




TECHNIKERARBEIT

Programmierung eines Mikrocontrollers für die Einbindung von Daten
in ein digitales Armaturenbrett.


In dieser Dokumentation wird die Umsetzung eines digitalen OLED Armaturenbrettes
für den Elektroroller Xiaomi M365 beschrieben.

Rosch Abdolaziz
roschabdolaziz@gmail.com


| | | |
|---|---|---|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20- 22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

INHALTSVERZEICHNIS


| | | |
|------|-------------------------------|----|
| 1 | Abbildungsverzeichnis | 4 |
| 2 | Tabellenverzeichnis | 4 |
| 3 | Hinweis zur Lesbarkeit | 4 |
| 4 | Einleitung | 5 |
| 5 | Vorwort..... | 5 |
| 6 | Problemstellung | 6 |
| 7 | Zielbestimmung..... | 7 |
| 7.1 | Mußkriterien | 7 |
| 7.2 | Wunschkriterien..... | 7 |
| 7.3 | Abgrenzungskriterien..... | 7 |
| 8 | Produkteinsatz | 8 |
| 8.1 | Anwendungsbereiche | 8 |
| 8.2 | Zielgruppen | 8 |
| 9 | Produktübersicht | 9 |
| 9.1 | IST-ZUSTAND..... | 9 |
| 9.2 | SOLL-ZUSTAND | 10 |
| 10 | Projektplanung..... | 11 |
| 10.1 | Beteiligte Personen | 11 |
| 10.2 | Kostenplanung | 11 |
| 10.3 | Hardwarekosten | 11 |
| 10.4 | Zeitplan | 12 |
| 11 | Benutzeroberfläche..... | 13 |
| 11.1 | Hauptmenü..... | 13 |
| 11.2 | Navigationsmenü | 14 |
| 11.3 | Informationsmenu..... | 15 |
| 11.4 | Beschleunigungsmenü | 16 |
| 11.5 | Use Case Menü Steuerung | 17 |
| 12 | Benötigte Hardware | 18 |
| 12.1 | Diode 1N4148 | 18 |
| 12.2 | Werkzeug..... | 18 |
| 12.3 | Widerstand | 18 |

| | | |
|---|---|---|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20- 22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

| | | |
|---------|--|----|
| 12.4 | Steckplatine und Rangierdraht | 18 |
| 12.5 | Multimeter | 18 |
| 12.6 | Lötkolbenset | 18 |
| 12.7 | USB UART Modul | 18 |
| 12.8 | M365 Gehäuse | 19 |
| 12.9 | OLED Bildschirm | 19 |
| 12.10 | Arduino Board | 20 |
| 12.10.1 | Arduino Pro mini | 20 |
| 12.10.2 | Arduino Mega | 21 |
| 13 | Benötigte Software | 22 |
| 13.1 | IDE | 22 |
| 13.2 | Schaltungen & Zeichnungen | 22 |
| 13.3 | Dokumentation | 22 |
| 13.4 | UML Diagramme | 22 |
| 14 | Kommunikation | 23 |
| 14.1 | Kommunikation Module | 23 |
| 14.1.1 | ESC – Electric Scooter Controller | 23 |
| 14.1.2 | BMS – Battery Management System | 23 |
| 14.1.3 | BLE – Bluetooth Modul | 23 |
| 14.1.4 | Schematische Darstellung der Kommunikations Module | 24 |
| 14.2 | M365 Protokoll | 25 |
| 14.2.1 | Paket Header | 25 |
| 14.2.2 | Paket Länge | 25 |
| 14.2.3 | Paket Adresse | 25 |
| 14.2.4 | Paket Kommando | 26 |
| 14.2.5 | Paket Argumente | 26 |
| 14.2.6 | Paket Payload | 26 |
| 14.2.7 | Paket Checksumme | 26 |
| 14.2.8 | Paket Beispiel | 27 |
| 14.3 | Serielle Kommunikationsprotokolle | 27 |
| 14.3.1 | Arduino OLED Bildschirm Kommunikation | 27 |
| 14.3.2 | Arduino M365 Kommunikation | 27 |
| 14.4 | Verkabelung Arduino Mega | 28 |

| | | |
|---|---|---|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20- 22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

| | | |
|---------|--|----|
| 14.5 | Verkabelung Arduino Pro Mini..... | 29 |
| 15 | Programmierung | 30 |
| 15.1 | Verwendete Bibliotheken..... | 30 |
| 15.1.1 | SSD1306AsciiAvrI2C.h | 30 |
| 15.1.2 | System5x7mod.h..... | 30 |
| 15.1.3 | EEPROM.h | 30 |
| 15.2 | Verwendete Methoden | 30 |
| 15.2.1 | void disableRX() | 30 |
| 15.2.2 | void enableRX() | 31 |
| 15.2.3 | void recievedata() | 31 |
| 15.2.4 | void processData()..... | 36 |
| 15.2.5 | void preparePacket(uint8_t option) | 38 |
| 15.2.6 | void preparewirtePacket(uint8_t writeoption) | 40 |
| 15.2.7 | void saveandloadconfig() | 42 |
| 15.2.8 | void displayclear() | 42 |
| 15.2.9 | void displaydata()..... | 43 |
| 15.2.10 | void setup() | 50 |
| 15.2.11 | void loop() | 51 |
| 15.3 | Quellen Programmierung | 52 |
| 16 | Probleme & Lösungen | 53 |
| 16.1 | Zwei Serielle Verbindungen | 53 |
| 16.2 | Pakete senden | 53 |
| 16.3 | Rückwärts fahren zeigt falsche Geschwindigkeit | 54 |
| 16.4 | Frontlicht flackert..... | 54 |
| 16.5 | M365 liefert keine 5V..... | 54 |
| 16.6 | Frontlicht Dekodierung | 55 |
| 16.7 | Bildschirm Navigation..... | 55 |
| 16.8 | Geschwindigkeit Anzeige zu klein | 55 |
| 17 | Lizenz und Veröffentlichung auf Github | 56 |
| 18 | Fazit..... | 57 |
| 19 | Erklärung | 58 |

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

1 ABBILDUNGSVERZEICHNIS


| | |
|--|----|
| Abbildung 1 Ohne OLED Armaturenbrett | 9 |
| Abbildung 2 Mit OLED Armaturenbrett | 10 |
| Abbildung 3 Hauptmenü | 13 |
| Abbildung 4 Navigationsmenü | 14 |
| Abbildung 5 Informationsmenü | 15 |
| Abbildung 6 Beschleunigungsmenü | 16 |
| Abbildung 7 Use Case Menü Steuerung..... | 17 |
| Abbildung 8 Paket Beispiel | 27 |
| Abbildung 9 Verkabelung Mega - fritzing | 28 |
| Abbildung 10 Verkabelung Mega - Original | 28 |
| Abbildung 11 Verkabelung Pro Mini - fritzing..... | 29 |
| Abbildung 12 Verkabelung Pro Mini - Original | 29 |
| Abbildung 13 loop() Methode..... | 51 |

2 TABELLENVERZEICHNIS

| | |
|-------------------------------|----|
| Tabelle 1 Hardwarekosten..... | 11 |
| Tabelle 2 Zeitplan | 12 |
| Tabelle 3 Paket Adressen..... | 25 |
| Tabelle 4 Kommandos..... | 26 |

3 HINWEIS ZUR LESBARKEIT

Variablen die vor allem im Kapitel 15 beschrieben werden, sind innerhalb dieser Dokumentation **Fett** geschrieben.

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

4 EINLEITUNG

Die Projektdokumentation wurde im Rahmen meines Abschlussprojektes zum Staatlich geprüften Techniker für Informationstechnologie erstellt. In dieser Dokumentation werden die wichtigsten Punkte der Projektarbeit erläutert und beschrieben.


5 VORWORT

Die Umstände, die zur Wahl des Themas geführt haben war die Neugier an meinem Elektro Roller (Xiaomi M365). Um genauer zu sein wollte ich wissen wie ich meinen M365 Optimieren kann. Dies führte dazu, dass ich mich immer tiefer in die Materie eingelesen habe. Zu Beginn gab es noch nicht viel Information über den Roller, da der Hersteller selbst nicht will, dass man zu viel an dem Roller verändert.

Beispielsweise ist die Firmware des Rollers nicht offengelegt. Nach einiger Zeit wurde in verschiedensten Online Foren veröffentlicht, wie die genaue Kommunikation zwischen der Bluetooth Schnittstelle und dem Handy funktioniert¹. Folgend wurde Firmware geknackt. In den Sommerferien habe ich dann auf Youtube ein Video gefunden, indem der Herr Darius M. ein Digitales Armaturenbrett für den Roller veröffentlicht hat². Im Anschluss habe ich Herrn M. per Mail geschrieben (Mailverlauf unter Anhänge). Herr M. hat mir geantwortet und mir einige Tipps gegeben wie ich an das Projekt herangehen kann.

¹ Vgl. <https://forum.electricunicycle.org/topic/2686-unraveling-ninebot-one-e-ble-protocol-success/>


² Vgl. <https://www.youtube.com/watch?v=btqhh5pPU9Q&t=48s>

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

6 PROBLEMSTELLUNG

Der Elektro Roller Xiaomi M365 besitzt eine Bluetooth Schnittstelle mit der sich einige Rollereigenschaften anzeigen oder ändern lassen. Der Hersteller selbst bietet dafür die von Ihm zur Verfügung gestellte App, um über die Bluetooth Schnittstelle Eigenschaften des M365 anzuzeigen oder zu ändern. Während einer Fahrt mit dem M365 ist jedoch unmöglich das Handy in der Hand zu halten und gleichzeitig zu fahren, da der Roller selbst keine Handy Halterung anbietet. Weitere Nachteile die bei der Benutzung eines Handys auftreten:

- Handy muss Betriebsbereit sein.
- Handy muss ständig über Bluetooth mit dem Roller verbunden sein.
- Handy verbraucht während der Nutzung Akku.
- Die Synchronisation zwischen der App und dem M365 kann bis zu mehrere Minuten dauern.

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

7 ZIELBESTIMMUNG

7.1 MUßKRITERIEN


- Menüeinstellungen im Armaturenbrett müssen gesteuert werden können.
- Anzeige für Geschwindigkeit.
- Anzeige für Batterie Level.
- Anzeige für Motortemperatur.
- Anzeige für Gefahrene Kilometer seit Start des Rollers.
- Rücklicht muss an- und ausgeschaltet werden können.
- KERS Steuerung.(Kinetic Energy Recovery System)
- Fahrtregler (Tempomat) muss gesteuert werden können.

7.2 WUNSCHKRITERIEN

- Einbau eines Zusatz Akkus.
- Firmware Anpassung für optimale Fahreigenschaften.
- Anbringen eines Sensors, der überprüft ob der Ständerhalter offen ist.
- Programmierung einer Android Anwendung welches mit den Mußkriterien übereinstimmt.
- Herstellen eines Gehäuses mit Hilfe eines 3D Druckers.

7.3 ABGRENZUNGSKRITERIEN

- Keine GPS Funktionen.
- Kein Zusatz Akku für den Mikrocontroller oder anderen Sensoren.

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

8 PRODUKTEINSATZ


Das Produkt dient Personen die sich einen Xiaomi M365 erworben haben oder noch erwerben wollen.

8.1 ANWENDUNGSBEREICHE

Im Straßenverkehr, Privatgelände

8.2 ZIELGRUPPEN

Die aktuelle Gesetzlage ist momentan im Wandel, deshalb ist eine genaue Zielgruppe schwer zu definieren. Aktuell darf jeder der das 14 Lebensjahr erreicht hat den Roller fahren, jedoch muss in Kenntnis genommen werden, dass eine Versicherungspflicht sowie andere Auflagen beachtet werden müssen.

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

9 PRODUKTÜBERSICHT


Das OLED Armaturenbrett soll direkt an den Roller angeschlossen werden können. Damit ist eine direkte Löt- oder Steckverbindung zum Schaltboard gemeint. Dies erspart den Einbau einer zusätzlichen Stromversorgung für den Mikrocontroller. Die Bluetooth Schnittstelle lässt sich direkt physikalisch über das Schaltboard ansteuern. Somit ist der OLED Bildschirm mit dem Mikrokontroller verbunden und Mikrokontroller selbst direkt mit dem Schaltboard vom M365.

9.1 IST-ZUSTAND

Auf dem Bild sieht man, dass vom Hersteller gelieferte Armaturenbrett in schwarz. Im IST-Zustand kann man lediglich 4 LEDs sehen. Die LEDs stellen die aktuelle Akku Kapazität dar.



Abbildung 1 Ohne OLED Armaturenbrett


| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

9.2 SOLL-ZUSTAND

Hier ist deutlich zu erkennen, dass sich im oberen Teil des Armaturenbrettes ein OLED Bildschirm befindet. Der Mikrocontroller befindet sich direkt darunter.



