OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG

INFORMATIK UND MATHEMATIK

Ergebnisvorstellung StarCar - Gruppe 2

12 - 1 - 2018

Gliederung

1. Projektziel
2. Software
3. Hardware
4. Sensoren
5. Steuerung
6. Kommunikationsprotokoll
7. Benutzeroberfläche
8. Integration

Projektziel

- Fahrtüchtiges Auto programmieren
- Zwei verschiedene Steuerungsmethoden erarbeiten
- Sensordaten ermitteln und ausgeben
- Grobe Raumdarstellung ermitteln
- Benutzeroberfläche erstellen
- Übertragung zwischen Arduino Mega und Raspberry Pi 3 aufsetzen

Software

Software

- Programmiersprachen: C/C++, Python und C#
- Raspian OS
- Arduino IDE 1.8.5
- Visual Studio 2017 15.3.1 + Arduino IDE Erweiterung
- IAR Embedded Workbench 6.50.1
- Device Monitoring Studio 7.81
- Advanced Serial Port Terminal 6.0
- PicoScope 6
- Fritzing 0.9.3

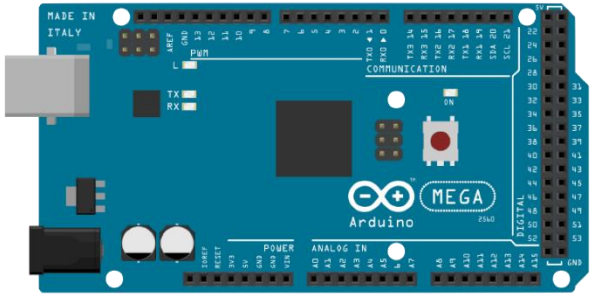
Software

- Simple Motor Control Center 1.2.0.0
- MATLAB R2017b
- PuTTY
- PSFtp
- Urg Viewer
- CuteCom
- DecaRangeRTLS

Hardware

Hardware

- Raspberry Pi 3
- Raspberry Pi 3 JOY-iT Display
- Arduino Mega 2560
  (zuvor Arduino Uno)
- Arduino USB Host Shield



Hardware

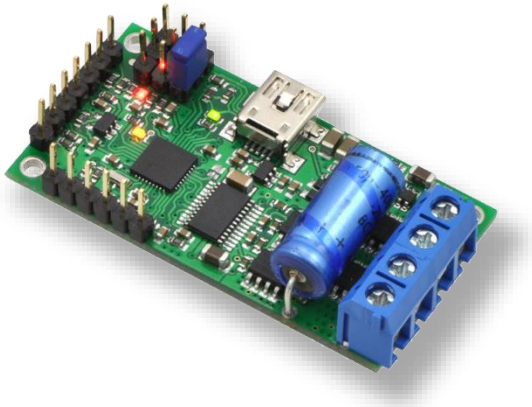
- XBOX 360 USB Controller von Microsoft
- eZ430-Chronos von Texas Instruments
- eZ430-Chronos-AccessPoint von Texas Instruments

Sensoren / Aktuatoren

Steuerung – Implementierung

Fahrzeug-seitig

- Pololu Simple Motor Controller 24v12
  - Ansteuerung: Serielle Schnittstelle
  - Protokoll: Pololu Binary Commands
  - Verkabelung: RX, TX und RST PINs
  - Steuerung im Bereich von -1000 bis 1000
    - negativer Wert = vorwärts fahren
    - positiver Wert = rückwärts fahren



Steuerung – Implementierung

Fahrzeug-seitig

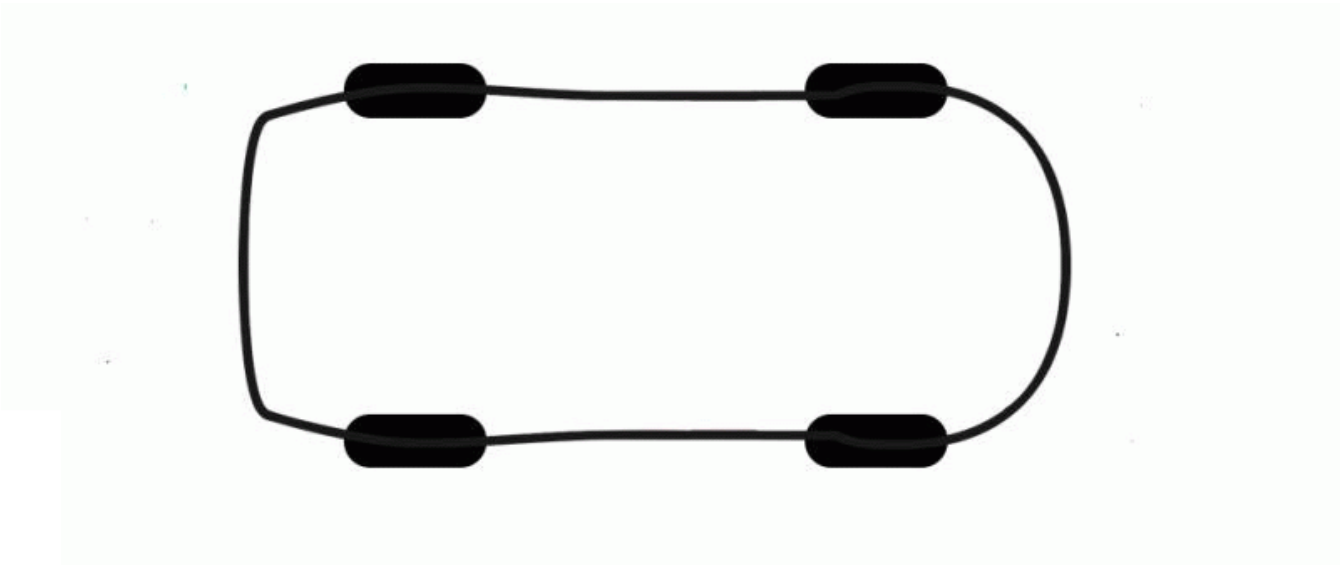
- RC-Car Servo 4519 DBB MG
  - Ansteuerung: Arduino Servo Library
  - Funktionsweise: 16 Bit Timer zur PWM
  - Verkabelung: SIG PIN
  - Steuerung im Bereich von 1100 bis 1600
    - linker Anschlag: 1100
    - neutrale Stellung: 1365
    - rechter Anschlag: 1600

Ultraschallsensor

- PING))) Ultrasonic Distance Sensor
  - Enger Akzeptanzwinkel
    - Reichweite: ca. 3 cm bis 3 m
    - 3-poliger Stecker
    - Leistungsanforderungen: +5 VDC; 35 mA aktiv
    - Kommunikation: positiver TTL-Implus

