Technische Universität Braunschweig

Institut für Flugführung

Serielle Kommunikation und API-Anleitung

Prof. Dr.-Ing. Peter Hecker, Dipl.-Ing. Paul Frost, Andreas Dekiert M. Sc.,
2. Juli 2019

Agenda

09. April	Einführung
16. April	Softwareprojektmanagement
23. April	Entwicklungstools
30. April	GitHub
07. Mai	Software-Dokumentation und Bug-Reporting
14. Mai	Einführung Arduino
21. Mai	Frei
28. Mai	Dateieingabe und -ausgabe
4. & 11. Juni	Tag der Lehre und Exkursionswoche
18. Juni	Einführung von Qt
25. Juni	GUI-Erstellung mit Qt
02. Juli	**Serielle Kommunikation**
09. Juli	API-Anleitung und Projektarbeit
16. Juli	Vorbereitung der Abgabe und Fragen
12. August 10:00	Abgabe

Lehrziele

Serielle Kommunikation und API-Anleitung

Als Teilnehmer soll ich am Ende dieser Übung...

- ☐ die serielle Schnittstelle vom PC ansteuern können
- ☐ den Arduino per serieller Schnittstelle steuern können
- ☐ Daten vom Arduino in einer GUI am PC anzeigen können

Serielle Schnittstelle

- Einführung
- QtSerial
- Arduino
- Bidirektionale Kommunikation

Serielle Schnittstelle

- Datenübertragung zwischen verschiedenen Geräten
- Bits werden nacheinander übertragen (seriell)
- Bekannte Standards:
  - RS-232
  - Serial ATA (SATA)
  - Universal Serial Bus (USB)

TX/RX-Ports des Arduino sind für maximal 5V ausgelegt. Bei RS232 kann die Spannung bis zu 12 V betragen.

Baudrate

- Die Baudrate gibt an, mit welcher Schrittgeschwindigkeit Daten übermittelt werden.
- Einheit: $\frac{\text{Symbol}}{\textit{Sekunde}}$
- Bei Übertragungen mit 2 Spannungen gilt die Einheit $\frac{\text{Bits}}{\text{Sekunde}}$

Baud-Rate	max. Länge
2.400	900 m
4.800	300 m
9.600	152 m
19.200	15 m
57.600	5 m
115.200	< 2 m

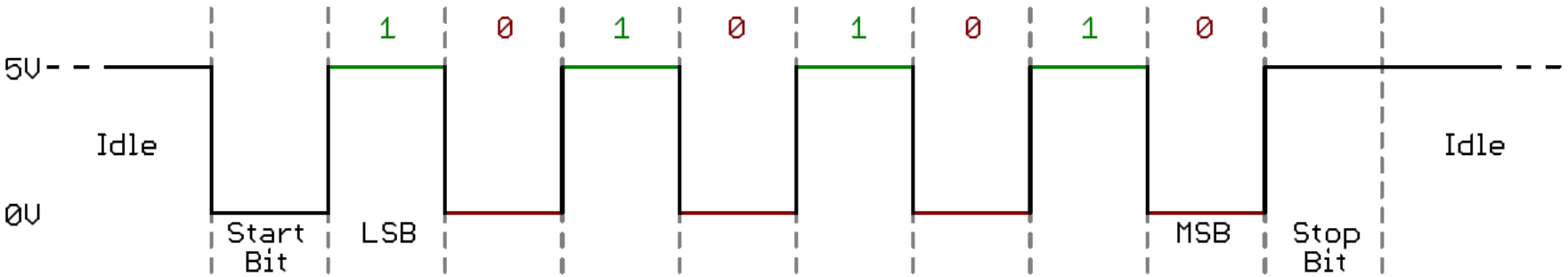
Wichtig:

Empfänger und Sender benötigen die gleiche Baudrate!

Start-/Stop-Bit

- Asynchrone Kommunikation
- Start-Bit kündigt Nachricht an
- Stop-Bit schließt Nachricht ab
- Dazwischen wird ein Datenset mit 5-9 Bits übertragen

  → Meist wird eine Länge von 8 oder 9 Bit verwendet



Paritätsbit (engl. Parity Bit)

- Das Paritätsbit folgt auf die Nachrichten-Bits
- Gibt an, ob die Anzahl der `1`-Bits gerade oder ungerade ist
- Zwei Varianten üblich:
  a) `Parity Bit = 1` wenn gerade Anzahl von `1`-Bits
  b) `Parity Bit = 1` wenn **un**gerade Anzahl von `1`-Bits
- Empfänger validiert empfangene Daten durch Zählen der `1`-Bits und Vergleich mit dem `Parity Bit`