Abbildung 2 Mit OLED Armaturenbrett

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

10 PROJEKTPLANUNG

10.1 BETEILIGTE PERSONEN

Für die ausgeführten Tätigkeiten wurden folgende Personen mit einbezogen:

Herr Ostheimer (MC Lehrer)

Herr Kohler (MC Lehrer)

10.2 KOSTENPLANUNG


Im folgenden Abschnitt werden die entstandenen Kosten erfasst und zusammengerechnet.

10.3 HARDWAREKOSTEN

Die Hardwarekosten beinhaltet sämtliche Kosten die beim Entwicklungsprozess entstanden sind. Die Kosten des M365 wurden hierbei nicht mit eingerechnet.

| Hardware | Preis in € |
|------------------------|---------------|
| Arduino Mega2560 | 14,99 |
| Arduino Pro Mini | 2,49 |
| Lötkolben Set | 10,99 |
| Widerstand 120 Ohm | 0,10 |
| Diode 1N4148 | 0,08 |
| OLED Bildschirm 128x64 | 1,49 |
| Kupfer Kabel | 0,99 |
| USB UART Modul | 1,50 |
| Steckplatine | 1,99 |
| Werkzeug | 9,99 |
| Multimeter | 19,99 |
| 3D Gedrucktes Gehäuse | 33,49 |
| Heißklebepistole | 14,99 |
| Gesamtkosten: | 113,08 |

Tabelle 1 Hardwarekosten


| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

Das Projekt wird in verschiedene Arbeitspakete Aufgeteilt die Arbeitspakete können aus der unteren Tabelle entnommen werden.

10.4 ZEITPLAN

| Arbeitspaket | Fälligkeit | SOLL Stunden | IST Stunden | Differenz |
|-------------------------|------------|-----------------|----------------|-----------|
| Informationsbeschaffung | 15.11.2018 | 30 | 50 | -20 |
| Hardwarebeschaffung | 14.12.2018 | 10 | 12 | -2 |
| Programmierung | 28.02.2019 | 80 | 74 | +6 |
| Dokumentation | 01.04.2019 | 40 | 32 | +8 |

Tabelle 2 Zeitplan

| | | |
|---|---|---|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

11 BENUTZEROBERFLÄCHE

Aktuell wird die Benutzeroberfläche in 4 Menus aufgeteilt.

11.1 HAUPTMENÜ

Das Hauptmenü wird beim Starten des M365 dargestellt. Aktuell wird oben links das Batterie Level in Prozent angezeigt. Oben rechts wird angezeigt, ob das Frontlicht an oder aus ist. Unten links kann die aktuelle Temperatur entnommen werden. Eine Option im Navigationsmenü erlaubt es unten rechts eine Batterie Warnung aufblinken zu lassen.

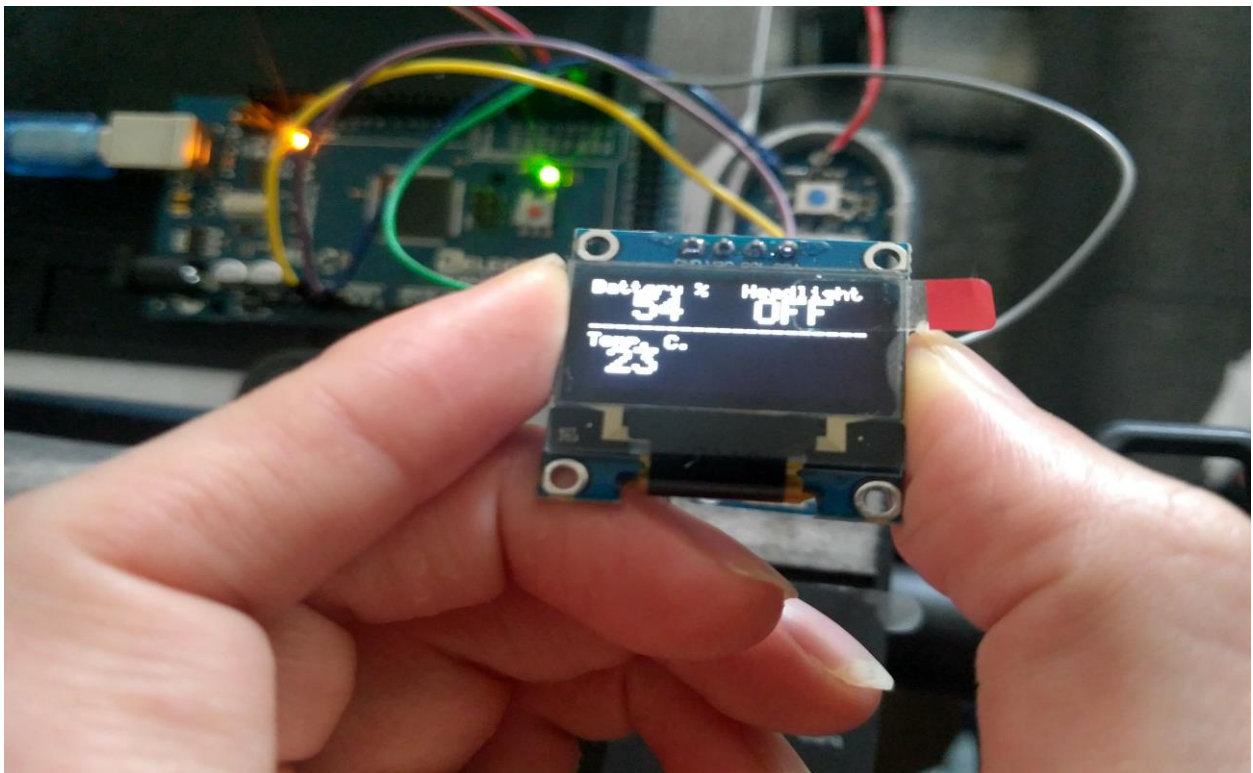



Abbildung 3 Hauptmenü

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

11.2 NAVIGATIONS MENÜ

Durch fast vollständiges betätigen des Brems- und Beschleunigungshebels wird das Navigationsmenü geöffnet. Der Pfeil bestimmt die aktuelle Position im Menü, durch betätigen des Bremseschalters wandert der Pfeil auf die nächste Position im Menü. Wird der Beschleunigungsschalter betätigt kann die aktuelle Option ausgewählt werden, je nachdem welche Option ausgewählt wurde, werden unterschiedliche Tasks ausgeführt.

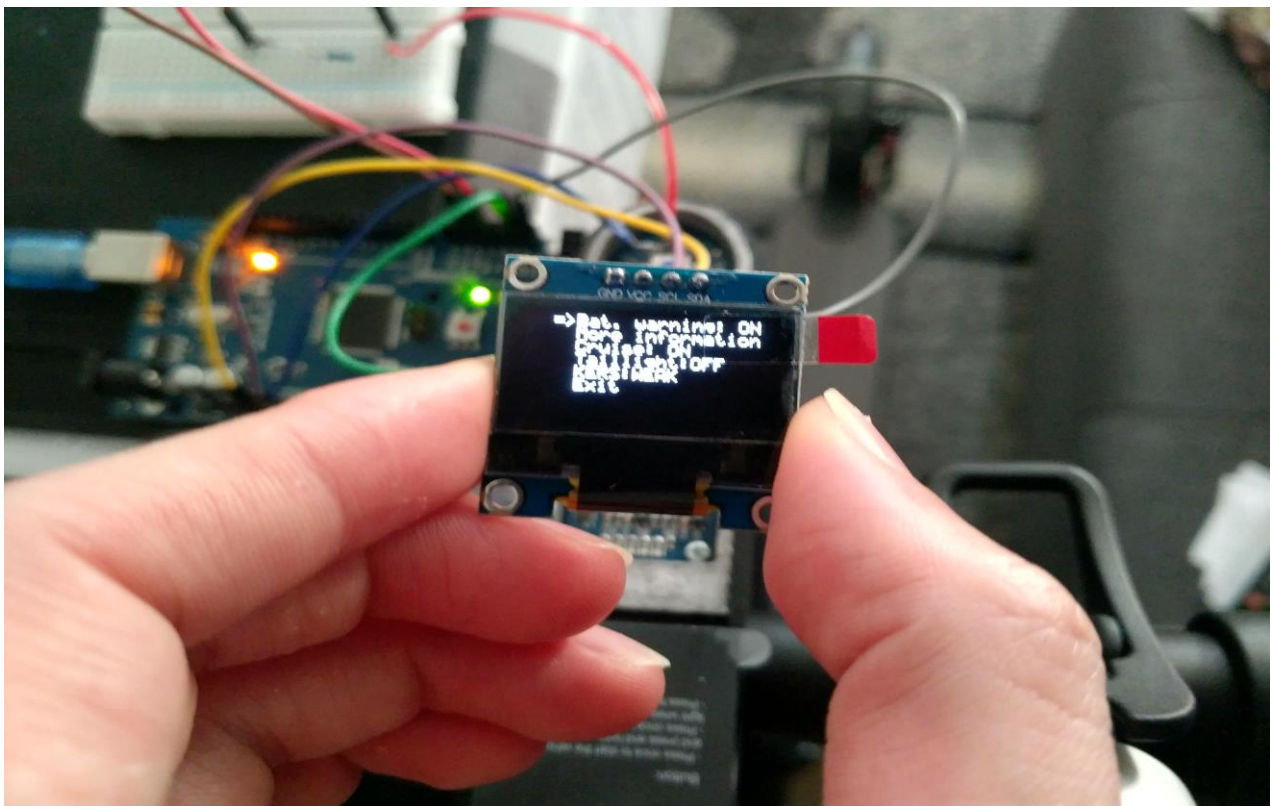



Abbildung 4 Navigationsmenü

| | | |
|---|---|---|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

11.3 INFORMATIONSMENU

Wenn im Navigationsmenü die Option „More Information“ ausgewählt wird öffnet sich das Informationsmenu. Aktuell kann aus dem Informationsmenü in der oberen Hälfte der Tachostand entnommen werden. In der unteren Hälfte wird der Tachostand zur aktuellen Fahrt in Kilometern angezeigt. Durch einmaliges betätigen des Bremshebels wird das Menü verlassen und das Navigationsmenü wieder geöffnet.

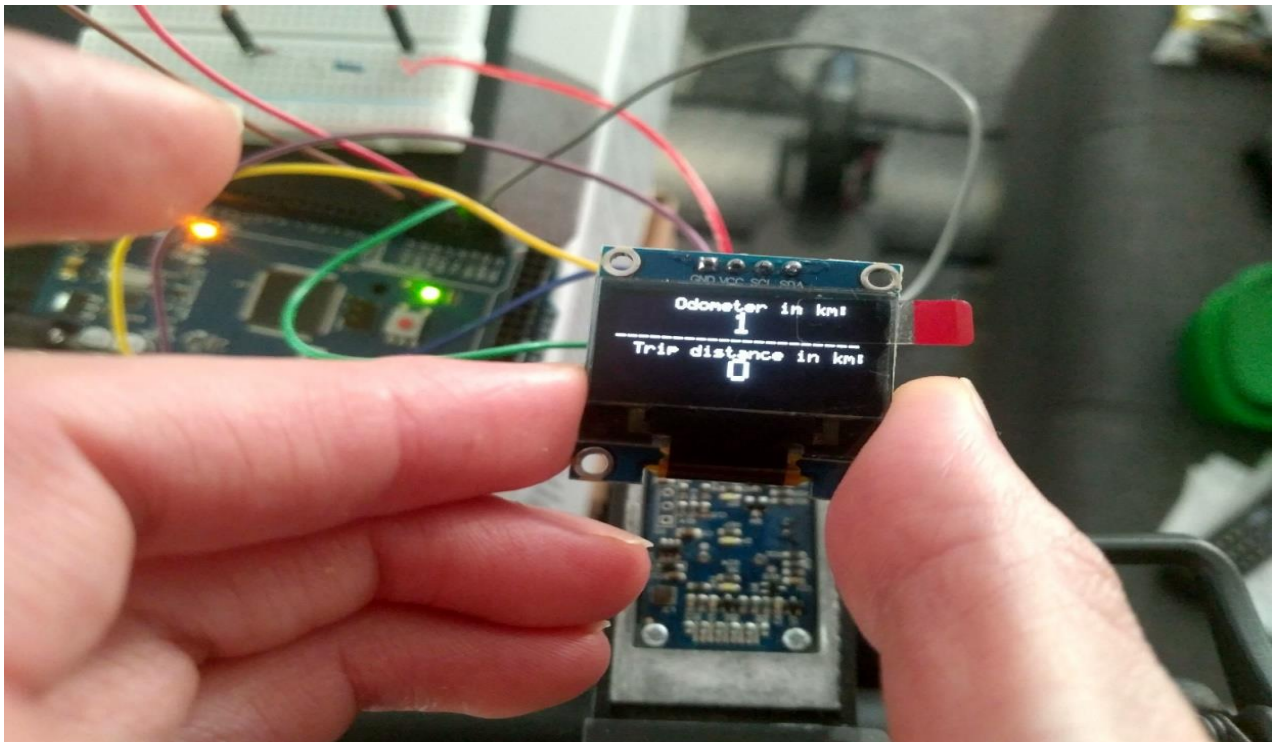



Abbildung 5 Informationsmenü

| | | |
|---|---|---|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

11.4 BESCHLEUNIGUNGSMENÜ

Das Beschleunigungsmenü wird aktuell bei einer Geschwindigkeit von größer als 2 Km/h angezeigt. Hierbei spielt es keine Rolle welches Menü gerade angezeigt wird. Aktuell kann aus dem Beschleunigungsmenü die Geschwindigkeit in Km/h und der Akkustand in Prozent entnommen werden.

