

Martin Pyka

Simon Kesting

Mikrocontrollersysteme WS19/20

Prof. Dr.-Ing. Karl-Ragmar Riemschneider

Fahrzeug-Inforamtions-Display

Analoger Tachometer, Digitaler Tageskilometerzähler, Bewegungsrichtung

Inhaltsverzeichnis

[Einleitung 2](#_Toc28963024)

[1. Planung 3](#_Toc28963025)

[1.1 Laboraufteilung 3](#_Toc28963026)

[1.2 Zeitplan und Arbeitsaufteilung im Team 4](#_Toc28963027)

[2. Konzept 5](#_Toc28963028)

[2.1 Bewegungsrichtung – und Raddrehzahl – Messung 5](#_Toc28963029)

[2.1.1 Messung der Bewegungsrichtung 5](#_Toc28963030)

[2.1.2 Messung der Raddrehzahl 5](#_Toc28963031)

[2.2 Aufteilung des Grafik-Display 6](#_Toc28963032)

[2.3 Test und Realisierung 7](#_Toc28963033)

[3. Programm 8](#_Toc28963034)

[3.1 Initialisierungen 11](#_Toc28963035)

[3.2 Interrupt-handler 13](#_Toc28963036)

[3.3 LC-Display 15](#_Toc28963037)

[3.3.1 Touch 16](#_Toc28963038)

[3.3.2 Display 17](#_Toc28963039)

[4. Ergebnisse 24](#_Toc28963040)

[4.1 Planung 24](#_Toc28963041)

[4.2 Konzept 24](#_Toc28963042)

[4.3 Programm 24](#_Toc28963043)

[Fazit 24](#_Toc28963044)

[Tabellenverzeichnis 25](#_Toc28963045)

[Abbildungsverzeichnis 25](#_Toc28963046)

[Quellenverzeichnis 25](#_Toc28963047)

# 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

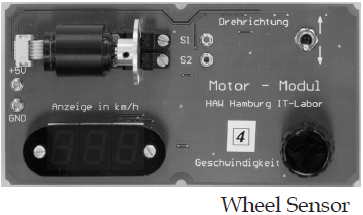
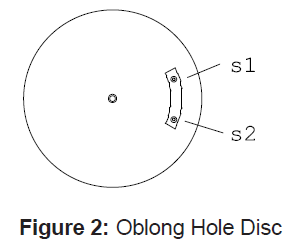


Abbildung : Peripheriebaustein - Rad Sensor und Signale s1, s2

[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.

Messung mit dem Oszilloskop, um Zeitverhalten zu analysieren.

### 2.1.1 Messung der Bewegungsrichtung

Variante 1 (alt):

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.

### 2.1.2 Messung der Raddrehzahl

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.

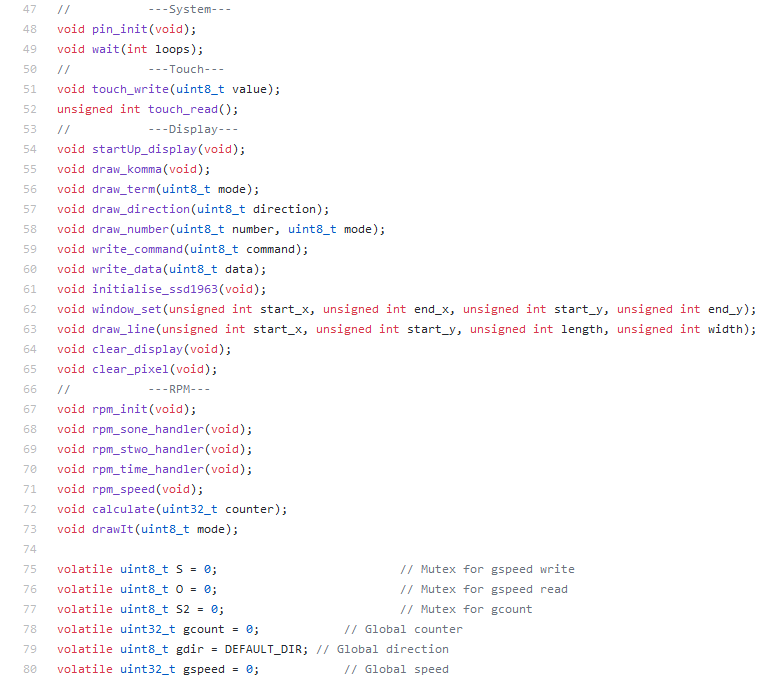
Header-Dateien wurden standardgemäß zu Beginn des Programmes eingefügt. Nachfolgend wurden Makros für verwendete Größen definiert.