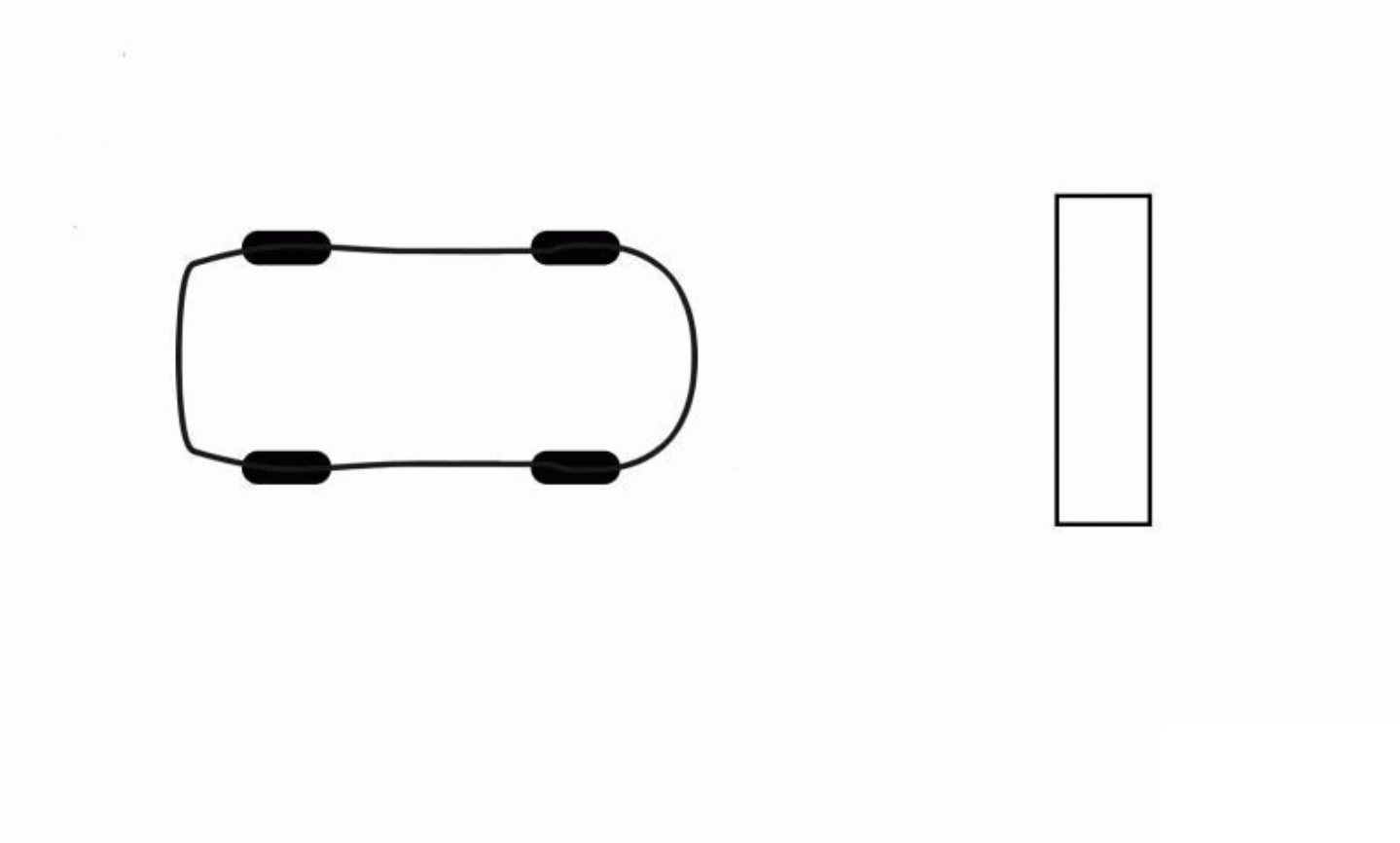
Ultraschallsensor

Funktionsweise



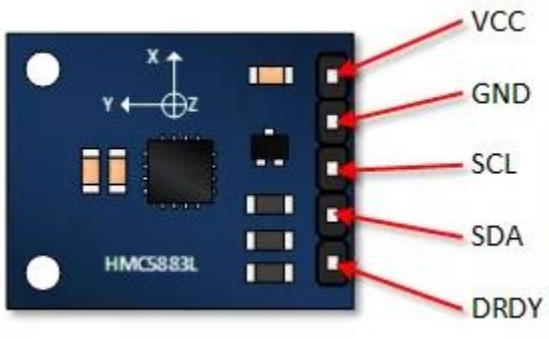
Ultraschallsensor

Stopp-Bedingung



Kompasssensor

- Tripple Achsen Messung
- I²C digital Interface
- Genauigkeit: ± 2° Grad Kompass Richtung

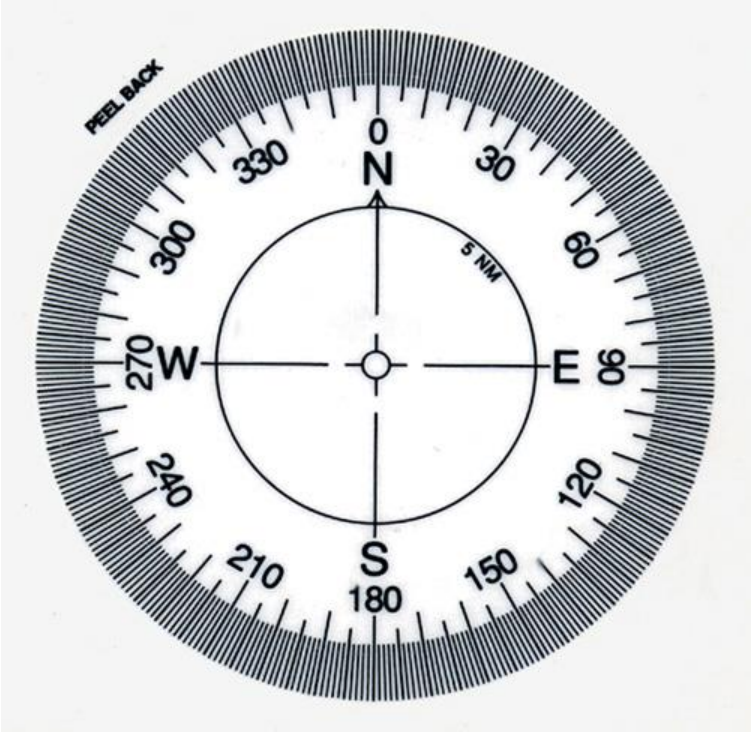


Kompasssensor

- Rohdaten in Nutzdaten umwandeln
- Magnetische Missweisung
- Weitere Umwandlung der werte

Kompasssensor

- Norden: +0°
- Osten: +90°
- Süden: -0°
- Westen: -90°



Beschleunigungssensor

- Tripple Achsen Messung
- I²C und SPI Digital Interface
- Genauigkeit kann eingestellt werden von 2 - 16g

Beschleunigungssensor

- Ausgegebene Werte:
  - Sofort nutzbar
  - Positiv beim Vorwärts fahren
  - Negativ beim Rückwärts fahren

Übertragung der Sensordaten

- Einen Bit-Frame für Sensoren erstellen
  - Ultraschall (front & back)
  - Beschleunigungssensor
  - Kompasssensor
- Abspeicherung aller Werte als int
- Bit-Frame Idee wurde verworfen
- Übertragung der Sensoren einzeln per Protokoll

Ultra-Wideband-Sensor

- 4 x EVB1000-Module
  - Einstellung der Dips
  - Mehrere verschiedene Einstellungen unter Berücksichtigung der Dokumentation verwendet
  - Keine Ausgabe der Information

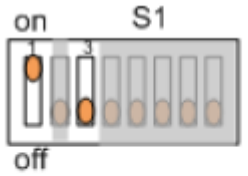


Figure 12: USB to SPI configuration

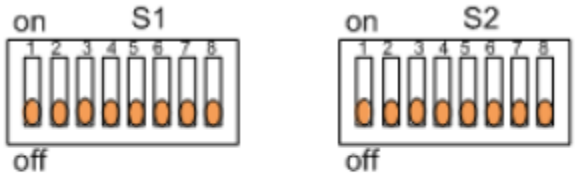


Figure 14: S1 and S2 configuration for external application control through USB

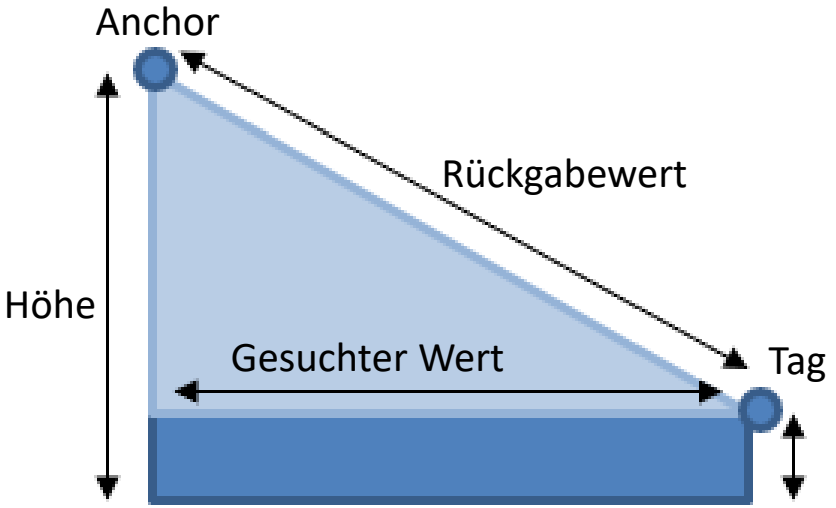
Ultra-Wideband-Sensor

- Programm DecaRangeRTLS
- Grundeinstellungen beim Kauf des Kits eingestellt
- Keine Werte bei der Ausführung des Programmes

Ultra-Wideband-Sensor

Berechnung

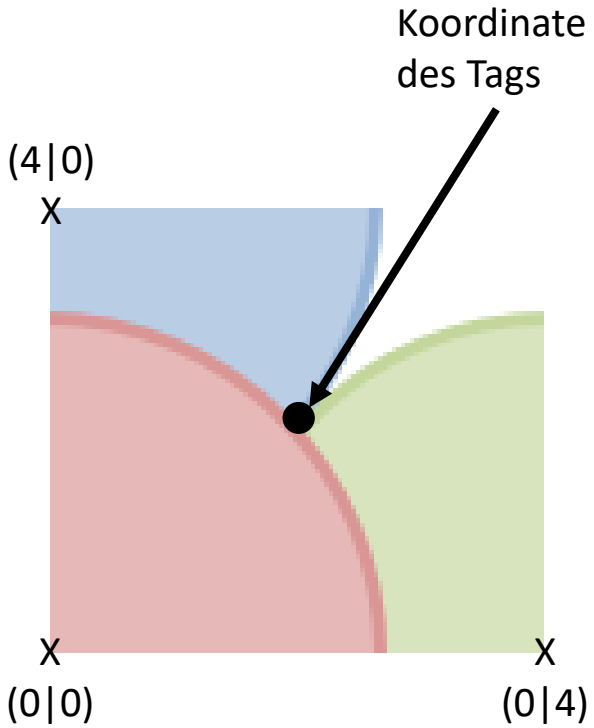
- 3 Anchor
- 1 Tag
- Berechnung der Entfernung von Anchor zu Tag
- Umwandlung von 3-dimensionaler Entfernung in 2-dimensionale Entfernung