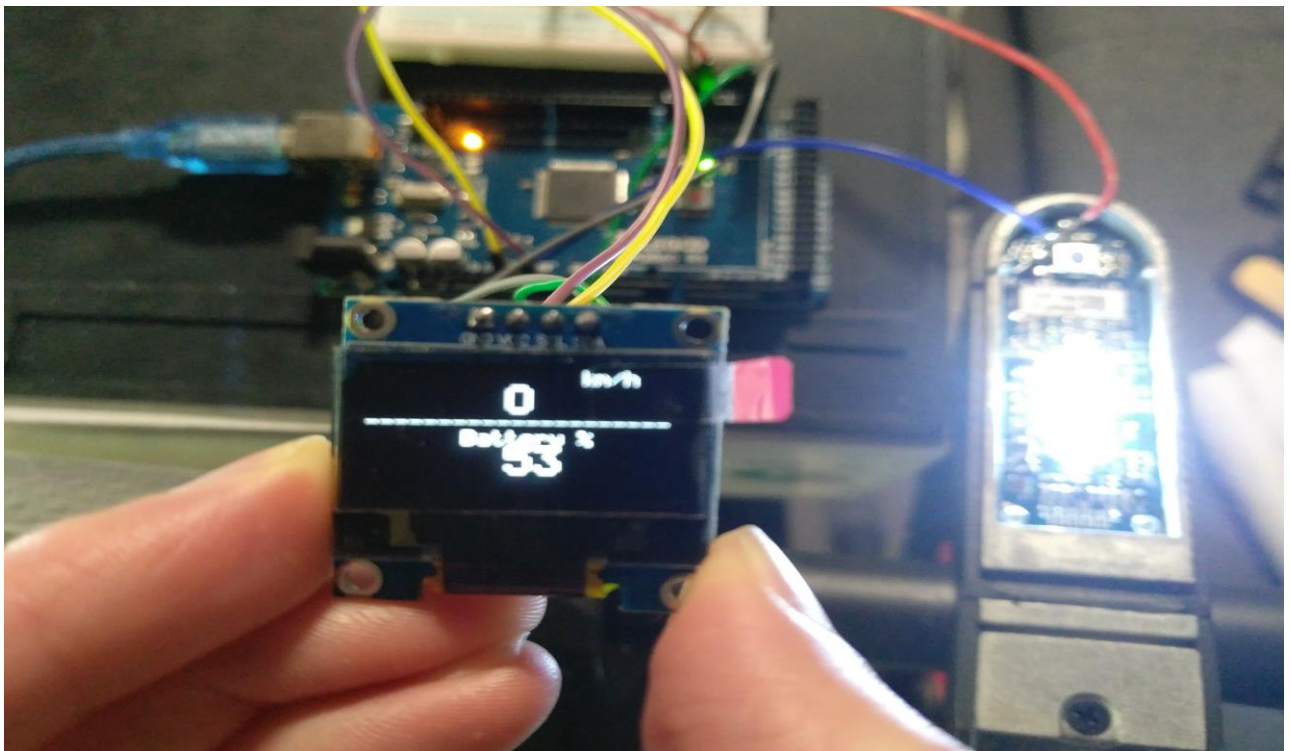


Abbildung 6 Beschleunigungsmenü

11.5 USE CASE MENÜ STEUERUNG

UseCase Diagram0

2019/04/10 astah* Evaluation

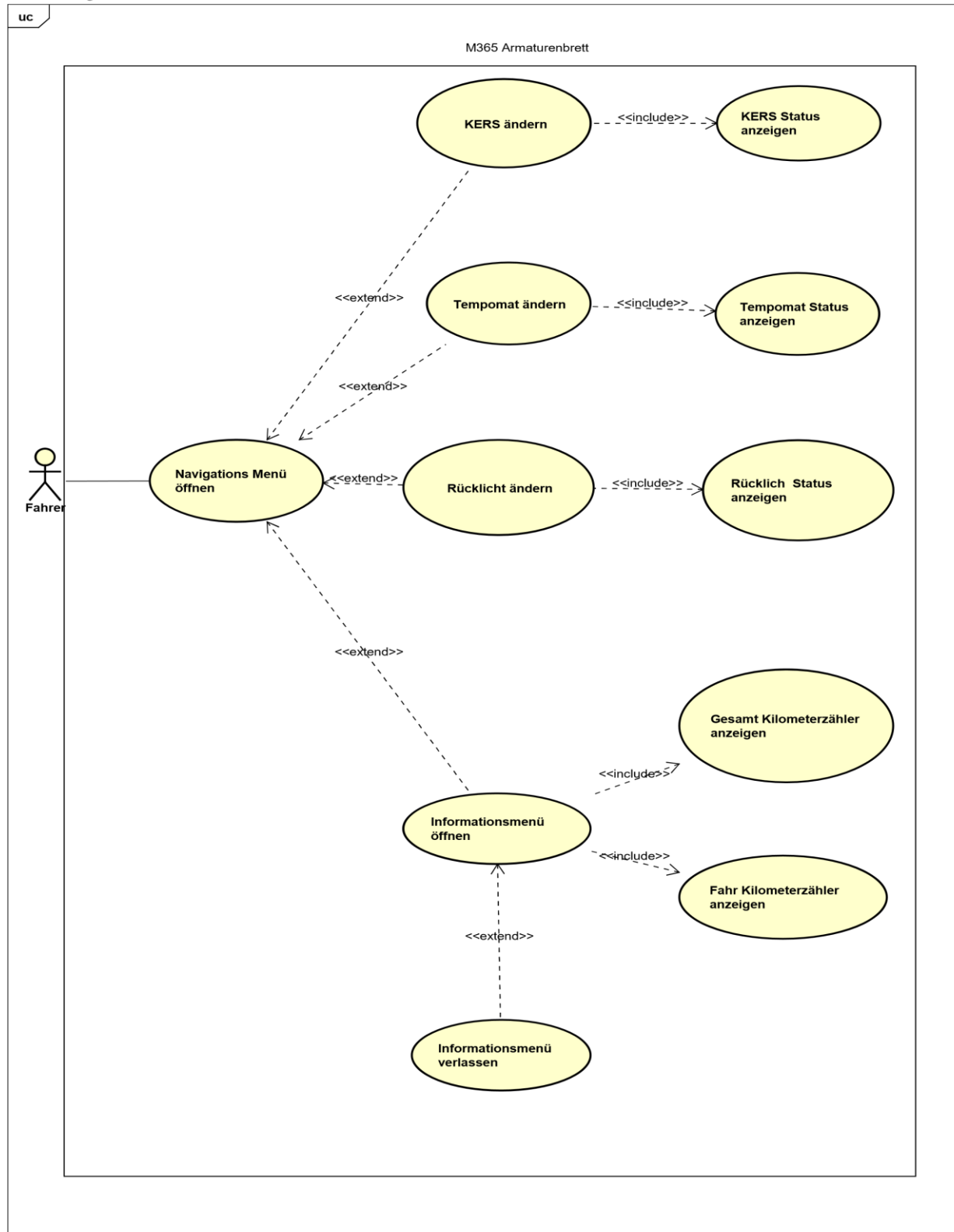



Abbildung 7 Use Case Menü Steuerung

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

12 BENÖTIGTE HARDWARE

In diesem Kapitel wird die benötigte Hardware um das Projekt zu realisieren erläutert.

12.1 DIODE 1N4148

Wird benötigt für die UART Kommunikation mit dem Arduino. Kann theoretisch mit jeder anderen Diode mit den gleichen Eigenschaften ausgetauscht werden.

12.2 WERKZEUG

Zu dem Werkzeug gehört eine Abisolierzange sowie ein Seitenschneider der benötigt wird um das Kabel zu bearbeiten.

12.3 WIDERSTAND

Es wird ein Widerstand von 110 Ohm benötigt.

12.4 STECKPLATINE UND RANGIERDRAHT

Vereinfacht den Entwicklungsprozess, da nicht ständig gelötet werden muss.

12.5 MULTIMETER


Zur Messung der Spannung am M365. Der Pin der den Arduino mit Spannung beliefert muss 5V liefern.

12.6 LÖTKOLBENSET

Hierzu gehört der Lötkolben, Lötzinn und eine Löthalterung.

12.7 USB UART MODUL

Der Arduino Pro Mini besitzt kein USB Interface um das Programm schlussendlich drauf zu laden. Das USB UART Modul kann an den Arduino Pro Mini Pins angeschlossen werden und per USB an den Computer um das Programm einzuspielen.

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

12.8 M365 GEHÄUSE


Der Arduino Pro Mini sowie der OLED Bildschirm werden schlussendlich am Gehäuse mithilfe einer Heißklebepistole befestigt. Das Gehäuse wurde mit einem 3D Drucker erstellt. Hierfür wurde die Firma 3Fraktur beauftragt. Die Benötigten 3D Dateien für das Gehäuse können in den Anhängen betrachtet werden.

12.9 OLED BILDSCHIRM

Es wurde sich für ein OLED Bildschirm entschieden, da dieser im Vergleich zu einen LCD folgende Vorteile bietet:

1. Verbesserte Bild Qualität in Bezug auf besserem Kontrast, Helligkeit, und dem Betrachtungswinkel.
2. Geringerer Stromverbrauch.
3. Besserer Haltbarkeit.

Die Diagonale des Displays beträgt 2,44 cm und passt genau in das M365 Gehäuse.

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

12.10 ARDUINO BOARD

In diesem Abschnitt werden die Entscheidungskriterien für die verwendeten Arduino Boards erläutert.

12.10.1 Arduino Pro mini³


Es muss beachtet werden, dass es verschiedenen Arduino Pro Mini Modelle gibt. Diese unterscheiden sich am verwendeten Mikrokontroller als auch ob sich direkt ein USB-Interface auf dem Arduino befindet. Entscheidungskriterien für den Arduino Pro Mini:

1. Der Platzmangel am M365 erlaubt es nicht einen großen Mikrokontroller zu nutzen.
2. Der M365 liefert 5V an einem Pin, dies erlaubt es die 16Mhz Variante des Mikrokontrollers zu nutzen.
3. Benötigt wird eine UART und I2C Schnittstelle für die Kommunikation mit dem Bus sowie dem Bildschirm.

12.10.1.1 Technische Spezifikation

| | |
|---------------------------|---|
| Microcontroller | ATmega328P |
| Circuit Operating Voltage | 3.3V or 5V (depending on model) |
| UART | 1 |
| SPI | 1 |
| I2C | 1 |
| Flash Memory | 32KB of which 2 KB used by bootloader * |
| SRAM | 2 KB * |
| EEPROM | 1 KB * |
| Clock Speed | 8 MHz (3.3V versions) or 16 MHz (5V versions) |

³ Vgl. <https://store.arduino.cc/arduino-pro-mini>

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

12.10.2 Arduino Mega⁴

Der Arduino Mega wurde schlussendlich für den Entwicklungsprozess genutzt. Es muss beim Entwicklungsprozess darauf geachtet werden, dass der Speicher vom Arduino Pro Mini und dem Arduino Mega sich stark unterscheiden.


Entscheidungskriterien für den Arduino Mega:

1. Arduino Mega besitzt vier Physikalische Serial Pins welche UART unterstützen. Zwei werden mindesten für die Entwicklung benötigt.
2. Programmierung wird erleichtert, da der Pro Mini kein USB Interface besitzt.
3. Mit der Verwendung einer Steckplatine wird kein ständiges Löten benötigt.
4. Stromversorgung wird über USB geregelt.

12.10.2.1 Technische Spezifikation

| | |
|-------------------|---|
| Microcontroller | ATmega2560 |
| Operating Voltage | 5V |
| UART | 4 |
| SPI | 1 |
| IC2 | 2 |
| Flash Memory | 256 KB of which 8 KB used by bootloader |
| SRAM | 8 KB |
| EEPROM | 4 KB |
| Clock Speed | 16 MHz |

⁴ Vgl. <https://www.arduino.cc/reference/en/language/functions/communication/serial/>

| | | |
|--|---|--|
|  it.schule stuttgart | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

13 BENÖTIGTE SOFTWARE

13.1 IDE

Als IDE wurde die offizielle Arduino IDE verwendet.

