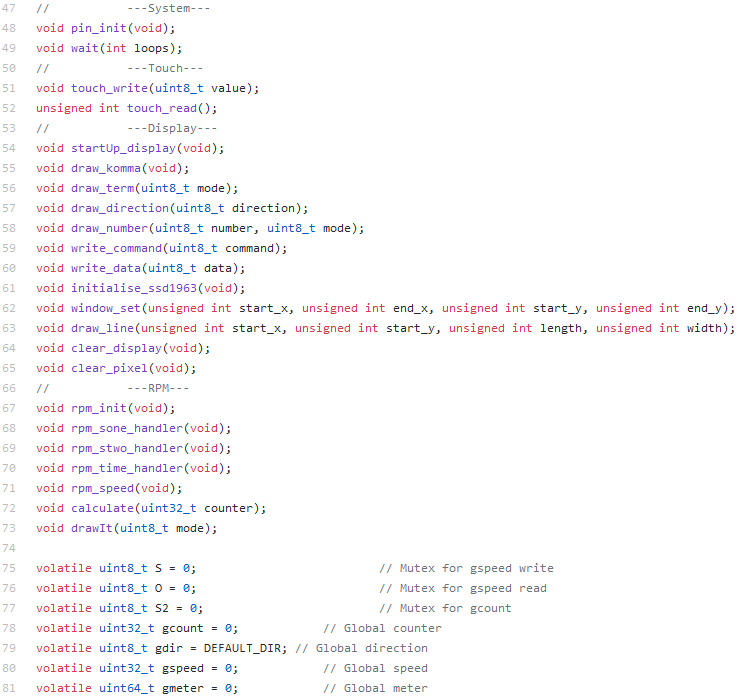
Header-Dateien wurden standardgemäß zu Beginn des Programmes eingefügt. Nachfolgend wurden Makros für verwendete Größen definiert.



Die Deklaration der Funktionen geschieht vor der „main“-Funktion, um die Realisierung und den Programmablauf mit Hilfe von Abschnitten zu verdeutlichen.

Anschließend wurden Mutex definiert, um kritische Zugriffe durch gegenseitigen Ausschluss zu verhindern.

Weitere global Initialisierte Variablen sind ebenfalls von zentraler Bedeutung im Programmverlauf.

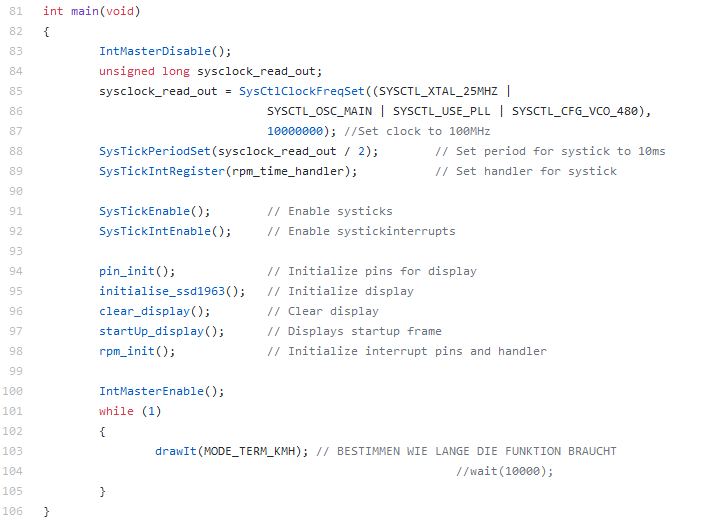


Der globale Zähler „gcount“ zählt die Umdrehungen des Rades in eine bestimmte Richtung, für einen definierten Zeitabschnitt.

In der „main“-Funktion werden alle Interrupts zu Beginn deaktiviert und am Ende wieder aktiviert. Dazwischen finden Initialisierungen und Einstellungen statt.

Zunächst wird der Systemtakt auf 100MHz eingestellt. Anschließend wird eine Periode des Systemtaktes auf 500ms gesetzt.

Der Interrupt-handler für den Systick-Timer wird in der Interrupt-Vektor-Tabelle (IVT) registriert. Danach wird der Systick-Timer gestartet und die Interrupts freigegeben.