Ultra-Wideband-Sensor

Berechnung

- Berechnung im Idealfall:
  - Anchor erhält Entfernung zu allen 3 Tags
  - Radius des Tags wird berechnet
  - Berechnung der Schnittpunkte der Kreise
  - Schnitt aller drei Kreise entspricht Standort des Tags



Lidar

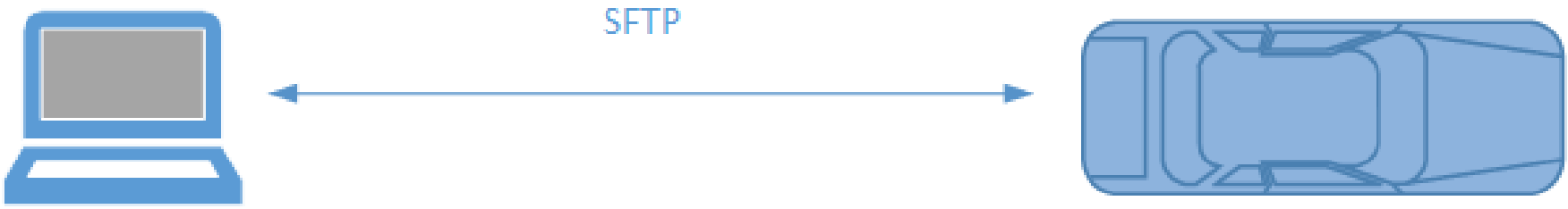
- Atmosphärenmessung durch Laserimpulse
- verwendeter Sensor: URG-04LX-UG01
- Reichweite: 0.06 - 4.0 m
- Messwinkel: 240 °
- Messgeschwindigkeit: 100 ms
- Steuerung durch Raspberry Pi über USB
- Verwendung der Hersteller Libraries

Lidar

- Funktion erstellt für einzelne Messung
- oft starkes Rauschen, viele fehlerhafte Messpunkte
- Verarbeitung der Daten mit MATLAB
- Glättung der Rohdaten mit gleitendem Mittelwert (Einzelmessung)
- Mittelung mehrerer Messungen

Sensor Fusion Datenübertragung

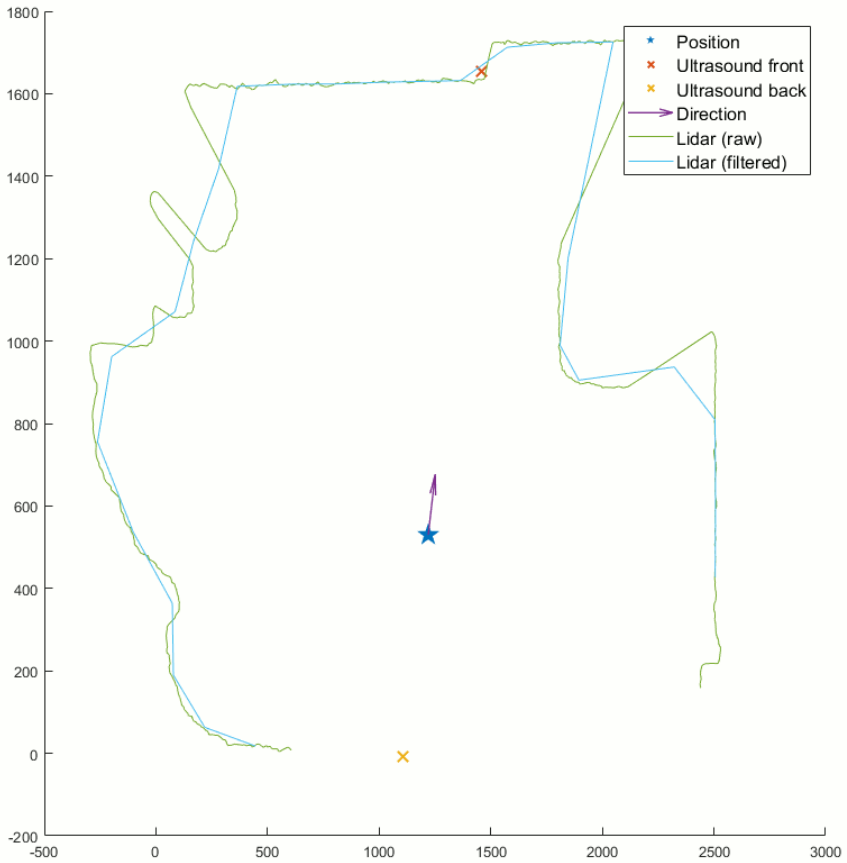
- Datenübertragung von Raspberry zu Laptop
- Sicherer Transfer durch SFTP
- Verarbeitung der Daten in MATLAB
- Transfer aktueller Daten in Intervallen



Sensor Fusion Datenübertragung

- Lidar: 682 x,y - Koordinaten, stark rauschend
- Ultraschall: Entfernung zu Gegenstand direkt vor/hinter Fahrzeug
- Magnetsensor: Gradzahl abweichend von Norden(0°)
- Beschleunigungssensor: Zahl für Bewegungsrichtung

Sensor Fusion Datenübertragung



Steuerung

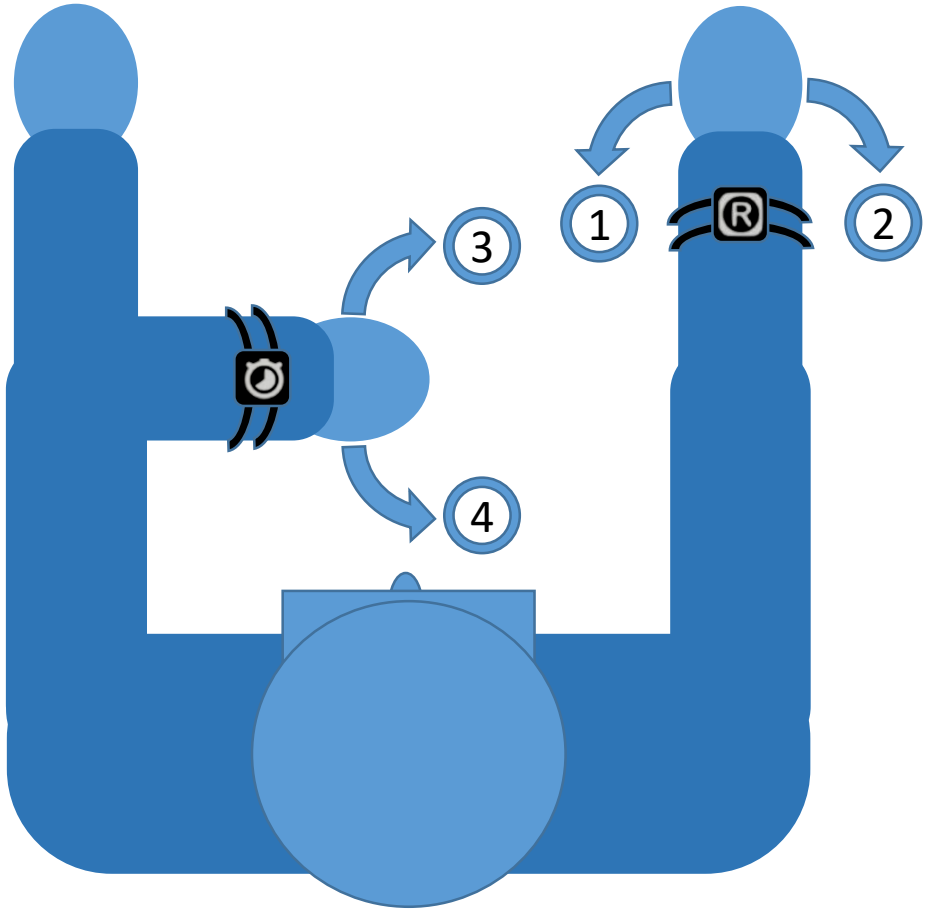
Steuerungsmodus – Sport Watch

Steuerungsschema

Steuerungsmodus – Sport Watch

Funktionsweise

1. Drehung nach **links**
   → nach **links** lenken
2. Drehung nach **rechts**
   → nach **rechts** lenken
3. Drehung nach **vorne**
   → **vorwärts** fahren
4. Drehung nach **hinten**
   → bremsen + **rückwärts** fahren

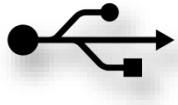


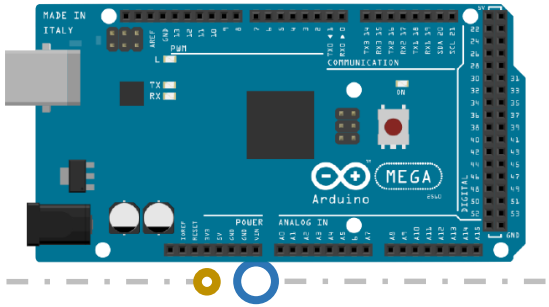
Steuerung – Implementierung

Controller-seitig: XBOX 360 USB Controller

- Ansteuerung: USB Host Shield Library 2.0
- Funktionsweise: „XBOXUSB“ Treiber
- Verkabelung: über USB Shield auf Arduino