13.2 SCHALTUNGEN & ZEICHNUNGEN

Für die Darstellungen der Schaltungen wurde Microsoft Visio sowie ⁵fritzing genutzt.


13.3 DOKUMENTATION

Die Dokumentation wurde mit Microsoft Word verfasst.

13.4 UML DIAGRAMME

UML Diagramme wurden mit Astah Professional erzeugt.

⁵ Vgl. <http://fritzing.org/projects/>

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

14 KOMMUNIKATION

In diesem Kapitel werden die verschieden genutzten Kommunikationsmodule sowie die Kommunikationsprotokolle erläutert.

14.1 KOMMUNIKATION MODULE⁶

Die Kommunikation des M365 erfolgt über drei Hauptmodule.

14.1.1 ESC – Electric Scooter Controller

An dem ESC sind alle Sensoren des M365 angeschlossen. Der ESC enthält Register in welchem sich Daten über Rücklicht, Tempomat... befinden.


14.1.2 BMS – Battery Management System

Das BMS Modul steht in ständiger Kommunikation mit dem ESC. Das BMS enthält Register wo sich Daten zur Batterie befinden.

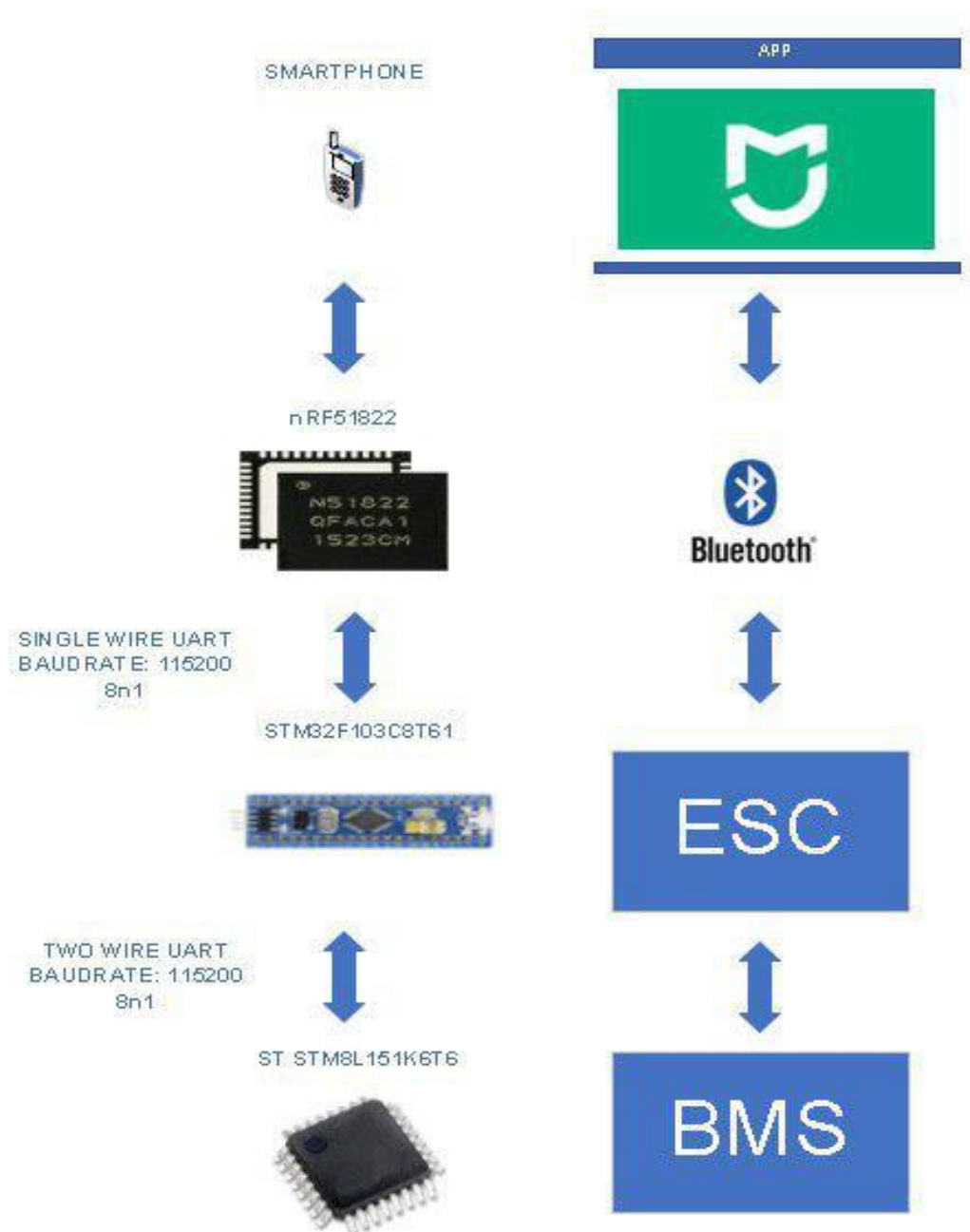
14.1.3 BLE – Bluetooth Modul


Das Bluetooth Modul steht ebenfalls in ständiger Kommunikation mit dem ESC. Wenn ein Handy verbunden ist, werden benötigte Pakete vom BLE Modul an den ESC weitergeleitet.

⁶ Vgl. <https://github.com/etransport/ninebot-docs/wiki/protocol>

| | | |
|---|---|---|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

14.1.4 Schematische Darstellung der Kommunikations Module



| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

14.2 M365 PROTOKOLL

⁷Die Hauptmodule versenden Pakete für die Kommunikation. Das Protokoll welches vom M365 für die Kommunikation von den Hauptmodulen ESC, BMS, BLE und Arduino verwendet wird ist im folgendem Kapitel näher erläutert.

14.2.1 Paket Header

Die ersten zwei Bytes eines Pakets haben immer die Header 0x55 und 0xAA.

14.2.2 Paket Länge

Das dritte Byte eines Pakets bestimmt immer die Länge eines Paketes. Die Länge ergibt sich durch das addieren der Bytes des Kommandos, Arguments sowie der Payload. Die Payload kann dabei mehr als ein Byte groß sein.


14.2.3 Paket Adresse

Die Adresse gibt an von wo das Paket herkommt oder an wen es gerichtet ist. Die wichtigsten Adressen können aus der folgenden Tabelle entnommen werden:

| Komponente | HEX Wert |
|----------------|----------|
| Request to ESC | 0x20 |
| Request to BLE | 0x21 |
| Request to BMS | 0x22 |
| Reply from ESC | 0x23 |
| Reply from BLE | 0x24 |
| Reply from BMS | 0x25 |

Tabelle 3 Paket Adressen

⁷ Vgl. <https://github.com/etransport/ninebot-docs/wiki/protocol>

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

14.2.4 Paket Kommando

Kommandos haben verschieden Funktionen. Beispielsweise kann mit dem Kommando 0x01 Register Werte vom ESC ausgelesen werden. Ein weiteres Kommando 0x03 erlaubt es Register Werte zu verändern.

Die für die Projektarbeit wesentlichen Kommandos können aus der folgenden Tabelle entnommen werden.

| Kommando | Beschreibung |
|----------|---|
| 0x03 | Schreibe und ändere Register Wert |
| 0x61 | Lese Register Werte ab Register X (X kann als Parameter angegeben werden) |
| 0x64 | Update Sensor Werte am ESC. Antwort wird für dieses Paket vom BLE Modul erwartet. |
| 0x65 | Update Sensor Werte am ESC. Antwort wird für dieses Paket nicht erwartet. |

Tabelle 4 Kommandos

14.2.5 Paket Argumente


Die Argumente sind jeweils immer vom Kommando abhängig. Ein Argument kann beispielsweise der Index eines Registers sein.

14.2.6 Paket Payload

Die Payload kann mit zusätzlichen Information zum aktuellen Paket befüllt sein. Die Payload kann aber auch leer sein.

14.2.7 Paket Checksumme

Um die Integrität der Nachricht zu gewährleisten hat jedes Paket Ende eine Checksumme aus 2 Bytes. Die Checksumme wird durch das addieren aller Bytes eines Paketes mit Ausnahme der Header berechnet. Die Summe der Addition muss noch mit einem XOR und 0xFFFF logisch verknüpft werden.

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

14.2.8 Paket Beispiel

Das folgende Paket deaktiviert das Rücklicht beim M365.

| Header1 | Header2 | Länge | Adresse | Kommando | Argument | Payload | Payload | Checksumme | Checksumme |
|---------|---------|-------|---------|----------|----------|---------|---------|------------|------------|
| 0x55 | 0xAA | 0x04 | 0x20 | 0x03 | 0x7D | 0x00 | 0x00 | 0x5B | 0xFF |

Abbildung 8 Paket Beispiel

Weitere Paket Strukturen können aus dem Programmcode oder dem Anhang ProtokollKommunikationv3 entnommen werden.

14.3 SERIELLE KOMMUNIKATIONSPROTOKOLLE

14.3.1 Arduino OLED Bildschirm Kommunikation

Für die Kommunikation vom Arduino zum OLED Bildschirm wird IC2 genutzt.

14.3.2 Arduino M365 Kommunikation⁸

Der M365 kommuniziert halb Duplex über ein Kabel mit dem Arduino. Das Protokoll welches verwendet wird ist UART mit einer Baudrate von 115200.