In der „while(1)-Schleife“ wird geprüft, ob gerade gedrückt wurde, den Modus der Digitalanzeige zwischen Geschwindigkeit und Tageskilometerzähler umzuschalten.

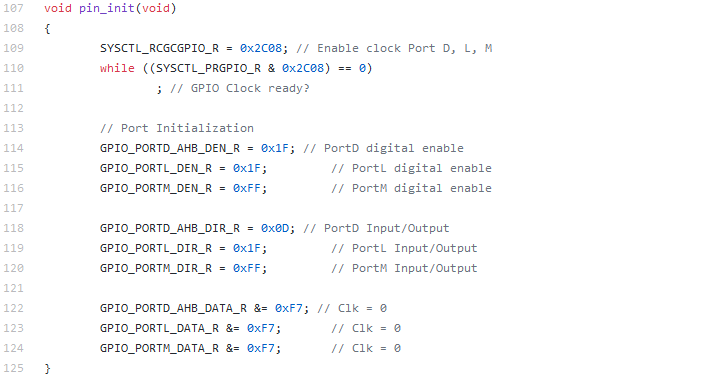
Das Display wird entsprechend im aktuellen Modus beschrieben.



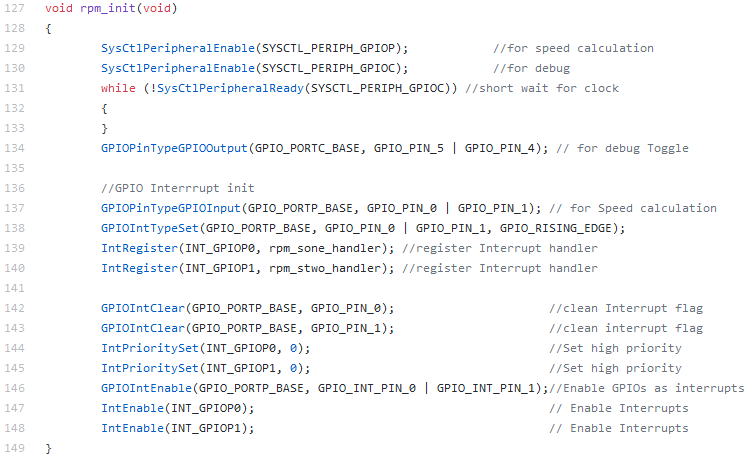
## 3.1 Initialisierungen

Die Initialisierung der Ports wurde in die Funktion „pin\_init“ ausgelagert.

Pins für die Displayansteuerung und Touch-Eingabe werden initialisiert.



In der Funktion „rpm\_init“ werden die Interrupts initialisiert. Port P wird freigeschaltet und die Pins als Interruptquellen im Rising-Edge-Modus gesetzt. Der Handler wird registriert. Interrupts werden priorisiert und freigeschaltet.







## 3.2 Interrupt-handler

Der Interrupt-handler „rpm\_sone\_handler“ verarbeitet das Eingangssignal S1 vom Wheel-Sensor-Modul.

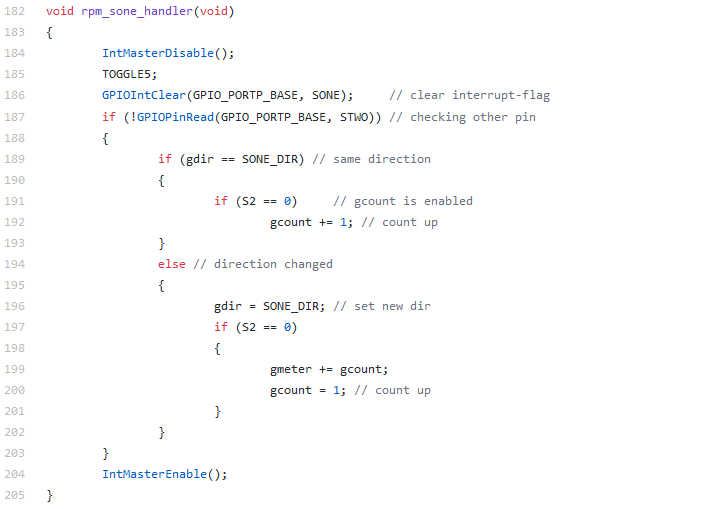
Der Interrupt-handler „rpm\_stwo\_handler“ verarbeitet das Eingangssignal S2 vom Wheel-Sensor-Modul. Durch den Rising-Edge-Modus haben wir in jedem Interrupt nur 2 Zustände. 10 und 11. Für vorwärts