Daten-Bits	Anzahl 1-Bits	a) Gerade	b) Ungerade
`0000 0000`	0	`0000 0000` **1**	`0000 0000` **0**
`0000 0111`	3	`0000 0111` **0**	`0000 0111` **1**
`0100 0111`	4	`0100 0111` **1**	`0100 0111` **0**

QSerialPort als Arduinoschnittstelle

- Abgeleitet aus `QIODevice` (Input/Output-Device)

→ Kann zum Lesen und Schreiben von Daten verwendet werden

- Seit Qt 5.1 in Qt eingebunden
- Verfügbar über das Modul `serialport`

```
Qt += serialport
```

Vorbedingung

- Der Rechner muss Zugang zu einer seriellen Schnittstelle haben.

  Das Arduino-Board verfügt über eine serielle Schnittstelle, welche auch per USB angesteuert werden kann.
- Der Portname dieser Schnittstelle muss identifiziert werden.

Nutzung des QSerialPort-Objekts

1. Erstellung des Objekts vom Typ `QSerialPort` unter Angabe des Serial-Portnamens
2. Öffnen der Schnittstelle über die Methode `open()`
   - `open(QIODevice::ReadOnly)`
   - `open(QIODevice::WriteOnly)`
   - `open(QIODevice::ReadWrite)`
3. Versand von Daten erfolgt über die Methode `write()`
4. Wenn Daten empfangen wurden, wird das Signal `readyRead()` emittiert
5. Über `read()` können die Daten ausgelesen werden

Die Methode `read()` sollte über eine Schleife aufgerufen werden, bis der Puffer leer ist (`bytesAvailable()`).

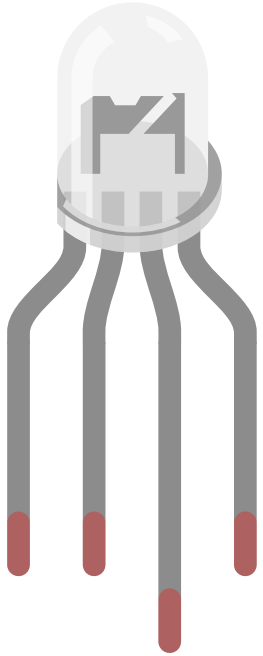
Gibt es Fragen oder Anmerkungen zu dem Unterthema
Serielle Schnittstelle in Qt?

Übung: GUI zur LED-Farbwahl

Es soll eine GUI programmiert werden, um eine an einen Arduino angeschlossene RGB-LED zu steuern.

RGB-LED

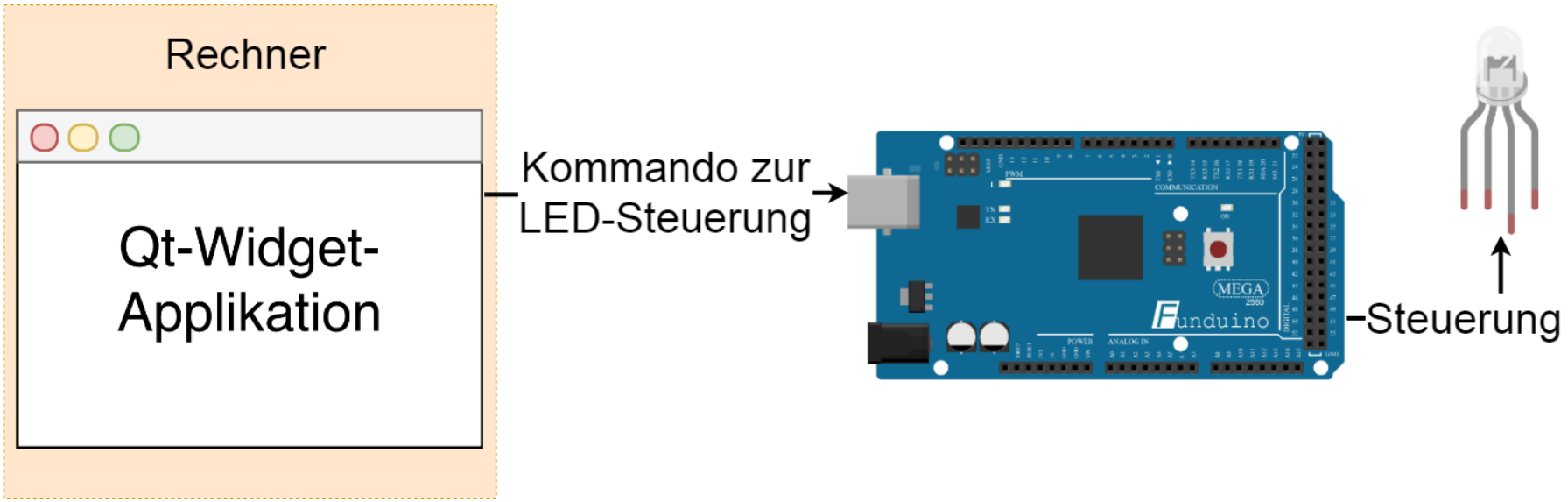
- Ein Bauteil, welches jeweils eine rote, grüne und blaue LED vereint
- Helligkeitssteuerung mittels PWM nach Farbe getrennt möglich
- Beliebige Farben sind durch Farbaddition darstellbar