⁸ Vgl. <https://github.com/etransport/ninebot-docs/wiki/protocol>

14.4 VERKABELUNG ARDUINO MEGA

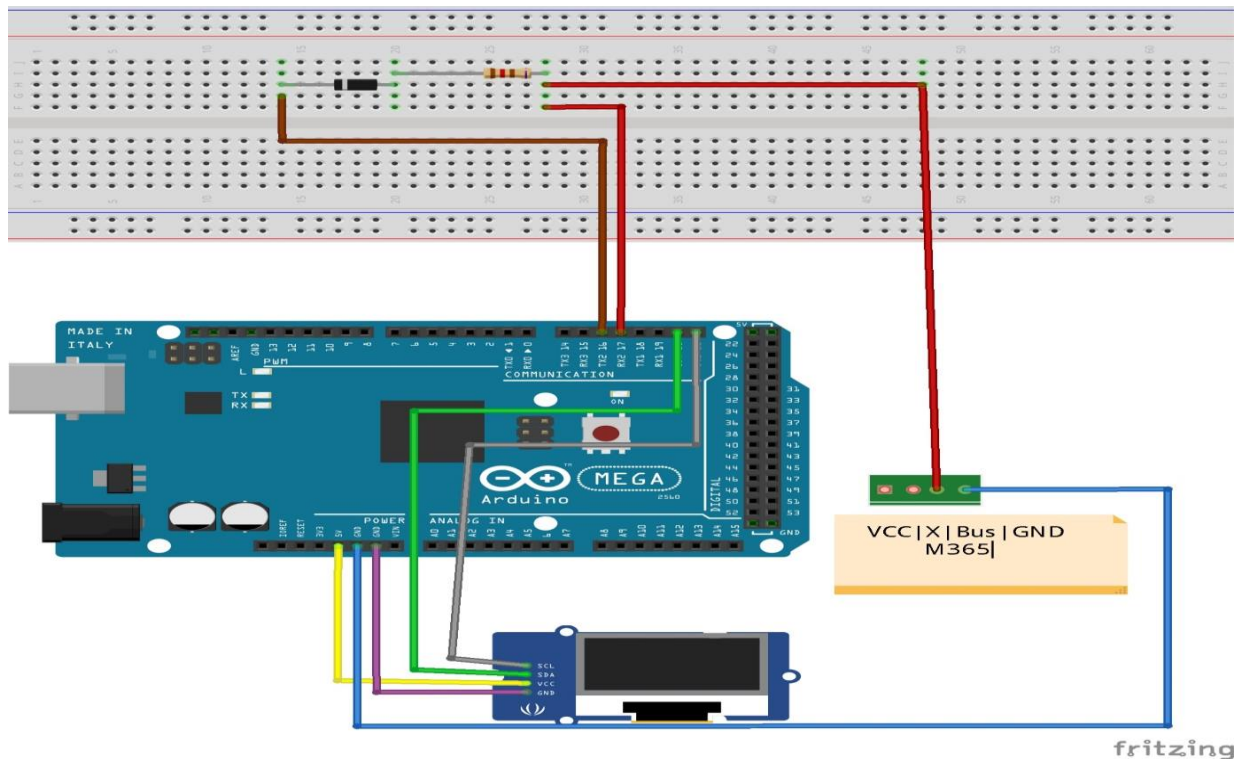


Abbildung 9 Verkabelung Mega - fritzing

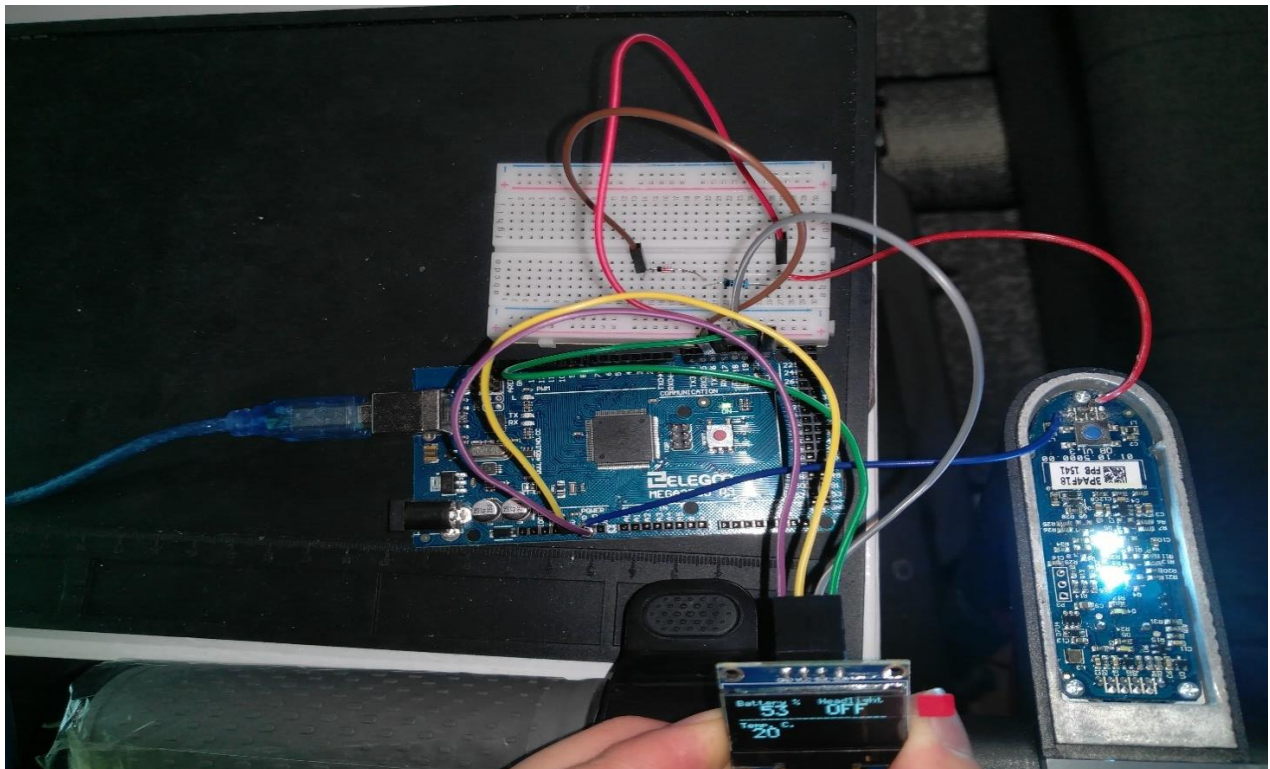


Abbildung 10 Verkabelung Mega - Original

14.5 VERKABELUNG ARDUINO PRO MINI

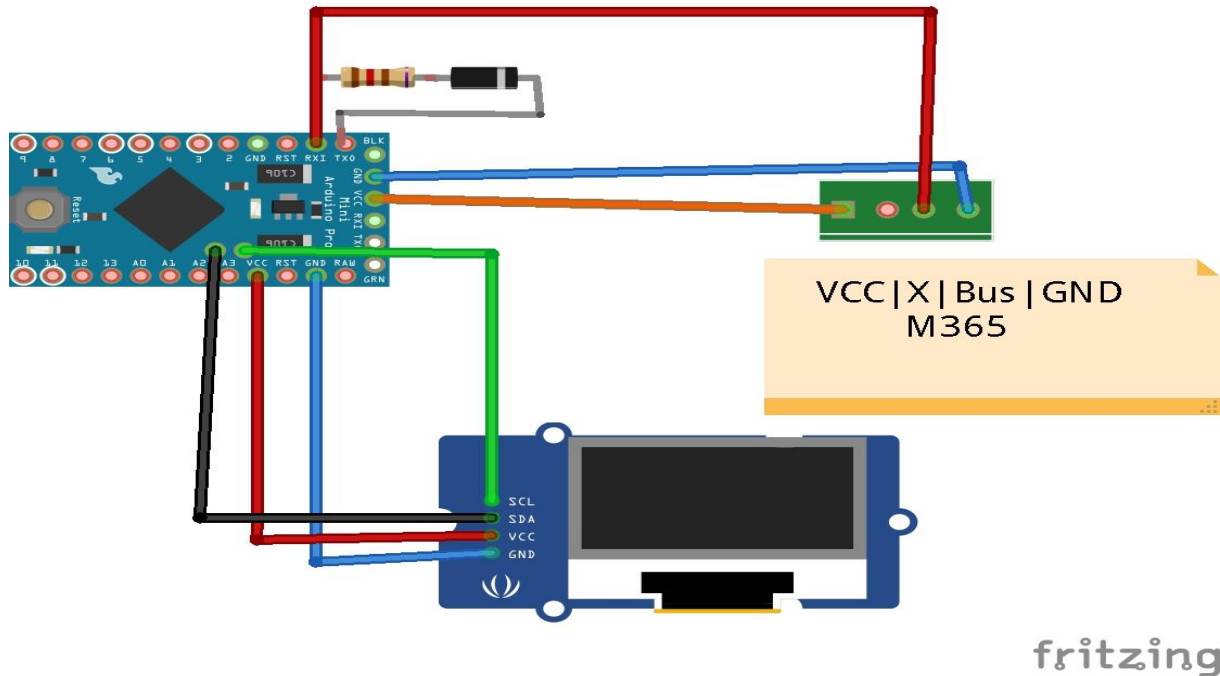


Abbildung 11 Verkabelung Pro Mini - fritzing

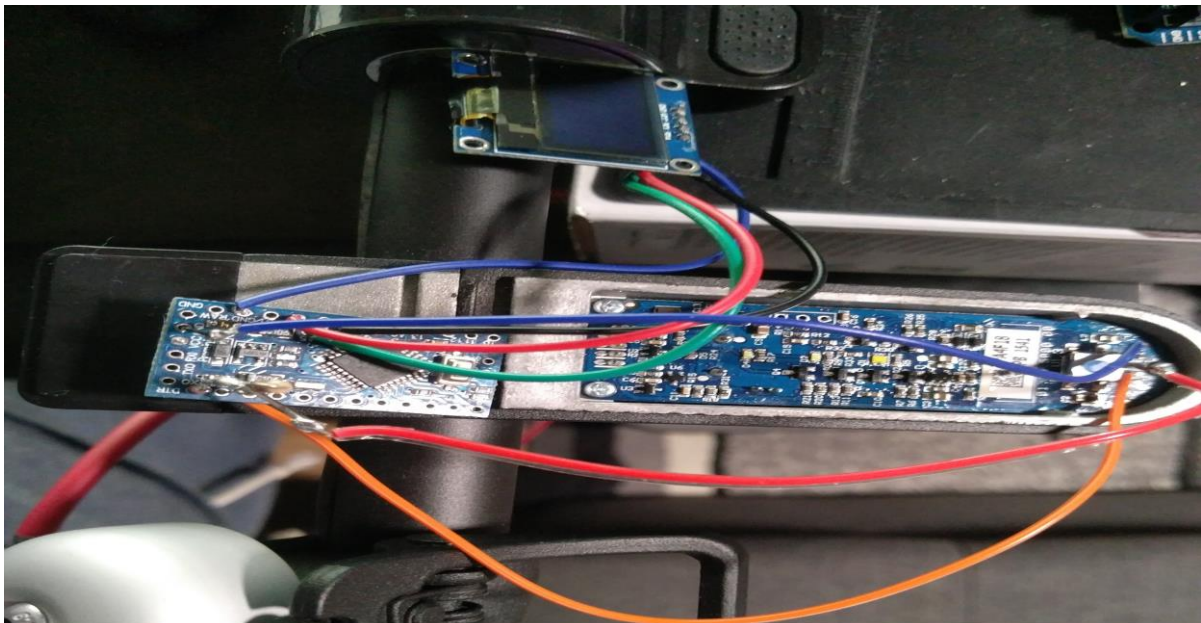



Abbildung 12 Verkabelung Pro Mini - Original

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

15 PROGRAMMIERUNG

In dem folgenden Abschnitt werden die Programm Aspekte erläutert.

15.1 VERWENDETE BIBLIOTHEKEN

15.1.1 SSD1306AsciiAvirl2C.h

Enthält Methoden um die Darstellung am OLED Bildschirm zu realisieren.

15.1.2 System5x7mod.h

Enthält die Schriftart und die Benötigten Zeichen für die Bildschirmanzeige.

15.1.3 EEPROM.h

Enthält die Methoden um Daten persistent in die EEPROM des Arduinos zu schreiben.

15.2 VERWENDETE METHODEN


15.2.1 void disableRX()

Die Methode wird innerhalb der recievedata() Methode aufgerufen. Durch das Löschen eines Bits im UCSR0B Registers kann der RX Pin im Arduino ausgeschaltet werden dies verhindert, dass selbst gesendet Paket dekodiert werden.

```

1. void disableRX()
2. {
3.     UCSR0B |= (1 << RXEN0);
4. }

```

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

15.2.2 void enableRX()

Nachdem erfolgreichen Senden eines Pakets muss der RX Pin im Arduino wieder aktiviert werden. Das vorher gelöschte Bit muss im UCSR0B Register wieder gesetzt werden.

```

1. void enableRX()
2. {
3.   UCSR0B &= ~(1 << RXEN0);
4. }

```

15.2.3 void recievedata()

Die Methode wird als erstes in der loop() Methode aufgerufen. Die Methode liest aus dem Arduino Serial Buffer das erste Byte, wenn Daten im Serial Buffer verfügbar sind. Der Inhalt des Bytes wird danach in die Variable **recievedbyte** gespeichert.

```

1. void recievedata()
2. {
3.   uint8_t recievedbyte;
4.
5.   while (Serial2.available() > 0)
6.   {
7.     recievedbyte = Serial2.read();


```

Die **recievedbyte** Variable wird im folgendem Switch Case in der Verzweigung null auf das erste Header Byte welches 0x55 sein muss geprüft. Wenn die Prüfung erfolgreich ist wird die Globale Variable **bufferreceiverstate** auf eins gesetzt. Nach einem erneuten Aufruf der recievedata() Methode in der loop() Methode wird daraufhin die erste Verzweigung ausgeführt. Die Verzweigung eins prüft ob das nächste Empfangene Byte dem Header 0xAA entspricht.

```

1. switch (bufferreceiverstate)
2. {
3.   case 0:
4.     if (recievedbyte == HEADER55) // = 0x55
5.     {
6.       bufferreceiverstate = 1;
7.     }
8.     break;
9.   case 1:
10.    if (recievedbyte == HEADERAA) // = 0xAA
11.    {
12.      bufferreceiverstate = 2;


```


| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

In der Verzweigung zwei wird das empfangene Byte auf die Paketlänge überprüft. Die Paketlänge darf hierbei nicht die Größe des Empfangs Buffers überschreiten. Die Länge des Pakets entscheidend schlussendlich wie lange das Switch Case die Empfangsroutine durchläuft, bevor das nächste Paket entgegengenommen wird. Hierfür wird die aktuelle Paketlänge in eine Globale Struct Member Variable **recieverbuffer.length** gespeichert. Die Variable muss immer +1 gerechnet werden, da die Paketlänge erst ab dem zweiten Byte zu zählen gilt. Des Weiteren beginnt ab der aktuellen Verzweigung die Checksummen Berechnung. Für die Checksummen Berechnung werden das Paketlängen Byte und alle folgenden Bytes die zum Paket gehören in der Globalen Variable **checksumcalculation** zusammenaddiert.

```

1.  case 2:
2.      if (recievedbyte > 35) //possibility that length is >35 which will overflow the buffer
3.      {
4.          bufferreceiverstate = 0;
5.      }
6.      bufferreadindex = 0;
7.      receiverbuffer.length = recievedbyte + 1; //packet length counts from second byte
8.      checksumcalculation = recievedbyte;
9.      bufferreceiverstate = 3;
10. break;
```

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

In der dritten Verzweigung werden die restlichen Bytes in dem Globalen Struct Array **receiverbuffer.buffer[bufferreadindex]** gespeichert. Die Globale Variable **bufferreadindex** wird immer in der Verzweigung zwei auf null gesetzt und in der Verzweigung drei um eins hochgezählt. Wie in der Verzweigung zwei müssen hier auch alle folgenden Bytes für die Checksumme zur **checksumcalculation** Variable hinzuaddiert werden. Am Ende der Verzweigung drei wird immer geprüft, ob der aktuelle **readindex** der Länge des Pakets entspricht. Ist dies nicht der Fall wird die Verzweigung drei so oft in der loop() Methode wiederholt bis das Ende eines Paketes erreicht wird.

```

1. case 3:
2.   receiverbuffer.buffer[bufferreadindex] = recievedbyte;
3.   bufferreadindex++;
4.   checksumcalculation = checksumcalculation + recievedbyte;
5.   if (bufferreadindex == receiverbuffer.length)
6.   {
7.     bufferreceiverstate = 4;
8.   }
9.   break;