01 und 11. Für rückwärts

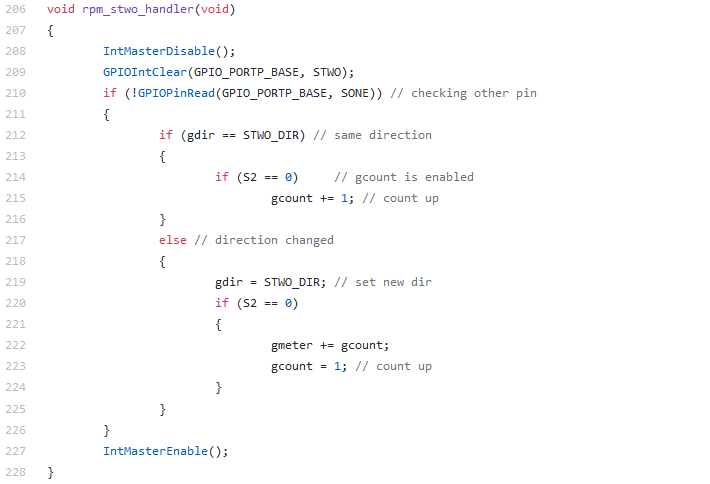
Da 11 nicht interessant ist wird geprüft ob der andere Pin high ist, wenn ja raus aus der Funktion, wenn nicht dann ist er der erste Pin, wodurch die Richtung bestimmt wird.

(S1, S2)

(1,0) (1,1) (0,1)



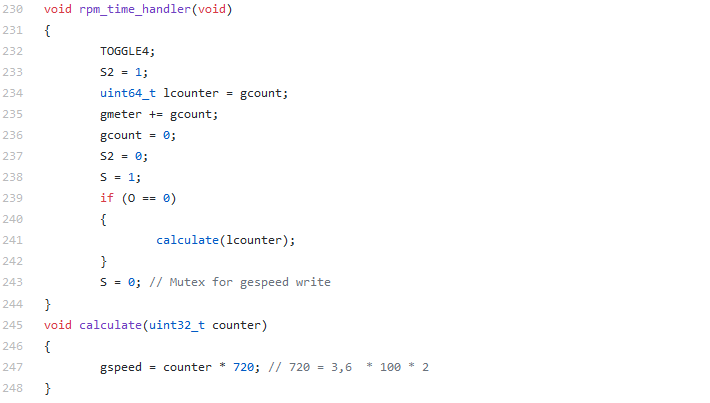
Es wurden Makros zum „Togglen“ definiert. Durch Maskierung mit einer Exklusiv-Oder-Verknüpfung, findet ein Wechsel zwischen High und Low bei jedem Aufruf statt. Dieser kann mithilfe einer LED visualisiert werden, um das Zeitverhalten der Interrupts zu analysieren.



Der Handler wird von dem Systick-Timer alle 500ms aktiviert.

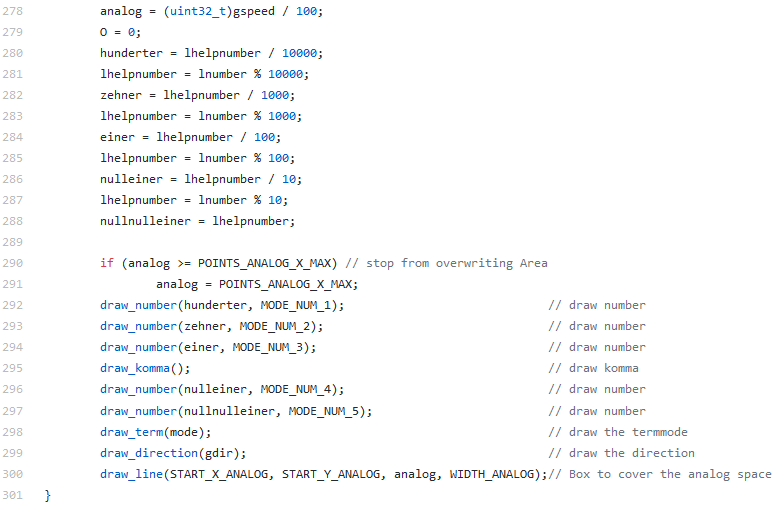
Der Interrupt-handler „rpm\_time\_handler“ liest den aktuellen Wert der globalen Zählervariable „gcount“ aus und übergibt diesen an die Funktion „calculate“ zur Berechnung der aktuellen Geschwindigkeit.