- Steuerung (mit Sticks): im Bereich -32.768 bis 32.768
  - negativer Wert = links lenken / rückwärts fahren
  - positiver Wert = rechts lenken / vorwärts fahren
  - Steuerwert für Lenkung: (LS_X / 32.768)
  - Steuerwert für Beschleunigung: (RS_Y / 32.768)

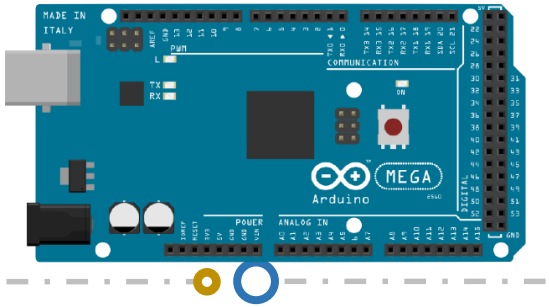




Steuerung – Implementierung

Controller-seitig: XBOX 360 USB Controller

- Steuerung (mit Trigger): im Bereich 0 bis 255
  - bei linkem Trigger: Rückwärts fahren
  - bei rechtem Trigger: Vorwärts fahren
  - Steuerwert für Beschleunigung: (-LT + RT) / 255

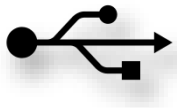


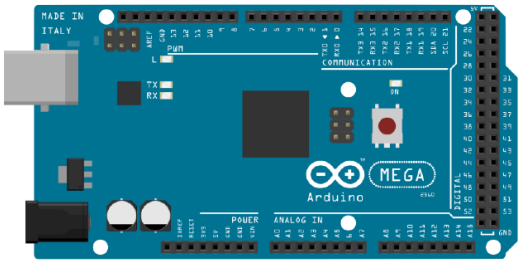
Steuerung – Implementierung

Controller-seitig: eZ430-Chronos

- Ansteuerung: USB Host Shield Library 2.0
- Funktionsweise: generischer USB Treiber
- Verkabelung: über USB Host Shield auf Arduino

- Verbindung (Arduino zum Access Point)
  - Protokoll: Serial Binary Commands
  - Befehle zum
    - Aktivieren / Deaktivieren des Access Points
    - Abfragen des Status und der Steuerdaten

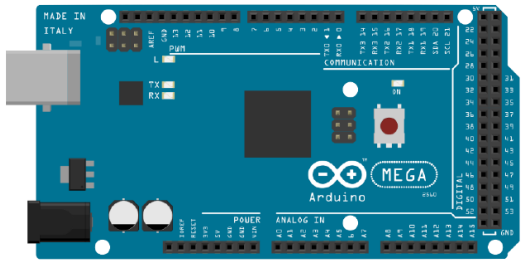




Steuerung – Implementierung

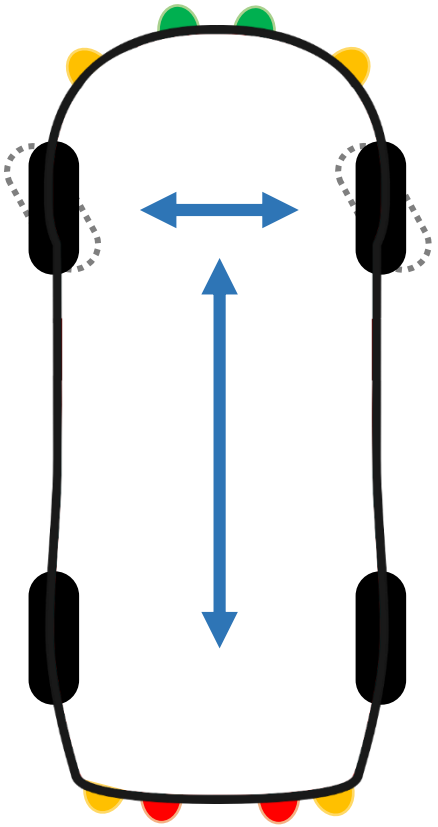
Controller-seitig: eZ430-Chronos

- Verbindung (Access Point zu Watches)
  - Protokoll: SimpliciTI über 868 MHz
  - Steuerdaten
    - Accelerator Daten (X, Y, Z)
    - Button ID → Tastenbefehl
    - Control ID → Steuermodus



Steuerungsmodi

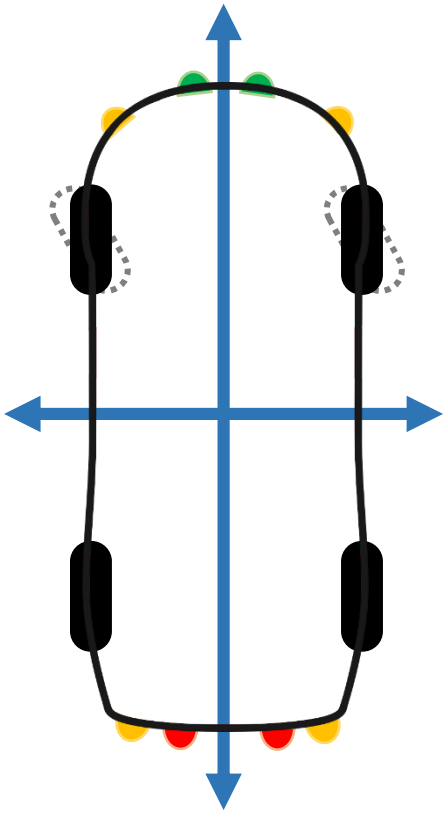
- Game Controller Steuerung
  - XBOX 360 USB Controller
  - 2 Modi für Beschleunigung
  - Steuert: Lenkung + Beschleunigung
- Sport Watch Steuerung
  - eZ430-Chronos
  - 2 Watches zur Steuerung
  - Steuern: Lenkung + Beschleunigung
- LED Steuerung
  - Beleuchtung in Fahrtrichtung
  - Leuchtsignaleffekte als Feedback bei
    - Ein- / Ausschalten des Motors
    - Erkennen eines Hindernisses



Steuerung

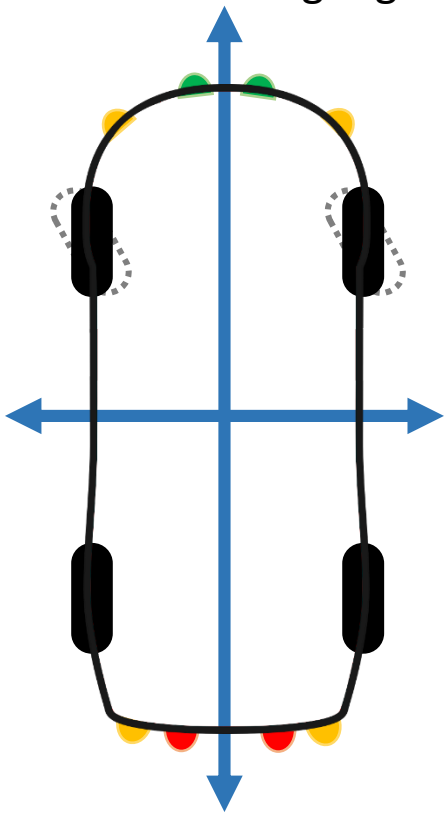
Umsetzung

1. Erfassung eines Steuerwertes
2. Skalierung des Steuerwertes um den 0-Punkt
3. Umrechnung des Steuerwertes in seinen Prozentsatz
4. Kalibrierung des Prozentsatzes
5. Speicherung des Prozentsatzes
6. Verarbeitung des Prozentsatzes
7. Umrechnung des Prozentsatzes in seinen Aktuatorwert
8. Anwendung des Aktuatorwertes