```

Wenn das Ende eines Pakets erreicht wird und das Switch Case in der Verzweigung vier und fünf sich befindet folgen nun die Checksummen Bytes des aktuellen Paketes. Die zwei checksummen Bytes werden in die jeweilige Globalen Variablen **checksum1** und **checksum2** gespeichert.

```

1. case 4:
2.   checksum1 = recievedbyte;
3.   bufferreceiverstate = 5;
4.   break;
5. case 5:
6.   checksum2 = recievedbyte;

```


| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

Der für die Weiterverarbeitung benötigte Teil des Pakets befindet sich nun im **recievebuffer.buffer[]**. Um die Richtigkeit und Integrität des aktuellen Paketes zu gewährleisten wird nun die Variable **checksumcalculation** mit einem XOR und 0xFFFF logisch verknüpft. Wenn die gebildete Checksumme mit der Paket Checksumme übereinstimmt wird das Paket mit der Methode processData() decodiert und weiterverarbeitet.

```

1. checksumcalculation = checksumcalculation ^ 0xffff;
2. if (checksumcalculation == ((uint16_t)(checksum2 << 8) + (uint16_t)checksum1))
3. {
4.     processData();
5. }

```


| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

In der Verzweigung fünf findet ebenfalls das senden statt. Es darf nur zwischen bestimmten Paketen gesendet werden. In der Verzweigung fünf wird genau nach dieser bestimmten Paket Struktur geschaut. Jedes Mal, wenn das Paket empfangen wird und ein aktuelles Paket zum Senden bereit steht wird eine Globale Variable **sendecounter** hochgezählt. Erreicht der **sendecounter** die 60 wird das aktuell vorbereitete Paket welches sich in dem Globalen Struct Array **sendebuffer.buffer** befindet mit der Methode `Serial.write()` versendet. Damit das eigene Paket nicht dekodiert wird, wird vor dem Senden eines Pakets die Methode `disableRX()` aufgerufen. Nach dem das Paket gesendet wurde wird `enableRX()` aufgerufen um das Empfangen von Paketen wieder zu ermöglichen. Als nächsten wird das Flag **packagePreparedFlag** auf 0 gesetzt damit, dass nächste zu versendende Paket vorbereitet wird. Der letzte Teil setzt die Globale Variable **bufferrecieverstate** auf null. Dies bewirkt das, dass nächste empfangene Paket den gesamten `recievedata()` Methoden Prozess durchläuft.

```

1.  if (receiverbuffer.buffer[bAddr] == 0x20 && receiverbuffer.buffer[bcmd] == 0x65 && receiverbuffer.buffe
    r[bArg] == 0x00 && packagePreparedFlag == 1)
2.  {
3.
4.      sendecounter++;
5.      //we will only send if we reciev 60x
6.      if (sendecounter > 60)
7.      {
8.          disableRX(); //disable reciever to not get our on messages
9.
10.         Serial2.write(sendbuffer.buffer, sendbuffer.length);
11.
12.         enableRX();
13.
14.         packagePreparedFlag = 0; //next package can be prepared
15.         sendecounter = 0;
16.     }
17. }
18.
19. bufferreceiverstate = 0; //next packet can be read start from beginning of switch case
20. break;
21. }
22. }

```

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

15.2.4 void processData()

Die Methode wird innerhalb der recievedata() Methode aufgerufen, wenn die Checksummenprüfung des aktuell empfangen Paketes erfolgreich ist. Die Paketstruktur jedes Paketes welches sich innerhalb des **recieverbuffer.buffer** befindet ist bekannt. Die Methode ist beliebig erweiterbar hierfür muss nur der Switch Case um das geforderte Paket erweitert werden.

Das Erste Paket welches die Struktur | 0x20 | 0x65 | 0x00 | hat, enthält die Brems und Beschleunigungswerte.

```

1.  switch (receiverbuffer.buffer[0])
2.  {
3.  case REQUESTTOESC: //0x20
4.      switch (receiverbuffer.buffer[1])
5.      {
6.      case 0x65:
7.          switch (receiverbuffer.buffer[2])
8.          {
9.          case 0x00:
10.             throttlebrakeValues.throttle = receiverbuffer.buffer[4];
11.             throttlebrakeValues.brake = receiverbuffer.buffer[5];
12.


```

Das zweite Paket welches die Struktur | 0x23 | 0x01 | 0xB0 | hat, enthält die Hauptinformation: Geschwindigkeit, Akkustand, Kilometerzähler, Temperatur und gefahrenen Kilometer seit dem Start des Rollers.

```

1.  case REPLYFROMESC: //0x23
2.      switch (receiverbuffer.buffer[1])
3.      {
4.      case 0x01: //read
5.          switch (receiverbuffer.buffer[2])
6.          {
7.          case 0xB0:
8.              maininformation.speed = ((int16_t)receiverbuffer.buffer[13]) | ((int16_t)receiverbuffer.buffer[14] << 8);
9.              maininformation.batterylevel = ((uint16_t)receiverbuffer.buffer[11]) | ((uint16_t)receiverbuffer.buffer[12]
<< 8);
10.             maininformation.odometer = ((uint32_t)receiverbuffer.buffer[17]) | ((uint32_t)receiverbuffer.buffer[18] <
< 8) | ((uint32_t)receiverbuffer.buffer[19] << 16) | ((uint32_t)receiverbuffer.buffer[20] << 24);
11.             maininformation.tripdistance = ((uint16_t)receiverbuffer.buffer[21]) | ((uint16_t)receiverbuffer.buffer[22]
<< 8);
12.             maininformation.temperature = ((uint16_t)receiverbuffer.buffer[25]) | ((uint16_t)receiverbuffer.buffer[26]
<< 8);

```

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

Das dritte Paket welches die Struktur | 0x20 | 0x64 | 0x00 | hat enthält die Information, ob das Frontlicht aktuell an oder ausgeschaltet ist. Es wurde ebenfalls noch ein Fix eingebaut der nur einen Frontlicht Wert 0 für aus oder 100 für an erlaubt.

```

1.  case REQUESTTOBLE: //0x20
2.      switch (receiverbuffer.buffer[1])
3.      {
4.          case 0x64:
5.              switch (receiverbuffer.buffer[2])
6.              {
7.                  case 0x00:
8.                      if ((receiverbuffer.buffer[5] == 0 || receiverbuffer.buffer[5] == 100))
9.                      {
10.                         maininformation.headlightstate = receiverbuffer.buffer[5];
11.                      }


```

Ganz zum Schluss der Methode processdata() wird noch eine Globale Variable **renewdisplayFlag** auf true gesetzt. In der loop() Methode wird nur der OLED Bildschirm Inhalt aktualisiert wenn neue Daten von der Methode processData() dekodiert wurden.

```

1.  renewdispalyFlag = true;

```

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

15.2.5 void preparePacket(uint8_t option)

Aktuell kann die preparePaket() Methode ein Paket mit folgender Struktur bauen:


| 0x55 | 0xAA | 0x06 | 0x20 | 0xB0 | 0x20 | 0x02 | 0x28 | 0x57 | 0xFE |. Die Methode ist durch Hinzufügen einer zweiten Verzweigung im Switch Case beliebig erweiterbar. Der Parameter **option** erlaubt es auszuwählen welches Paket gebaut werden soll.

Die Methode hat einen Zeiger vom Typ queuebuffer. Der Zeiger zeigt auf den **sendbuffer** welcher später das komplett zu sendende Packet enthält. Für die Gültigkeit eines Paketes muss ebenfalls eine Checksumme berechnet werden, deshalb wird die Globale Variable **checksumcalculation** zu Beginn jeden Aufrufs der Methode auf 0x00 gesetzt.

1. queuebuffer *ptsendequeue = &sendbuffer;
2. checksumcalculation = 0x00;

Jedes Paket enthält immer die Header | 0x55 | 0xAA |. Die Header Werte werden vor der eigentlichen Paketauswahl in den **sendbuffer** geschrieben.

1. (*ptsendequeue).buffer[0] = HEADER55;
2. (*ptsendequeue).buffer[1] = HEADERAAX;

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

Der **sendbuffer** wird nun mit den restlichen benötigten Bytes befüllt um das Paket komplett zu bauen. Die Checksummen Berechnung finden zum Schluss statt und die letzten zwei Bytes werden mit der entsprechenden Checksumme befüllt.

```

1. //55 AA 06 20 61 B0 20 02 28 27 57 FE
2. (*ptsendqueue).buffer[2] = READESCDATA[0];
3. (*ptsendqueue).buffer[3] = READESCDATA[1];
4. (*ptsendqueue).buffer[4] = READESCDATA[2];
5. (*ptsendqueue).buffer[5] = 0xB0;
6. (*ptsendqueue).buffer[6] = 0x20;
7. (*ptsendqueue).buffer[7] = packetunkown;
8. (*ptsendqueue).buffer[8] = throttlebrakeValues.throttle;
9. (*ptsendqueue).buffer[9] = throttlebrakeValues.brake;
10.
11. checksumcalculation += READESCDATA[0];
12. checksumcalculation += READESCDATA[1];
13. checksumcalculation += READESCDATA[2];
14. checksumcalculation += 0xB0;
15. checksumcalculation += 0x20;
16. checksumcalculation += packetunkown;
17. checksumcalculation += throttlebrakeValues.throttle;
18. checksumcalculation += throttlebrakeValues.brake;
19. checksumcalculation ^= 0xFFFF;
20.
21. (*ptsendqueue).buffer[10] = (uint8_t)(checksumcalculation & 0xFF);
22. (*ptsendqueue).buffer[11] = (uint8_t)((checksumcalculation & 0xFF00) >> 8);

```

Die gesamte Größe des Paketes muss noch angegeben werden. Hier ist zu beachten das es sich nicht um die Länge | 0x06 | handelt. Die Größe des Pakets wird für die Method Serial.write() benötigt. Serial.write() benötigt als Parameter ein Byte Array sowie die gesamte Größe des Byte Arrays.

```

1. (*ptsendqueue).length = 12;


```

Der Schluss der Methode setzt noch die Globale Variable **packagePreparedFlag** auf eins. Das Flag wird immer gesetzt, wenn ein Paket bereits zum Senden ist und sich im **sendbuffer** Array befindet.

```

1. packagePreparedFlag = 1;

```


| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

15.2.6 void preparewrittePacket(uint8_t writeoption)


Die Methode baut die sieben benötigten Pakete um Registerwerte im Xiaomi M365 zu ändern. Der Parameter **writeoption** erlaubt es das benötigte Paket auszuwählen. Im folgendem wird nur ein Teil der Methode erläutert.

Die Methode hat wie die perparePacket() Methode einen Zeiger vom Typ queuebuffer. Der Zeiger zeigt auf den **sendbuffer** welcher später das komplett zu sendende Packet enthält. Für das bauen das Paketes muss ebenfalls eine Checksumme berechnet werden deshalb wird die Globale Variable **checksumcalculation** zu Beginn jeden Aufrufs der Methode auf 0x00 gesetzt.

1. queuebuffer *ptsendequeue = &sendbuffer;
2. checksumcalculation = 0x00;

Jedes Paket enthält immer die Header | 0x55 | 0xAA |. Die Header Werte werden vor der eigentlichen Paketauswahl in den **sendbuffer** geschrieben. Die Paketgröße der benötigten Pakete ist bei dieser Methode immer zehn und kann vor der eigentlichen Paketauswahl gesetzt werden.