Mit Hilfe von Mutex werden kritische Zugriffe durch gegenseitigen Ausschluss verhindert.

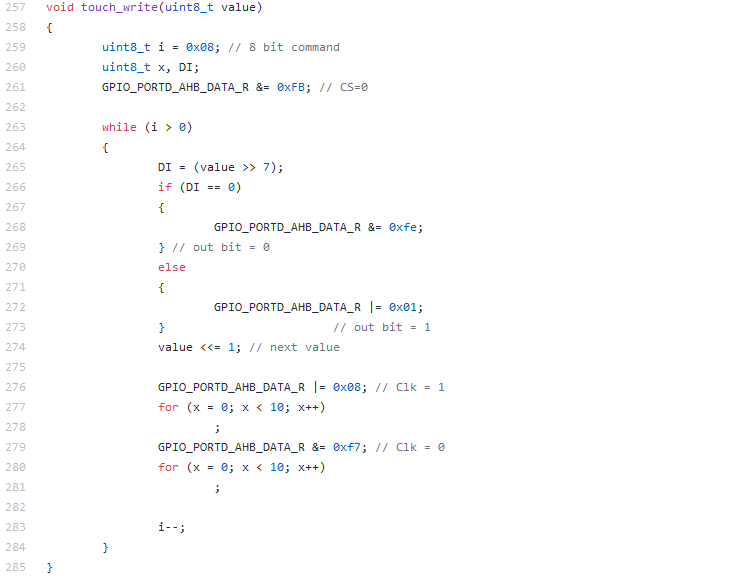


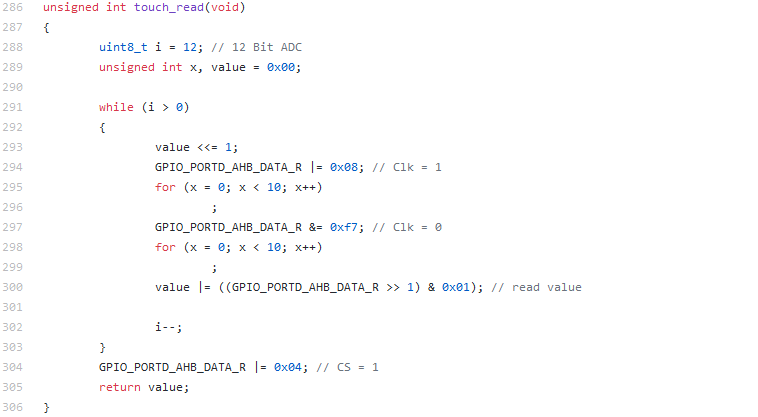
## 3.3 LC-Display



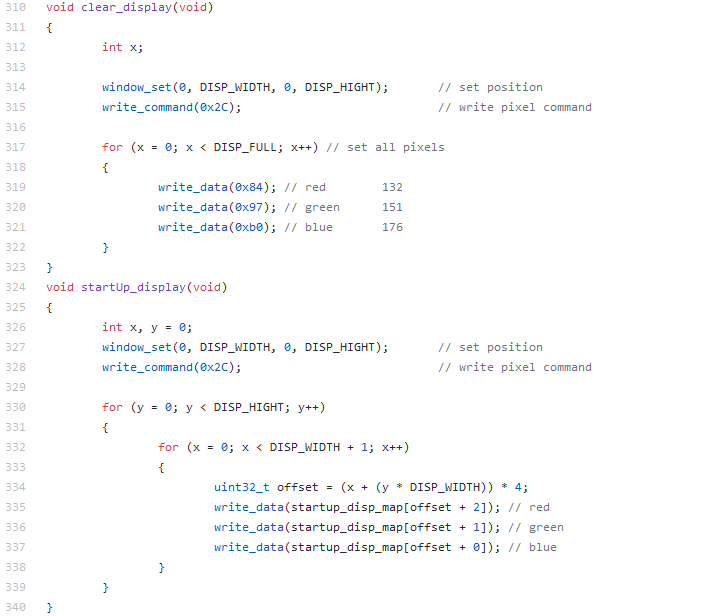


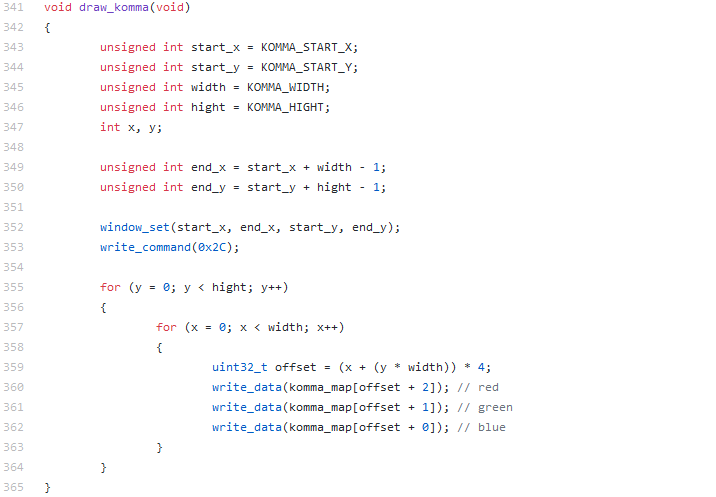
### 3.3.1 Touch





### 3.3.2 Display

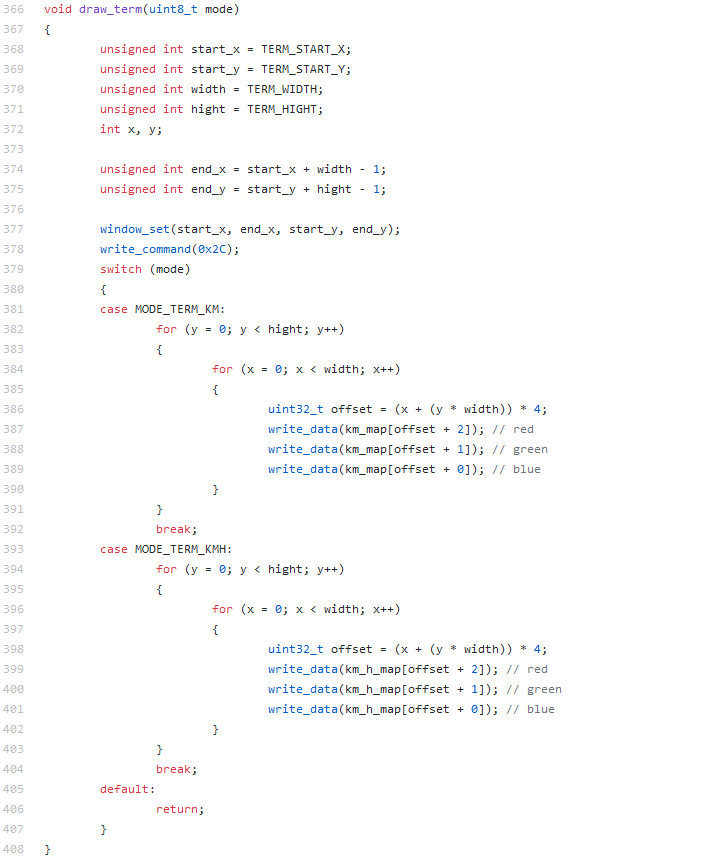


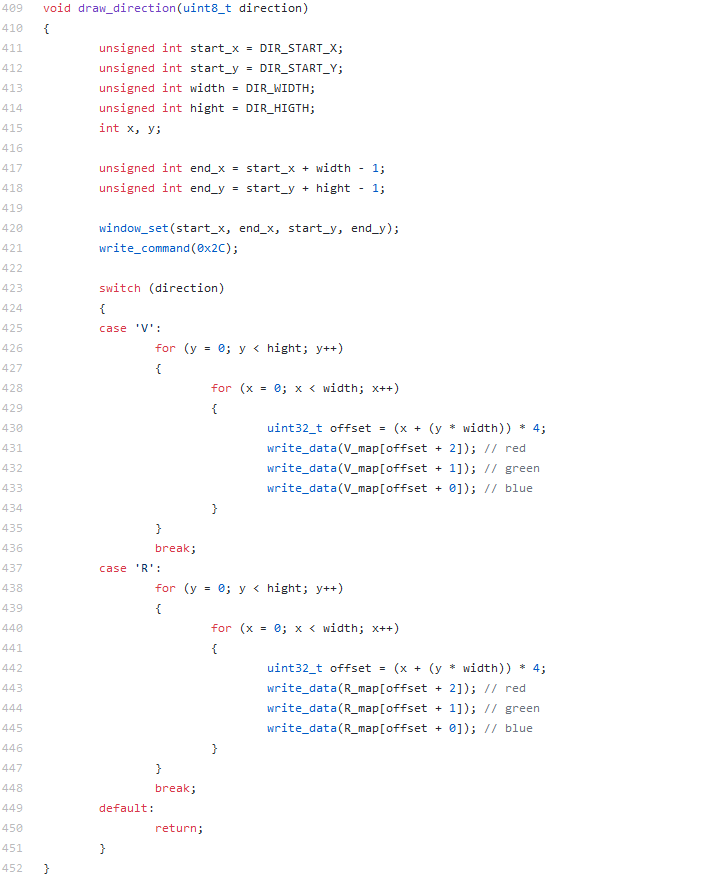