+100% = volle Beschleunigung nach **vorne**

-100% = Anschlag nach **links**

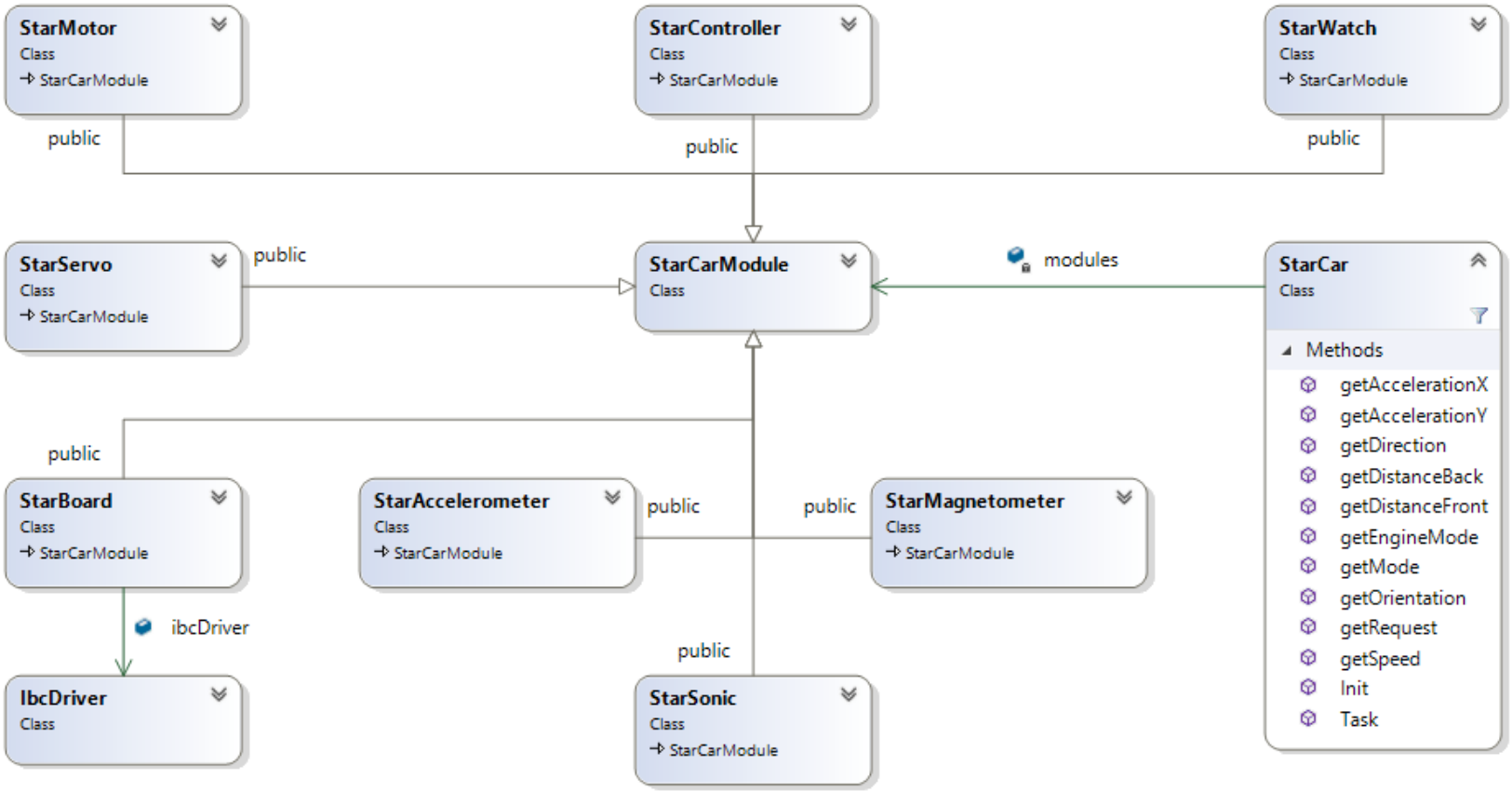


+100% = Anschlag nach **rechts**

-100% = volle Beschleunigung nach **hinten**

Integration aller Komponenten

Software-technisch



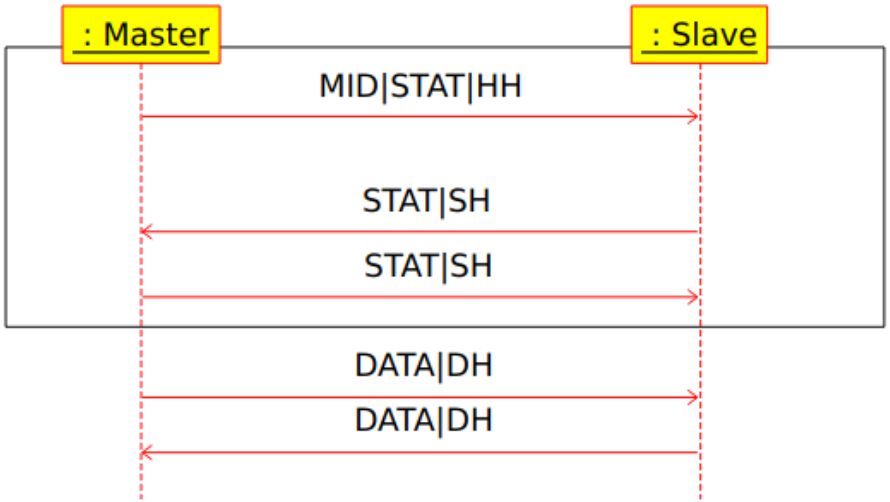
Kommunikationsprotokoll

Features

- IBP = Inter Board Protocol
- IBC = Inter Board Communication
- Aktive Features:
  - Identifizierbare Nachrichten
  - Fehlersicherheit
    - Kommunikationspartner bleiben synchron
    - Kommunikationspartner erkennen fehlerhafte Übertragung
  - Einfache Benutzung (Konfiguration pro Nachrichtenart nur an einer Stelle erforderlich)

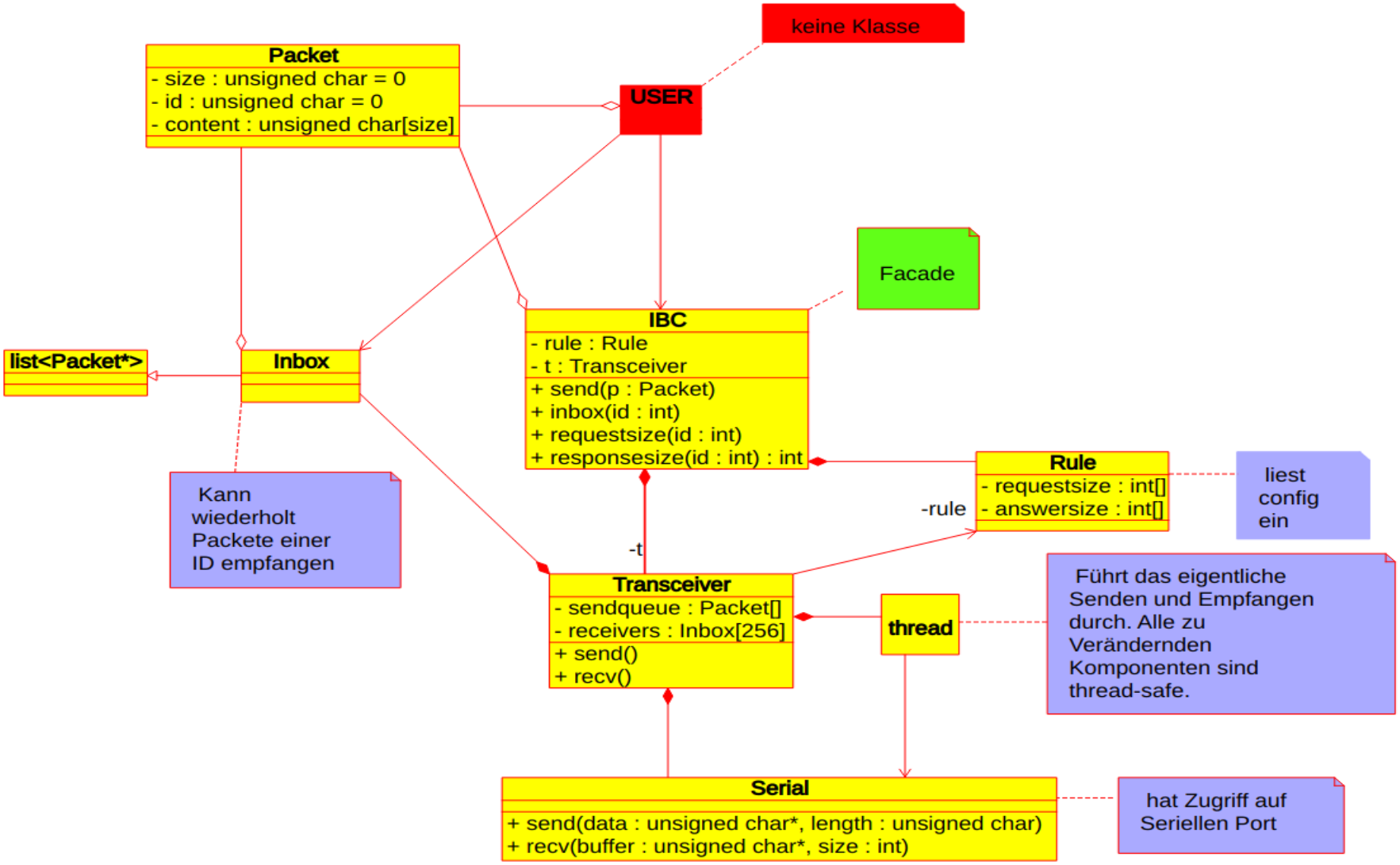
Ablauf einer Kommunikationseinheit

MID :[8bit] Message ID
stellt auch Größeninformationen dar
STAT:[4bit] Statusfeld für Kommunikationsrelevante Flags
SH :[4 bit] Statushash (Checksumme über STAT)
HH :[4 bit] Headhash (Checksumme über MID und STAT)
DATA[x|y bit] : Payload
DH :[8bit] Datenhash (Checksumme über gesendete Payload)



"3 Way Handshake", um die Wahrscheinlichkeit einer falschen Übertragung der ID und damit falscher Größeninformation der Payload, zu verringern.