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.

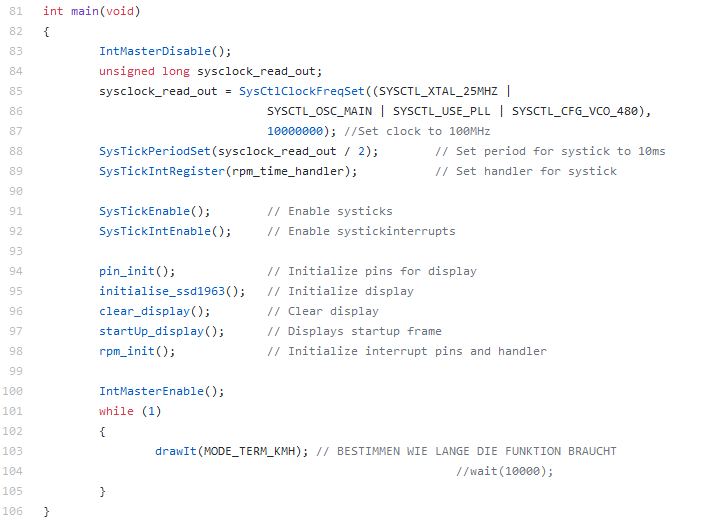


Der globale Zähler „gcount“ zählt die Umdrehungen des Rades in eine bestimmte Richtung, für einen definierten Zeitabschnitt.

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 10ms gesetzt.

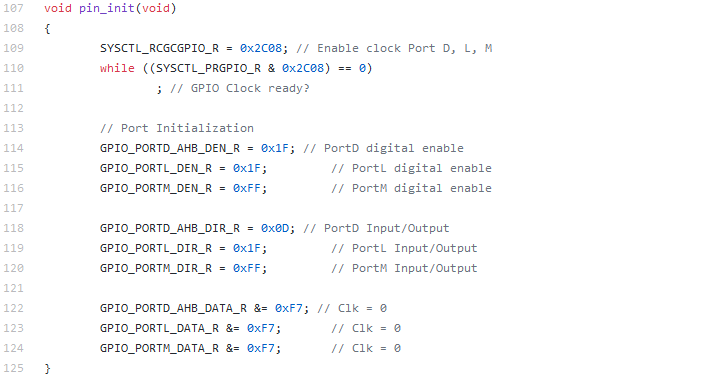
Der Interrupt-handler für den Systemtakt wird in der Interrupt-Vektor-Tabelle (IVT) registriert. Danach wird der Systemtakt-Interrupt freigegeben.