Die Darstellung der Zahlen (0-9) und Buchstaben (V, R) basiert auf Daten, welche über Paint mit Hilfe eines Online-Konverters extrahiert und in header-Dateien abgespeichert wurden.

Mit den folgenden Funktionen findet der Zugriff auf die eingefügten header-Dateien statt.

In den Maps sind die Zahlen Buchstaben und Kommata in 4 chars pro Pixel gespeichert in der Reihenfolge 8bit Blau, 8bit Grün, 8bit Rot und 8bit Platzhalter (0xff). Die Position der Zahlen und Indizes wurde in Paint anhand des Startup-Bildes ermittelt.





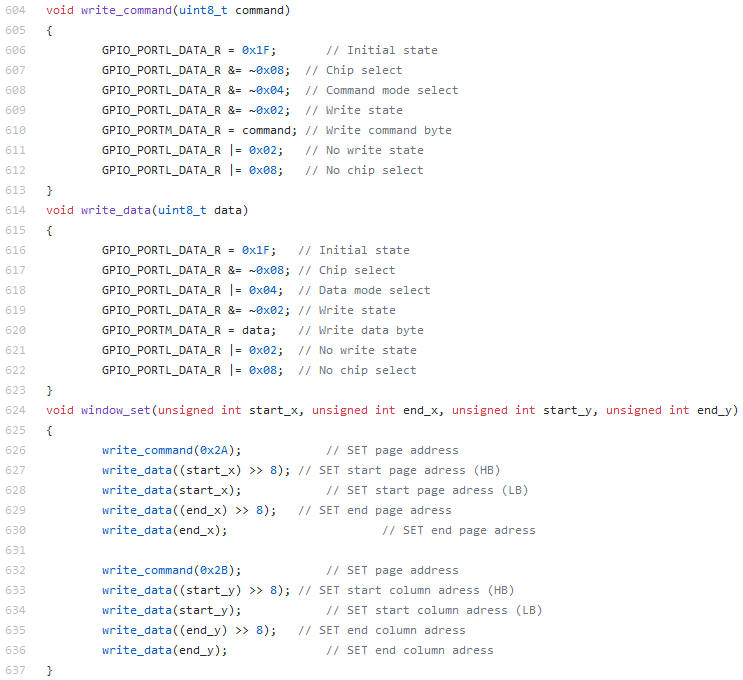




Es wird in ausgewählter Farbe, passend zum Hintergrund, überschrieben.



Mit Hilfe der Funktion „window-set“ wird variabel Einfluss auf den zu beschreibenden Bereich des LC-Display genommen.



# 4. Ergebnisse

Es werden die Ergebnisse des Projektes diskutiert. Rückblickend wird überlegt, welche Herangehensweisen für weitere Projekte sinnvoll wären und welche weniger.