1. (*ptsendequeue).buffer[0] = HEADER55;
2. (*ptsendequeue).buffer[1] = HEADERA;
3. (*ptsendequeue).length = 10; //write packages have all lenght of 10

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

Je nach ausgewählter Option wird der **sendbuffer** nun mit den restlichen benötigten Bytes befüllt um das Paket komplett zu bauen. Die Checksummen Berechnung finden zum Schluss statt und die letzten zwei Bytes werden mit der entsprechenden Checksumme befüllt.

```

1.  switch (writeoption)
2.  {
3.  case WKERSWEAK:                               //1 5AA42037B005DFF tested
4.      (*ptsendequueue).buffer[2] = 0x04;         //size of packet
5.      (*ptsendequueue).buffer[3] = REQUESTTOESC; //0x20
6.      (*ptsendequueue).buffer[4] = CMDWRITE;     //0x03
7.      (*ptsendequueue).buffer[5] = RKERS;        //register
8.      (*ptsendequueue).buffer[6] = KERSWEAK;
9.      (*ptsendequueue).buffer[7] = 0x00;
10.
11.     checksumcalculation += 0x04;
12.     checksumcalculation += REQUESTTOESC;
13.     checksumcalculation += CMDWRITE;
14.     checksumcalculation += RKERS;
15.     checksumcalculation += KERSWEAK;
16.     checksumcalculation += 0x00;
17.     checksumcalculation ^= 0xFFFF;
18.
19.     (*ptsendequueue).buffer[8] = (uint8_t)(checksumcalculation & 0xFF);
20.     (*ptsendequueue).buffer[9] = (uint8_t)((checksumcalculation & 0xFF00) >> 8);

```


Der Schluss der Methode setzt noch ein Globale Variable **packagePreparedFlag** auf

1. Das Flag hat die gleiche Funktion wie bei der preparePacket() Methode und signalisiert, dass sich ein Paket im sende Buffer befindet.

```

1.  packagePreparedFlag = 1;

```

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

15.2.7 void saveandloadconfig()

Die Methode wird einmalig in der setup() Methode aufgerufen. Das Programm muss für die Bildschirmsteuerung die aktuell gesetzten Werte für Tempomat, Rücklicht, KERS... kennen um diese richtig darstellen zu können.

```

1. void saveandloadConfig()
2. {
3.     bool configsaved = EEPROM.read(0);
4.     if (configsaved == true)
5.     {
6.         batterywarning = EEPROM.read(1);
7.         cruise_state = EEPROM.read(2);
8.         taillight_state = EEPROM.read(3);
9.         kers_state = EEPROM.read(4);
10.    }

```

15.2.8 void displayclear()


Um zu verhindern, dass der Bildschirm beim Anzeigen neuer Daten flackert wurde diese Methode mit aufgenommen. Das Flackern entstehe dadurch, dass die clear() Methode mehrfach in einer Verzweigung ausgeführt wird.

```

1. void displayclear(uint8_t newdisplayvalue)
2. {
3.
4.     if (olddisplayvalue != newdisplayvalue)
5.     {
6.         ic2display.clear();
7.         olddisplayvalue = newdisplayvalue;
8.     }
9. }

```

Vereinfacht versichert die displayclear() Methode, dass die ic2display.clear() Methode nur einmalig innerhalb einer Verzweigung ausgeführt wird.

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

15.2.9 void displaydata()


Die Methode wird für die Anzeige und Steuerung aller Menüs am OLED Bildschirm genutzt.

Zu Beginn der Methode wird eine Zeiger Variable deklariert. Die Zeigervariable zeigt auf die Hauptinformation die dargestellt werden soll. Für die Menü Steuerung wird die aktuelle Geschwindigkeit des M365 benötigt. Die Geschwindigkeit lässt sich aus einem dekodierten Paket entnehmen und muss durch 1000 geteilt werden um eine Umwandlung Km/h zu gewährleisten.

```
1. maininfo *ptrmaininformation = &maininformation;
2. uint16_t speed = abs(((ptrmaininformation).speed / 1000));
```

Das Beschleunigungsmenü wird geladen sobald die aktuelle Geschwindigkeit größer als 1 Km/h ist. Unterdessen wird die Globale Variable **display_option** auf null gesetzt. Diese entscheidet schlussendlich welches Menü angezeigt werden soll.

```
1. //display speed menu
2. if (speed > 1)
3. {
4.     display_option = 0;
5. }
```


| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

In Anlehnung werden von den Paketen dekodierten Brems- und Beschleunigungswerte Logische Zustände wie eine eins oder null zu gewissen. Eine eins bedeutet der Hebel gedrückt ist, null genau das Gegenteil.

```

1.  if (speed <= 2)
2.  {
3.    int8_t controlbrakevalue = -1;
4.    int8_t controlthrottlevalue = -1;
5.    if (millis() > buttontime + 500)
6.    {
7.      if (throttlebrakeValues.brake >= 160)
8.      {
9.        buttontime = millis();
10.       controlbrakevalue = 1;
11.      }
12.      else if (throttlebrakeValues.brake <= 50)
13.      {
14.
15.        controlbrakevalue = -1;
16.      }
17.      else
18.      {
19.        controlbrakevalue = 0;
20.      }
21.
22.      if (throttlebrakeValues.throttle >= 160)
23.      {
24.        buttontime = millis();
25.
26.        controlthrottlevalue = 1;
27.      }
28.      else if (throttlebrakeValues.throttle <= 50)
29.      {
30.        controlthrottlevalue = -1;
31.      }
32.      else
33.      {
34.        controlthrottlevalue = 0;
35.      }
36.    }

```

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

Um in das Navigationsmenü zu gelangen muss der Variable **controlbrakevalue** und **controlthrottlevalue** eine eins enthalten. Die Globale Variable **menuposition** wird für Steuerung der einzelnen Menü Optionen benötigt und beim Öffnen des Navigationsmenüs immer auf 0 gesetzt.

```

1.  if ((controlbrakevalue == 1) && (controlthrottlevalue == 1))
2.  {
3.
4.      menuposition = 0; //menu position to change values
5.      display_option = 1; //open navigation menu = 1
6.      ic2display.clear();
7.  }


```

Daraufhin folgt ein Switch Case welches die **display_option** Variable prüft. Jede Verzweigung im Switch Case stellt ein Menü dar welches angezeigt werden soll. Die erste Verzweigung d.h. wenn **display_option** eins ist, stellt das Navigationsmenü dar. Aktuell ist dies das einzige Menü welches eine Steuerung implementiert hat. Aus diesem Grund befindet sich innerhalb der Verzweigung eins ein weiter Switch Case welches nach der Variable **menuposition** prüft. Momentan können die Werte des Menüs je nachdem welche **menuposition** gerade gesetzt ist durch einmaliges drücken des Beschleunigungshebels verändert werden.

```

1.  switch (display_option)
2.  {
3.      case 1:
4.          if ((controlthrottlevalue == 1) && (controlbrakevalue == -1))
5.          {
6.              switch (menuposition)
7.              {

```

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

Derweil ist die Logik innerhalb der Verzweigungen sehr ähnlich aufgebaut, deshalb wird nur die Verzweigung null bis zwei erläutert. Wenn der Beschleunigungshebel die Logische eins enthält und die **menupostion** null ist wird die **batterywarning** Option je nach vorher instanziiertem Wert aktiviert oder deaktiviert. Infolgedessen muss der Zustand konsistent gehalten werden, deshalb wird der aktuell Wert in die EEPROM gespeichert.

```

1.  switch (menuposition)
2.  {
3.      case 0:
4.          if (batterywarning == true)
5.          {
6.              batterywarning = false;
7.          }
8.          else
9.          {
10.             batterywarning = true;
11.          }
12.         EEPROM.update(1, batterywarning);
13.         ic2display.clear();
14.         break;


```

Wenn die **menupostion** Variable aktuell mit dem Wert eins instanziiert ist und der Beschleunigungshebel ebenfalls die Logische eins hat wird **display_option** Variable auf drei gesetzt. Dies hat zur Folge, dass das Informationsmenü beim nächsten Durchgang der loop() Methode angezeigt wird.

```

1.  case 1:
2.      display_option = 3;
3.      ic2display.clear();
4.      break;

```

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

Die Verzweigung zwei ist die erste worin ein Register Wert beim M365 durch betätigten des Beschleunigungshebels geändert wird. Hierfür wird je nach Einstellung das benötigte Paket durch die Methode preparewritePacket() vorbereitet und bei nächsten loop() durchlauf gesendet. Wie bei der Verzweigung null muss der Wert ebenfalls konsistent gehalten werden. Dies geschieht indem der Wert ebenfalls in die EEPROM gespeichert wird.

```

1. case 2:
2.     if (cruise_state == true)
3.     {
4.         cruise_state = false;
5.         preparewritePacket(7);
6.     }
7.     else
8.     {
9.         cruise_state = true;
10.        preparewritePacket(6);
11.    }
12.    EEPROM.update(2, cruise_state);
13.    ic2display.clear();
14.    break;


```

Die **menuposition** Variable entscheidet schlussendlich welche Option aktuell im Navigationsmenü ausgewählt ist. Dieser Wert wird durch betätigten des Bremshebels verändert. Momentan gibt es im Navigationsmenü fünf Optionen zur Auswahl die genutzt werden können, dementsprechend springt die Variable **menuposition** bei >5 wieder auf den Anfang.

```

1. if ((controlbrakevalue == 1) && (controlthrottlevale == -1))
2. {
3.     if (menuposition < 5)
4.     {
5.         menuposition++;
6.     }
7.     else
8.     {
9.         menuposition = 0;
10.    }
11. }

```


| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

Im Anschluss werden die benötigten Zeichen auf den Bildschirm angezeigt und je nachdem welcher Wert bei **menuposition** instanziiert ist ein Pfeil dargestellt welcher die aktuell ausgewählte Option darstellt. Die folgenden Programmzeilen sind gekürzt

```

1. ic2display.set1X();
2. ic2display.setCursor(15, 0);
3.
4. if (menuposition == 0)
5. {
6.     ic2display.print((const __FlashStringHelper *)d_arrow);
7. }
8. else
9. {
10.    ic2display.print(" ");
11. }
12. ic2display.print((const __FlashStringHelper *)d_navi_batterywarning);
13. if (batterywarning == false)
14. {
15.    ic2display.print((const __FlashStringHelper *)d_off);
16. }
17. else
18. {
19.    ic2display.print((const __FlashStringHelper *)d_on);
20. }
21. ic2display.setCursor(15, 1);
22. if (menuposition == 1)
23. {
24.    ic2display.print((const __FlashStringHelper *)d_arrow);
25. }
26. else
27. {
28.    ic2display.print(" ");
29. }
30. ic2display.print((const __FlashStringHelper *)d_navi_batteryinformation);


```

Die Globale Variable **display_option** ist standardmäßig mit dem Wert zwei instanziiert welches in der Verzweigung zwei das Hauptmenü darstellt.

```

1. case 2:
2.     displayclear(0);
3.     ic2display.set1X();
4.     ic2display.setCursor(0, 0);
5.     ic2display.print((const __FlashStringHelper *)d_main_battery);
6.     ic2display.setCursor(70, 0);
7.     ic2display.print((const __FlashStringHelper *)d_main_headtlight);
8.     ic2display.setCursor(20, 1);
9.     ic2display.set2X();
10.    ic2display.print((*ptrmaininformation).batterylevel);
11.    ic2display.setCursor(76, 1);

```

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

Derweil kann die Frequenz des Blinkens, sowie das Batterie Level ab dem das Blinken Beginnen soll eingestellt werden.

```


1.  if (((*ptrmaininformation).batterylevel < 56) && (batterycounter == 50) && (batterywarning == true))
2.  {
3.      ic2display.print((const __FlashStringHelper *)d_b);
4.      batterycounter = 0;
5.  }
6.  else
7.  {
8.      batterycounter++;
9.      ic2display.print(" ");
10. }
```

Die dritte Verzweigung stellt das Informationsmenü dar. Das Informationsmenü kann nur aus dem Navigationsmenü geöffnet werden. Die Variable **display_option** muss hierfür auf drei gesetzt sein.

```

1.  displayclear(6);
2.  ic2display.setCursor(30, 0);
3.  ic2display.set1X();
4.  ic2display.print((const __FlashStringHelper *)d_info_odometer);
5.  ic2display.setCursor(60, 1);
6.  ic2display.set2X();
7.  ic2display.print((*ptrmaininformation).odometer / 1000);
8.  ic2display.setCursor(0, 3);
9.  ic2display.set1X();
10. ic2display.print((const __FlashStringHelper *)d_line);
11. ic2display.setCursor(10, 4);
12. ic2display.print((const __FlashStringHelper *)d_info_tripdistance);
13. ic2display.setCursor(60, 5);
14. ic2display.set2X();
15. ic2display.print((*ptrmaininformation).tripdistance / 100);
```

Das Informationsmenü kann durch drücken des Beschleunigungshebels verlassen werden. Daraufhin wird die Variable **display_option** auf eins gesetzt welches das Navigationsmenü darstellt.