## 3.1 Initialisierungen

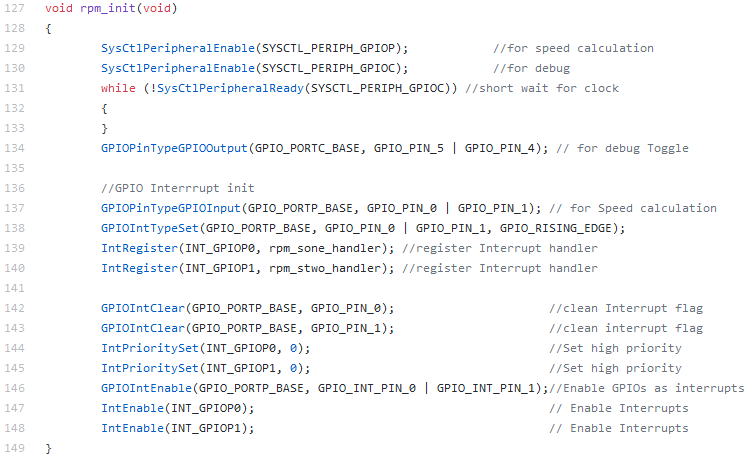
Die Initialisierung der Ports wurde in die Funktion „pin\_init“ ausgelagert.

…



In der Funktion „rpm\_init“ werden die Interrupts initialisiert.

…







## 3.2 Interrupt-handler

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

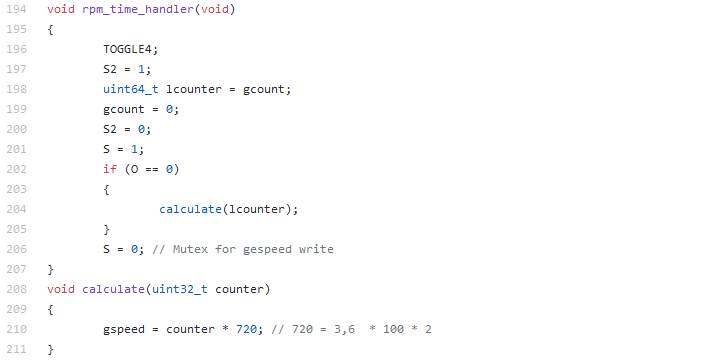
Der Interrupt-handler „rpm\_stwo\_handler“ verarbeitet das Eingangssignal S2 vom Wheel-Sensor-Modul.



Toggle

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 kritischer Zugriffe durch gegenseitigen Ausschluss verhindert.