## 4.1 Planung

In der Planungsphase wurden die Tätigkeiten der einzelnen Labortermine abgeschätzt. Es fiel relativ schwer im Vornherein zu planen, da der Umfang bestimmter Aufgabengebiete nicht bekannt war. Effektives Planen ist vor Allem mit Erfahrung möglich. Es ist sinnvoll das strukturierte Planen zu üben, um mit zunehmender Erfahrung neue Projekte besser abschätzen zu können. Zudem hilft eine gute Planung bei der Dokumentation und damit als Leistungsnachweis.

## 4.2 Konzept

Die Konzeptphase war sehr hilfreich, um die Planung zu konkretisieren. Durch die Überlegung, wie einzelne Aufgabenfelder umzusetzen sind, wurde das Projekt insgesamt klar.

Manche Ansätze des Konzeptes mussten im Laufe der Programmierung überdacht werden.

Gründe für eine veränderte Umsetzung im Programm waren das Sparen von Ressourcen durch weniger globale Variablen, damit einhergehend eine bessere Übersichtlichkeit. Es wurde versucht Redundanzen zu vermeiden.

Die Sicherheit der Funktionalität ist durch übersichtlichere Programmierung gestiegen.

Die Messung der Bewegungsrichtung (2.2.1) ist in der Realisierung an die Vorüberlegungen angelehnt.

## 4.3 Programm

Das Ergebnis zur Berechnung der Geschwindigkeit ergibt sich aus dem Wert des globalen Zählers „gcount“, welcher mit jeder Umdrehung der Scheibe in einer Richtung hoch zählt und der periodischen Abfrage durch den Systick-Handler. Durch eine hohe Frequenz der Abfrage, bei einer Periodendauer von 10ms, können nur wenige Umdrehungen gezählt werden. Damit verbunden wird das Rechenergebnis sehr ungenau.

Wir mussten einen Kompromiss finden, zwischen ausreichender Genauigkeit des Rechenergebnisses und genügend hoher Frequenz der Abfrage.

Es wurde die Zeit zwischen den periodischen Abfragen auf 500ms erhöht.

Der Counterwert wird in der längeren Zeit höher, was eine größere Genauigkeit der Berechnung bedeutet. Die Reaktion auf Geschwindigkeitsänderungen nimmt dadurch allerdings ab, was durch breitere Balken sichtbar wird.

# Fazit

Das Projekt hat durch eine freiere Aufgabenstellung wesentlich mehr Eigenständigkeit abverlangt als vergleichbare Praktika. Die Vorgabe von Arbeitsphasen, welche sich an der Planung und Konzeptionierung orientieren, haben geholfen den Einstieg in das Projekt zu finden. Im Laufe der Programmierung konnten die Planungen und Überlegungen im Konzept zur Hilfe genommen werden.

Einige Programmabschnitte mussten durch neue Erkenntnisse verändert werden.

Das Programm erfüllt die Mindestanforderungen, mit der zusätzlichen Funktion, dass durch Berühren des Displays zwischen dem Tageskilometerzähler und einer digitalen Geschwindigkeitsanzeige gewechselt werden kann.

# Tabellenverzeichnis

[Tabelle 1: Excel-Tabelle mit Zeitplan und Arbeitsaufteilung 4](#_Toc28953896)

# Abbildungsverzeichnis

[Abbildung 1: Peripheriebaustein - Rad Sensor und Signale s1, s2 5](#_Toc29219044)

[Abbildung 2: Oszilloskop, Messung des Zeitverhaltens der Sensorsignale 5](#_Toc29219045)

[Abbildung 3: Display - Start Up 7](#_Toc29219046)

[Abbildung 4: Display - Vorwärts 7](#_Toc29219047)

[Abbildung 5: Display - Rückwärts 7](#_Toc29219048)

# Quellenverzeichnis

[1] Projektanleitung „Fahrzeug-Informations-Display“, HAW, Fakultät Technik und Informatik

[2] XPT2046 Touch Screen Controller, Data Sheet, XPTEK, 2007.5

[3] TivaWare Peripheral Driver Library, Users Guide, SPMU298D, März 2013 – Revised July 2016