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

Der letzte Teil der Methode stellt das Beschleunigungsmenü dar. Dieses Menü befindet sich nicht indem Switch Case.

```

1. //show speed menu when no other menu option is used
2. if ((display_option != 1) && (display_option != 2) && (display_option != 3))
3. {
4.   displayclear(2);
5.   ic2display.set1X();
6.   ic2display.setCursor(89, 0);
7.   ic2display.print((const __FlashStringHelper *)d_km);
8.   ic2display.setCursor(58, 1);
9.   ic2display.set2X();
10.  ic2display.print(speed);
11.  ic2display.setCursor(0, 3);
12.  ic2display.set1X();
13.  ic2display.print((const __FlashStringHelper *)d_line);
14.  ic2display.setCursor(40, 4);
15.  ic2display.set1X();
16.  ic2display.print((const __FlashStringHelper *)d_main_battery);
17.  ic2display.setCursor(58, 5);
18.  ic2display.set2X();
19.  ic2display.print((*ptrmaininformation).batterylevel);
20.  display_option = 2;
21. }

```

15.2.10 void setup()

Der Inhalt der Methode wird vom Arduino nur einmalig ausgeführt. Die Methode initialisiert die UART Verbindung und holt die aktuell persistenten Daten aus der EEPROM und setzt zum Schluss die benötigte Konfiguration um den OLED Bildschirm anzusteuern.

```

1. void setup()
2. {
3.   Serial.begin(115200);
4.
5.   saveandloadConfig();
6.
7.   //display configuration
8.   ic2display.setI2cClock(400000L);
9.   ic2display.begin(&Adafruit128x64, I2C_ADDRESS);
10.  ic2display.setFont(System5x7mod);
11.  ic2display.clear();
12. }

```

15.2.11 void loop()

Wird ständig wiederholt und ruf die bereits beschriebenen Methoden auf um das digitale Armaturenbrett zu ermöglichen.

```

1. void loop()
2. {
3.   recieveData();
4.
5.   if (packagePreparedFlag == 0)
6.   {
7.     preparePacket(1);
8.   }
9.
10.  if (renewdispalyFlag == true)
11.  {
12.    displaydata();
13.    renewdispalyFlag = false;
14.  }
15. }

```

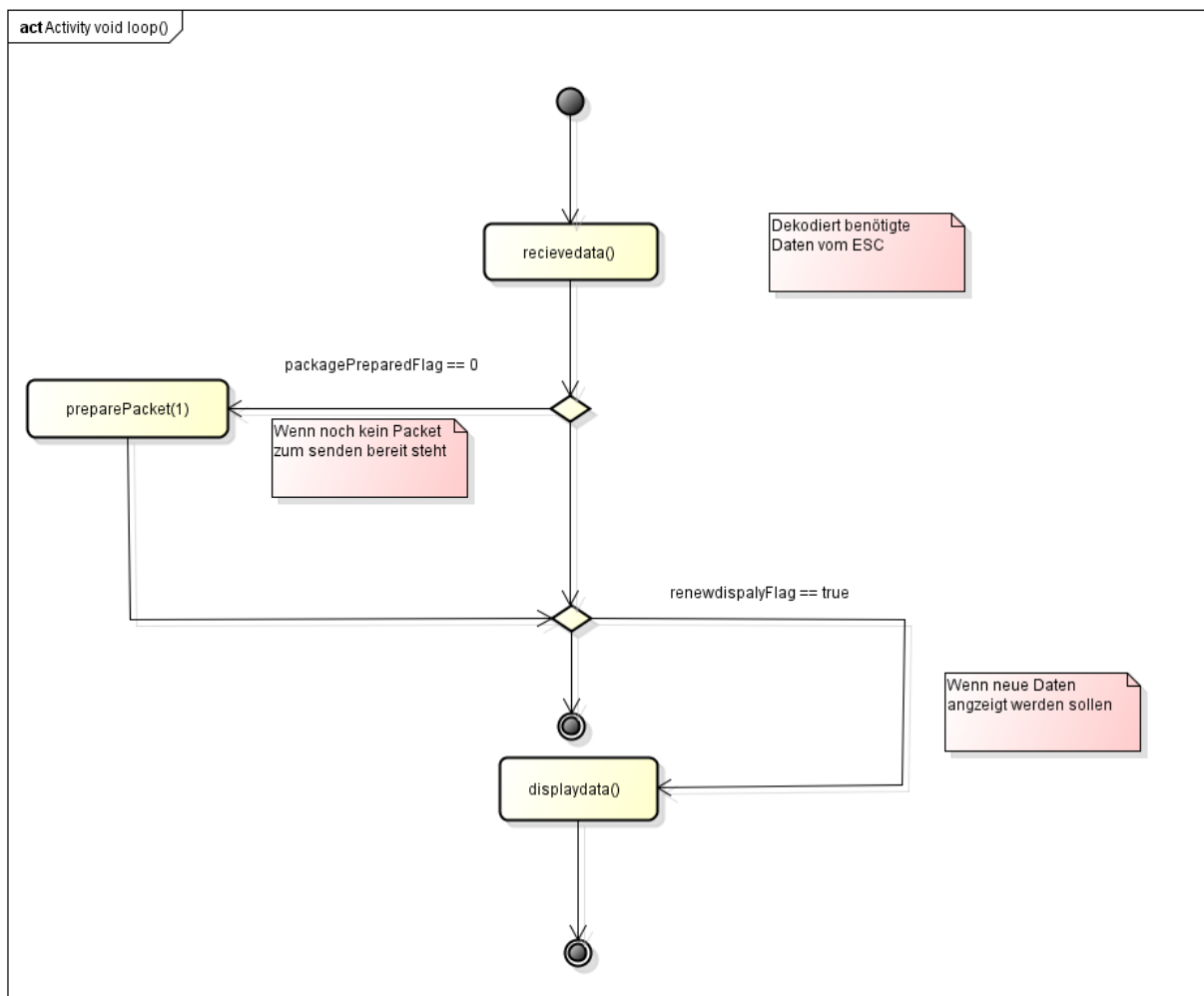



Abbildung 13 loop() Methode

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

15.3 QUELLEN PROGRAMMIERUNG


Laut Lizenz (siehe Kapitel 17) weiße ich Explizit darauf hin das die Methode `recievedata()` nach Abstimmung mit dem Entwickler aus einem ⁹Projekt entnommen wurde und nach meinen Bedürfnissen angepasst.

Die grobe Struktur der ¹⁰Bildschirm Steuerung wurde ebenfalls verbessert nach meinen Bedürfnissen angepasst.

⁹

https://gitlab.com/esp32m365/esp32_xiaomi_m365_display/blob/master/esp32_m365_oled/src/m365client.cpp

¹⁰ https://github.com/augisbud/m365_dashboard/blob/master/M365/M365.ino

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

16 PROBLEME & LÖSUNGEN

16.1 ZWEI SERIELLE VERBINDUNGEN

Problem:

Für das Debugging der Kommunikation wurde Hardware benötigt welche zwei UART Verbindungen gleichzeitig ermöglicht. Aus diesem Grund wurde zu Beginn ein Arduino Uno gekauft. Dieser kann mit der Software Serial Bibliothek¹¹ zwei UART Verbindungen aufbauen jedoch kann der Uno nicht die erforderliche Baudrate liefern.

Lösung:

Der Arduino Mega wurde besorgt dieser besitzt über drei Physikalische UART Schnittstellen welche alle die erforderliche Baudrate liefern können.

16.2 PAKETE SENDEN


Problem:

Zu häufiges Senden von Paketen lässt das Arduino Programm abstürzen.

Lösung:

Ein Counter wurde implementiert der nach einer Anzahl X das aktuelle Paket im Buffer sendet.

¹¹ <https://www.arduino.cc/en/Reference/SoftwareSerial>

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

16.3 RÜCKWÄRTS FAHREN ZEIGT FALSCHES GESCHWINDIGKEIT

Problem:

Der Bildschirm zeigt beim Rückwärtsfahren eine Geschwindigkeit von 65Km/h

Lösung:

Datentyp welcher die Geschwindigkeit enthält musste angepasst werden.

16.4 FRONTLICHT FLACKERT

Problem:

Bei einem meiner zwei M365 flackert das Frontlicht beim Einschalten.

Lösung:

¹²Laut dem Telegram Gruppe „Display for Xiaomi Mijia M365“ soll eine Sicherung kaputt sein bzw. muss neu gelötet werden. Bisher kam ich noch nicht dazu dies zu testen, da das Problem nicht als kritisch angesehen wird.


16.5 M365 LIEFERT KEINE 5V

Problem: Einer meiner zwei M365 Roller liefert keine 5V an dem Pin der den Arduino Pro Mini mit Spannung versorgen soll.

Lösung:

Die gleiche Sicherung welche das Frontlicht zum Flackern bringt soll, verursacht dieses Problem. Bisher kam ich hier noch nicht dies zu testen, da ich einfach der anderen M365 genutzt habe.

¹² Siehe Anhang Fuse_F2.png sowie M365_Fuse.pdf

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

16.6 FRONTLICHT DEKODIERUNG

Problem:

Die Bildschirmanzeige hat den Frontlichtwert ständig falsch angezeigt. Die Methode welches das benötigte Paket dekodiert hat verschiedene Werte geliefert.

Lösung:

Nach langer Analyse hat sich herausgestellt, dass der Wert 100 Frontlicht an darstellt steht und 0 Frontlicht aus darstellt.

16.7 BILDSCHIRM NAVIGATION

Problem:

Die Bildschirmnavigation im Navigationsmenü reagiert zu schnell auf den Beschleunigungs- und Bremshebel.

Lösung:


Beim Betätigen eines Schalters wird eine Timer Variable instanziiert. Ein neuer Wert für einen Schalter kann erst nach einer gewissen Zeit eingelesen werden.

16.8 GESCHWINDIGKEIT ANZEIGE ZU KLEIN

Problem: Eine Testfahrt hat leider ergeben, dass die Geschwindigkeitsanzeige bei der Fahrt zu klein angezeigt wird. Die aktuell verwendete Bibliothek erlaubt es nur die Schriftgröße zu verdoppeln

Lösung:

Bitmaps erlauben es Zeichen größer darzustellen. Implementierung der Bitmaps werden aus Zeitgründen erst zur Präsentation vorgeführt.

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

17 LIZENZ UND VERÖFFENTLICHUNG AUF GITHUB


Nach Abschluss der Präsentation wird der Code für jeden zugänglich auf Github veröffentlicht. Als Lizenz wird die ¹³Creative Commons V4 – BY NC SA. Demzufolge darf jeder in jedem Medium den Code bearbeiten und verteilen. Es müssen folgende Bedingungen eingehalten werden:

Namensnennung: Es müssen angemessene Urheber- und Rechteangaben gemacht werden. Einen Link zur Lizenz beigefügt und angeben, ob Änderungen vorgenommen wurden.

Nicht kommerziell: Das Material darf nicht für kommerzielle Zwecke genutzt werden.

Weitergabe unter gleichen Bedingungen: Wenn das Material verändert oder anderweitig direkt darauf aufbauen, dürfen die Beiträge nur unter derselben Lizenz wie das Original verbreitet werden.

¹³ <https://creativecommons.org/licenses/by-nc-sa/4.0/>

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

18FAZIT

Die Projektarbeit hat meine Kenntnisse besonders in der Hardware nahen Programmierung erweitert. Konkret haben sich besonders das Wissen in der Hochsprache C ausgeprägt. Die angegebenen Mußkriterien konnten alle erfüllt werden und benötigten nur noch einen kleinen Feinschliff. Derzeit wird ebenfalls noch an den Wunschkriterien Zusatz Akku sowie der Firmware Optimierung gearbeitet, diese können sicherlich bis zur Präsentation ebenfalls erledigt werden. Wie Kapitel 16 Probleme & Lösungen beschrieben ist, sind noch ein paar Probleme offen für welche Sicherlich aber dauerhafte Lösungen gefunden werden

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken die Information über die Kommunikation rund um den M365 veröffentlicht haben:

¹⁴Paco Gorina: Für die Recherche rund um das Protokoll und deren Packet Struktur.

¹⁵CamiAlfa: Für die Recherche um den Inhalt des Protokolls, Checksummen Berechnung und die Bereitstellung einer Bibliothek.

¹⁶smartnick: Welcher mir bei technischen Fragen rund um die Kommunikation zur Seite stand.

¹⁷augisbud: Welcher die Verkabelung zum M365 zur Verfügung gestellt hat.

¹⁸4ng4rp3: Welcher die benötigten 3D Dateien für das Gehäuse zur Verfügung gestellt hat.


¹⁴ <http://www.gorina.es/9BMetrics/>

¹⁵ <https://github.com/CamiAlfa/M365-BLE-PROTOCOL>

¹⁶ https://gitlab.com/esp32m365/esp32_xiaomi_m365_display

¹⁷ https://github.com/augisbud/m365_dashboard

¹⁸ <https://www.thingiverse.com/thing:3042060>

| | | |
|---|---|--|
|  | Technikerarbeit 2018 / 2019 | it.schule stuttgart Breitwiesenstrasse 20-22 70565 Stuttgart |
| | Programmierung eines Mikrocontrollers für die Einbindung von Daten in ein digitales Armaturenbrett. | |

19 ERKLÄRUNG

Hiermit erkläre ich, dass die vorliegende Technikerarbeit eine eigenständige Leistung darstellt und nicht auf der Basis einer bereits vorhandenen Techniker-, Diplom- oder ähnlicher Arbeit erstellt wurde. Verwendete Quellen habe ich vollständig und nachprüfbar aufgeführt. Bei der Durchführung und Ausarbeitung wurden nur die zulässigen Hilfsmittel verwendet.

Mir ist bewusst, dass bei einem Verstoß gegen diese Erklärung innerhalb der gesetzlichen Einspruchsfristen auch im Nachhinein die Leistungsbewertung aberkannt werden kann. Damit erlischt die Berechtigung zum Tragen der Berufsbezeichnung „staatlich geprüfter Techniker“.

Stuttgart, den