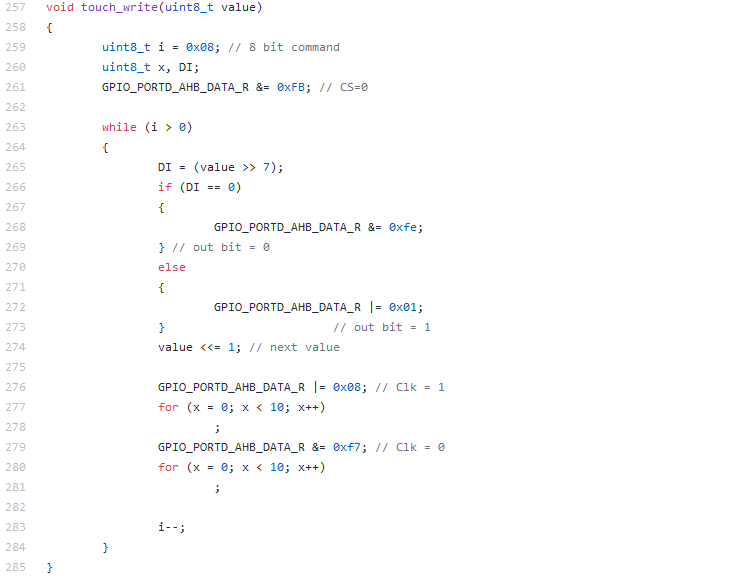
## 3.3 LC-Display

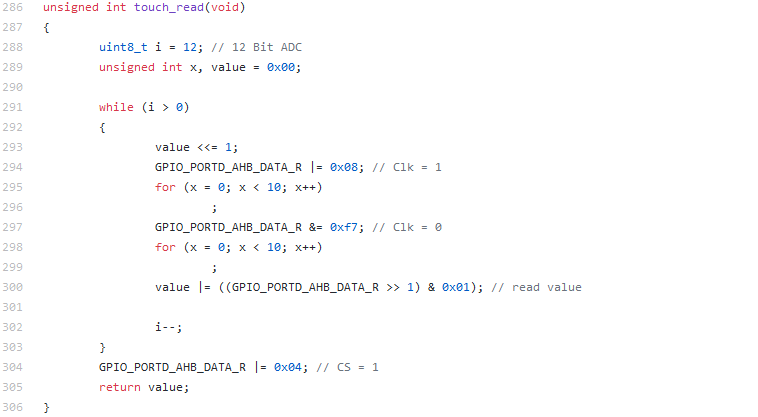
Die Darstellung der Zahlen (0-9) und Buchstaben (V, R) basiert auf Daten, welche über Paint mit Hilfe eines Online-Konverters extrahiert und in header-Dateien abgespeichert wurden.