API masterseitig



Benutzungsbeispiel masterseitig

```
29 int run (IBC* ibc)
30 {
31     char buff [4] = "Hi!";
32
33     Inbox *i = new Inbox(ibc->getInbox(180));
34
35     Packet p (254, 4,(uint8_t*) buff);
36     ibc->send(p);
37
38     //wait for an answer to arrive
39     std::this_thread::sleep_for(std::chrono::seconds(1));
40
41     i->fetch();
42
43     if(i->size())
44     {
45         std::cout << i->front() << '\n';
46     }
47
48     delete i;
49
50     return 0;
51 }
```

Codegenerator Beispiel slaveseitig

```
348 /* IBC_MESSAGE_BEGIN 253 16 4 */
349         case 253:
350         {
351 
352 
353 /*   Recv exactly 16 bytes in the following                              */
354 /*   Also calculate their data hash along the way by                     */
355 /*      xoring all bytes together once                                   */
356 /*      or use the provided function                                     */
357 /*   Make the hash public to the IBC by setDH(Your DATAHASH HERE)   */
358 /* IBC_PRESERVE_RECV_BEGIN 253 vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv*/
359 
360             byte buffr253[16];
361             recv(buffr253,16);
362 
363             //DONT FORGET TO HASH
364             setDH(createDH(buffr253,16));
365 
366 /* IBC_PRESERVE_RECV_END 253 ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*/
367 
368   char datahash = recv();
369   send(sstat);
370 
371 /*Send exactly 4 bytes in the following                 */
372 /*Also calculate their data hash along the way by                   */
373 /*  xoring all bytes together once                                  */
374 /*  or use the provided function createDH(..)                 */
375 /* Make the hash public to the IBC by setDH(Your DATAHASH HERE) */
376 /* IBC_PRESERVE_SEND_BEGIN 253 vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv*/
377 
378       char message [4] = "me2";
379             send(((byte*)message), 4);
380 
381             //DONT FORGET TO HASH
382             setDH(createDH(((byte*)message), 4));
383 
384 /* IBC_PRESERVE_SEND_END 253 ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^*/
385         }
386         break;
387 /* IBC_MESSAGE_END 253 16 4 */
```

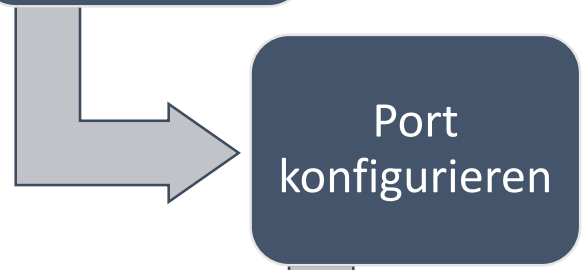
Benutzeroberfläche

Touchdisplay Raspberry Pi 3

- Display: Joy-IT TFT 3.2 Zoll
- Vorteil:
  - Kann direkt auf den GPIO‘s installiert werden
  - Gleichzeitige Verwendung von Touchdisplay und einem angeschlossenen Monitor über HDMI
- Nachteil:
  - 26 der 40 verfügbaren GPIO‘s werden durch das Display belegt

Serieller Port


```
usleep(200000);             // wait some time ... otherwise it will cause problems with the serial port
```




```
cfsetispeed(&SerialPortSettings,B115200);        // Set Read Speed as 115200
cfsetospeed(&SerialPortSettings,B115200);        // Set Write Speed as 115200
```


```
sleep(3);    // Wait until Arduino restarts
```

Warum Qt?

Plattformunabhängige GUI-Bibliothek

- Windows / Linux / OSX / iOS / Android

Kein Cross-Compiler nötig

- Einmaliges kompilieren auf der Zielplattform (Raspbian Stretch)

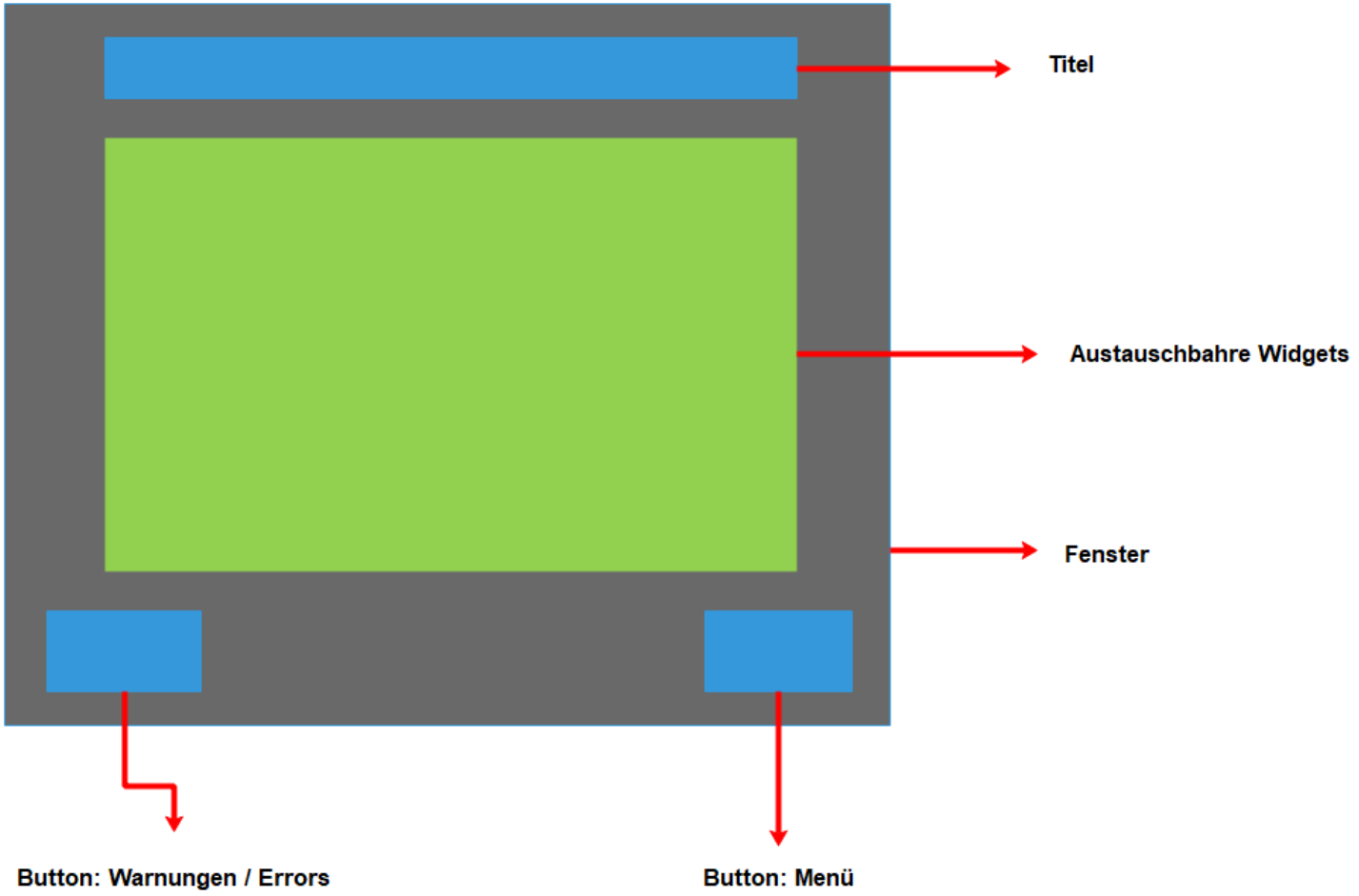
C++ / Qt Bibliotheken (nicht GUI)

- Qt bietet auch eine große Auswahl an nicht GUI Bibliotheken

Entwicklungsumgebung

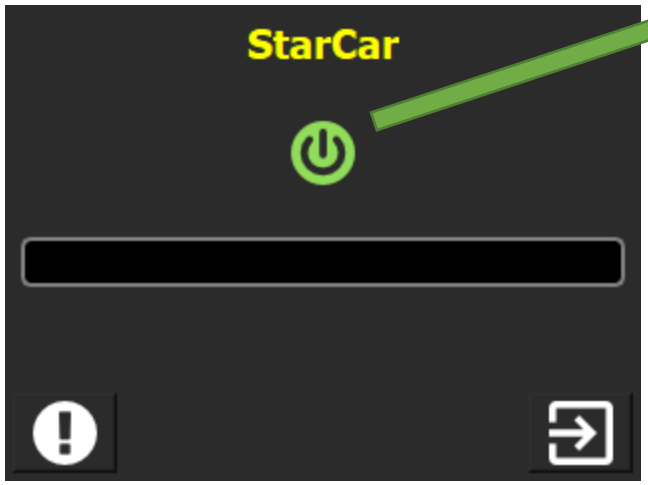
- Qt Creator (Direkte Entwicklung auf dem Raspberry möglich)
- Sehr gute Dokumentation / Große Community

Layout und genereller Aufbau



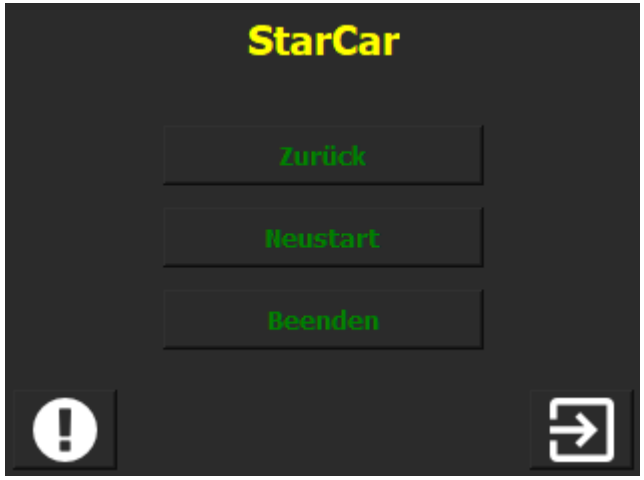
Startseite

Startet die Initialisierung



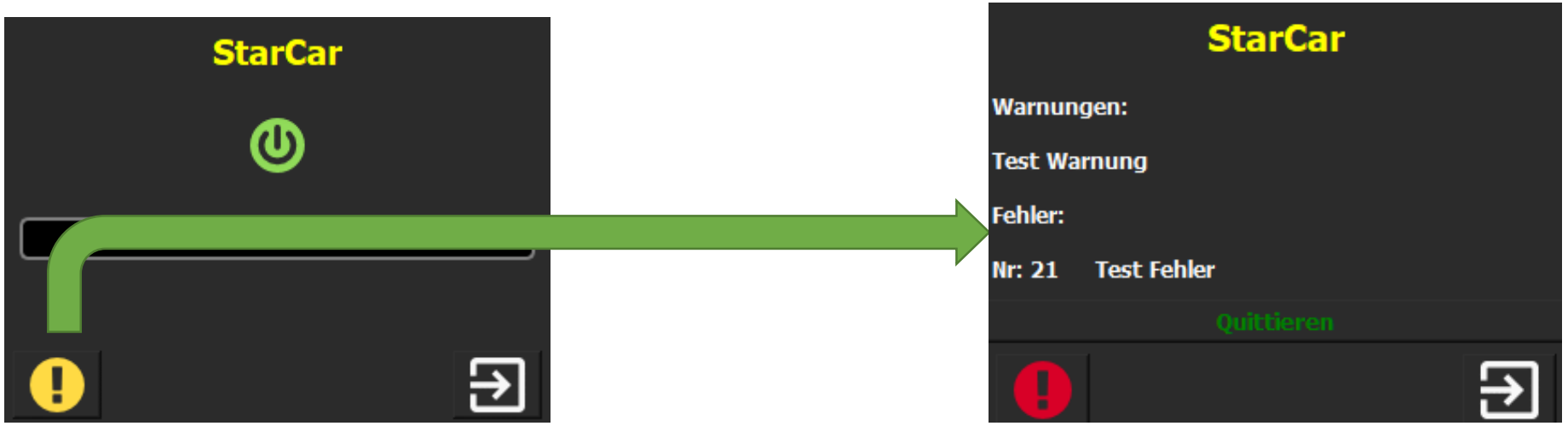

```
#ifdef Q_OS_LINUX
    *IBCPointer = new IBC("/dev/ttyUSB0","/home/pi/DT_WS1718_02_StarCar/pi/IBP/IBC_config.cfg");
#endif
```

Öffnet das Menü



- Blinkt bei Warnungen / Fehler
- Zeigt Meldungen an

Warnungen und Fehler




```
alertThread->fireError("Test Fehler",21);
alertThread->fireWarning("Test Warnung");
```

Auswahl der Modi


```
void OperationModeWidget::slotShowClockControlModeWidget(){

    emit showclockcontrollmodewidget();
}

void OperationModeWidget::slotShowControllerControlModeWidget(){

    emit showcontrollercontrolmodewidget();
}

void OperationModeWidget::slotShowSensorValuesWidget(){

    emit showsensorvalueswidget();
}
```

```
void HomeWindow::slotShowOperationModeWidget(){


    connect(operationModeWidget, SIGNAL(showclockcontrollmodewidget()), this, SLOT(slotShowClockControlModeWidget()));
    connect(operationModeWidget, SIGNAL(showcontrollercontrolmodewidget()), this, SLOT(slotShowControllerControlModeWidget()));
    connect(operationModeWidget, SIGNAL(showsensorvalueswidget()), this, SLOT(slotShowSensorValuesWidget()));

}
```

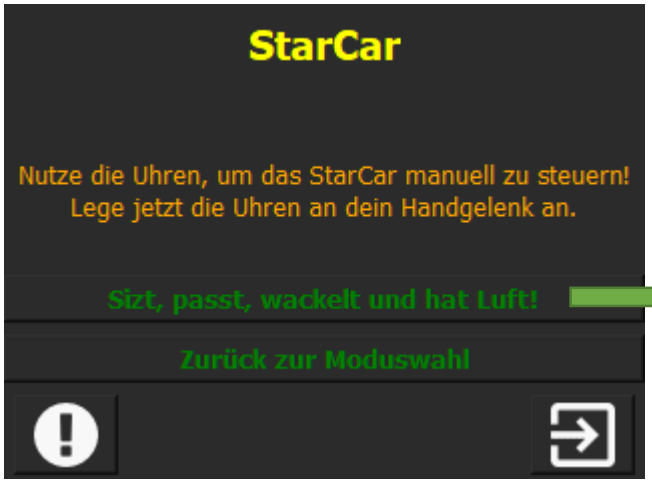
Controllersteuerung


```
#ifdef Q_OS_LINUX

    Packet ControllerPacket(100,0);
    IBCPointer->send(ControllerPacket);

#endif
```

Uhrsteuerung




```
#ifdef Q_OS_LINUX

    Packet ClockPacket(101,0);
    IBCPointer->send(ClockPacket);

#endif
```

Sensorwerte

- UltraschallVorne.txt
- UltraschallHinten.txt
- Kompass.txt
- Beschleunigung.txt
- UWB.txt
- Lidar.txt

Integration

Integration aller Komponenten

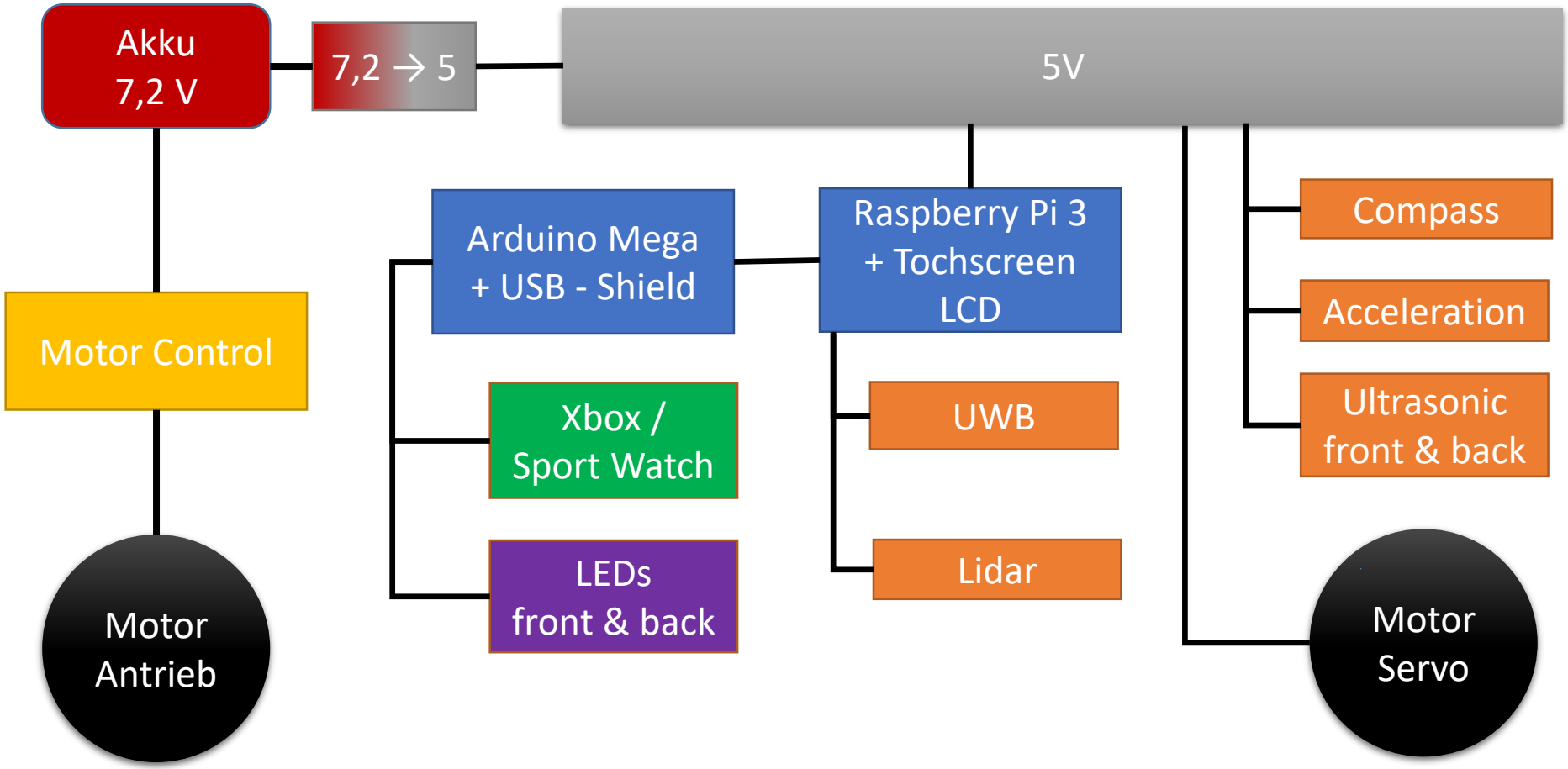
Hardware-technisch / Zusammenbau

- Aufbau StarCar:
  - Unterhalb der Spanholzplatte:
    - LEDs (front & back)
    - Akku
    - Antriebs- & Servomotor
  - Oberhalb der Spanholzplatte
    - Sensoren (UWB, Lidar, Ultrasonic)
    - Raspberry Pi 3
    - Arduino Mega
    - Motorsteuerung Antriebsmotor
    - Steckplatine
      - Sensoren (Acceleration, Compass)
      - Spannungsregler