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

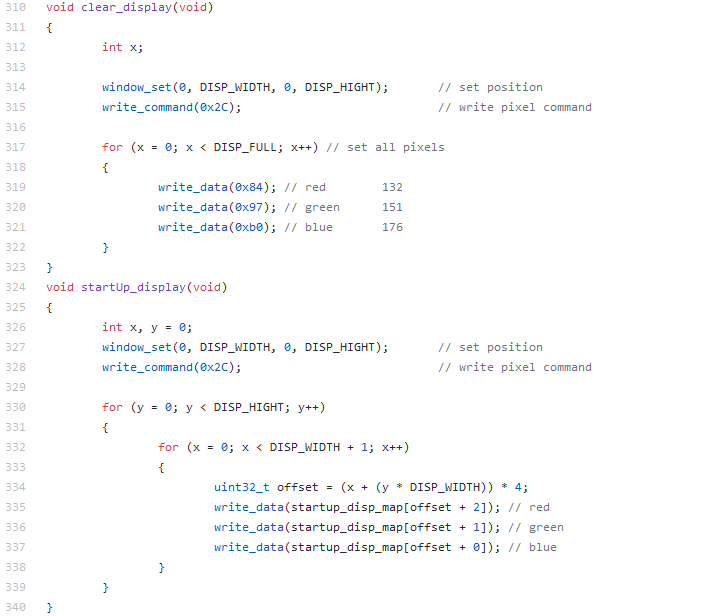


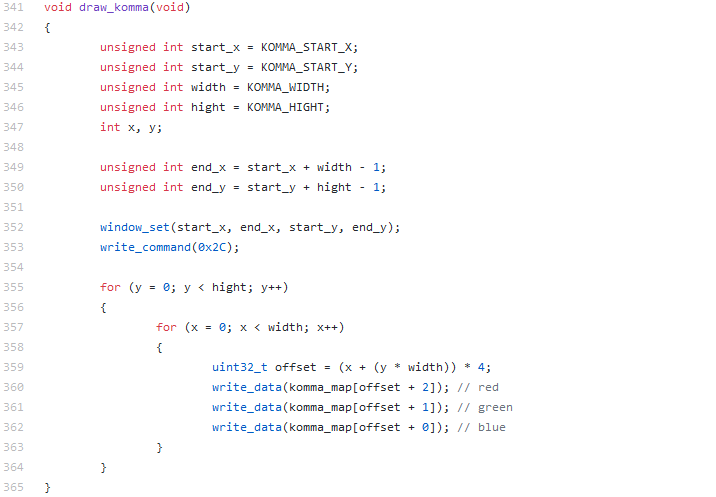
### 3.3.1 Touch



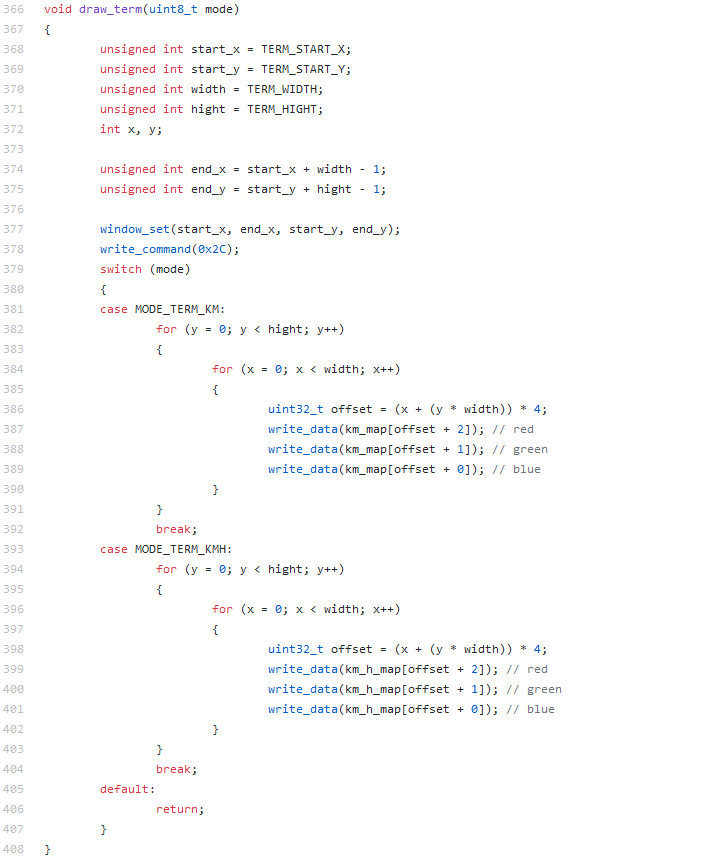


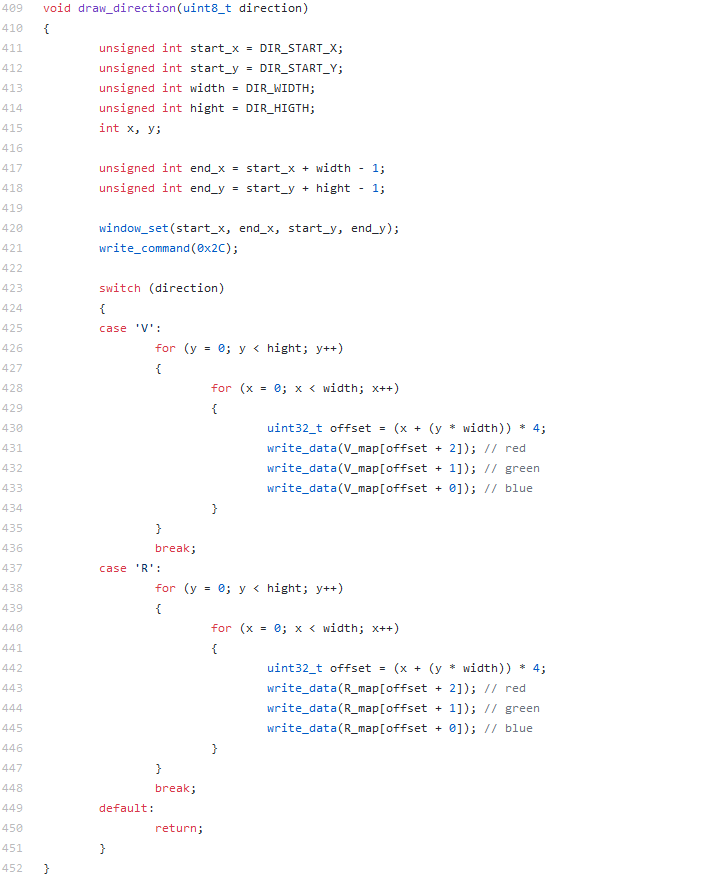
### 3.3.2 Display





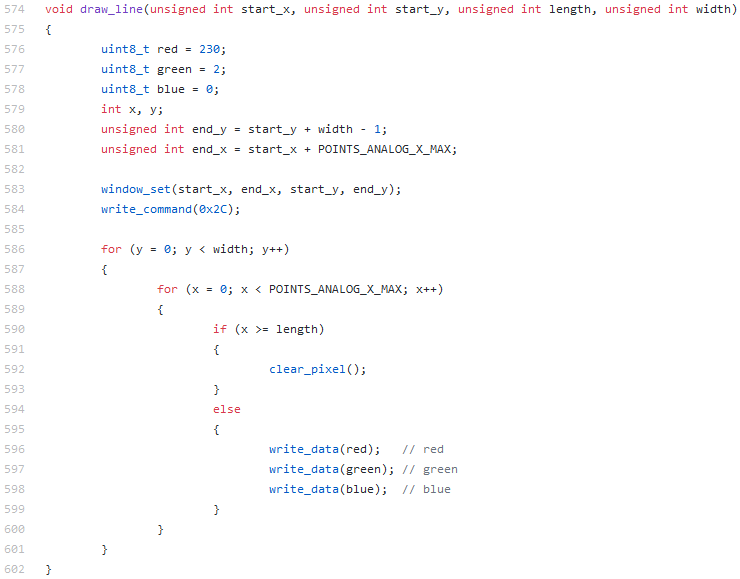
Beschreibung zum Auslesen der maps



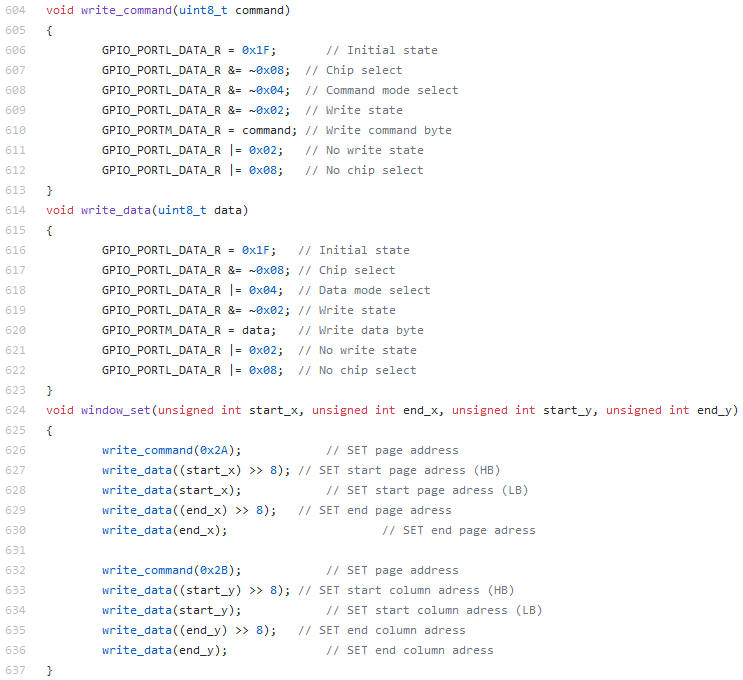








Mit Hilfe der Funktion „window-set“ wird variabel Einfluss auf den zu beschreibenden Bereich des LC-Display genommen.



# 4. Ergebnisse

(Bewertung und Diskussion der Ergebnisse)

(Rückblick -> Änderungen)

## 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.

## 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.

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

## 4.3 Programm

# Fazit

# Tabellenverzeichnis

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

# Abbildungsverzeichnis

[Abbildung 1: Peripheriebaustein - Rad Sensor und Signale s1, s2 5](#_Toc28953869)

[Abbildung 2: Display - Start Up 6](#_Toc28953870)

[Abbildung 3: Display - Vorwärts 6](#_Toc28953871)

[Abbildung 4: Display - Rückwärts 6](#_Toc28953872)

# Quellenverzeichnis

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

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