Integration aller Komponenten

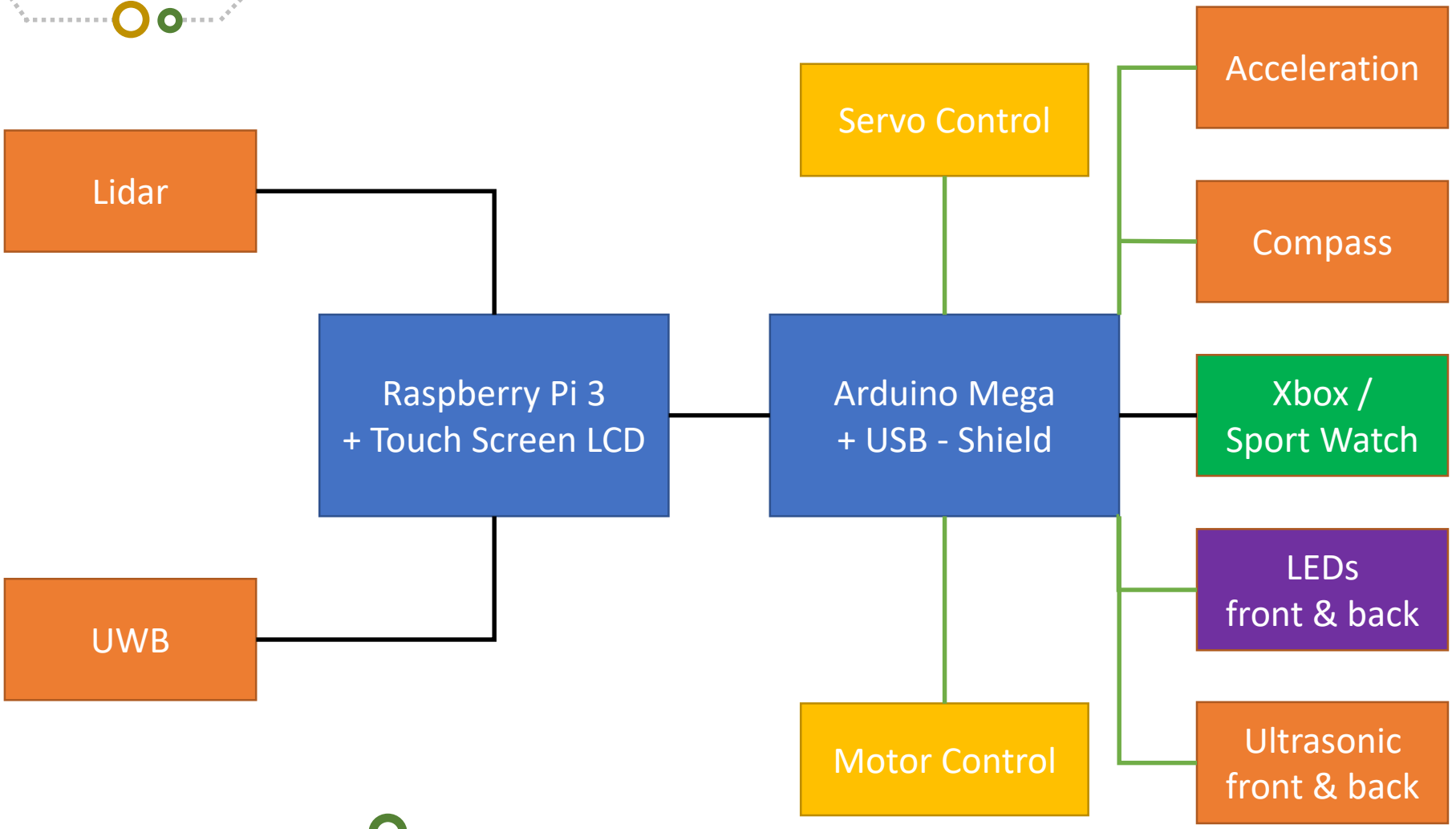
Hardware-technisch / Verkabelung Spannungsversorgung

Übersicht



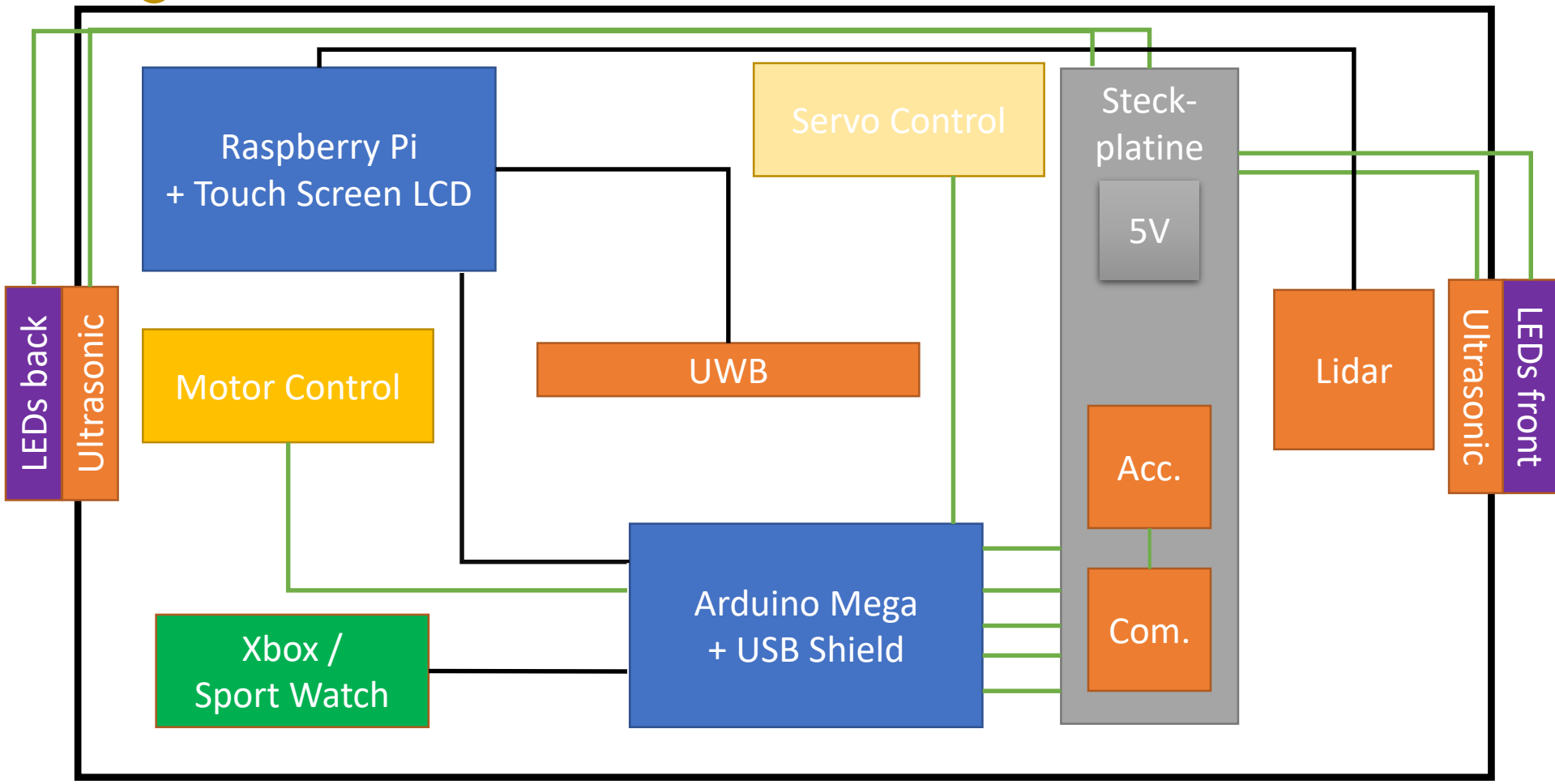
Integration aller Komponenten

Hardware-technisch / Verkabelung Daten Übersicht



Integration aller Komponenten

Hardware-technisch / Verkabelung Daten



Vielen Dank für eure Aufmerksamkeit!