Übung: GUI zur LED-Farbwahl

Es soll eine GUI programmiert werden, um eine an einen Arduino angeschlossene RGB-LED zu steuern.

Schema



Übung: GUI zur LED-Farbwahl

Es soll eine GUI programmiert werden, um eine an einen Arduino angeschlossene RGB-LED zu steuern.

Vereinbarungen

- Kommunikation zwischen GUI und Arduino erfolgt seriell
- GUI und serielle Kommunikation werden getrennt entwickelt
- Vereinbartes Signal als Schnittstelle:

```
void sendColor(char colorIdentifier, char colorValue);
```

`colorIdentifier` gibt die Farbe an (101: Rot, 102: Grün, 103: Blau)
`colorValue` gibt den Farbwert an (0 - 100)

Übung: GUI zur LED-Farbwahl

Es soll eine GUI programmiert werden, um eine an einen Arduino angeschlossene RGB-LED zu steuern.

Übertragungsprotokoll

- Sende 2 Bytes je Farbeinstellung
  `Byte 0: colorIdentifier`
  `Byte 1: colorValue`

`Byte 0`	`Byte 1`	
101	0	→ 0 % Rot
101	100	→ 100 % Rot
102	50	→ 50 % Grün

- Schnittstellenparamter

Parameter	**Einstellung**
Baudrate	9600
Datenset	8 Bits
Paritätsbit	Keins (`none`)
Stop-Bit	1 Bit
⇒	8-N-1

Praxisdemonstration: Erweiterung der GUI um serielle Kommunikation

Der Quellcode wird nach der Veranstaltung hochgeladen.

Lösungsweg: Öffnen der seriellen Schnittstelle

1. Füge in der Projektdatei das Modul `serialport` hinzu
2. Erstelle die Klasse `ArduinoInterface` (Basisklasse: `QObject`)
3. Lege die Membervariable `m_serialPort` des Typs `QSerialPort` an
4. Nehme im Konstruktor für die Instanz `m_serialPort` folgende Einstellungen vor:
   - 4.1 Stelle über die Methode `setPortName()` die richtige Portnummer ein
   - 4.2 Stelle über die Methode `setBaudRate()` die richtige Baudrate ein
   - 4.3 Stelle über die Methode `setDataBits()` die richtige Zahl Datenbits ein
   - 4.4 Stelle über die Methode `setParity()` die richtige Parität ein
   - 4.5 Stelle über die Methode `setStopBits()` die richtigen Stopbits ein
   - 4.6 Öffne über die Methode `open()` den Port zum Schreiben von Daten

Lösungsweg: Senden von Daten

1. Erstelle in der Klasse `ArduinoInterface` den Slot `sendColorToArduino()` mit zwei Eingangsparametern:
   - `colorIdentifier` des Typs **`char`**
   - `colorValue` des Typs **`char`**
2. Leite die eingegangenen Daten über die Methode `write()` an `m_serialPort` weiter
3. Damit die Daten sofort versendet werden, rufe in der Instanz die Methode `flush()` auf
4. Verbinde das Signal der GUI `sendColor()` mit dem hier angelegten Slot `sendColorToArduino()`

Gibt es Fragen oder Anmerkungen zu dem Unterthema
Serielle Schnittstelle in Qt?

Arduino Initialisierung serielle Schnittstelle

```
void setup() {
  // Beim Oeffnen wird die Baudrate festgelegt
  Serial.begin(9600);
}
```

Tabelle 1: Standardwerte

Parameter	**Einstellung**
Datenset	8 Bits
Paritätsbit	Keins (engl. none)
Stop-Bit	1 Bit
$\Rightarrow$	8-N-1

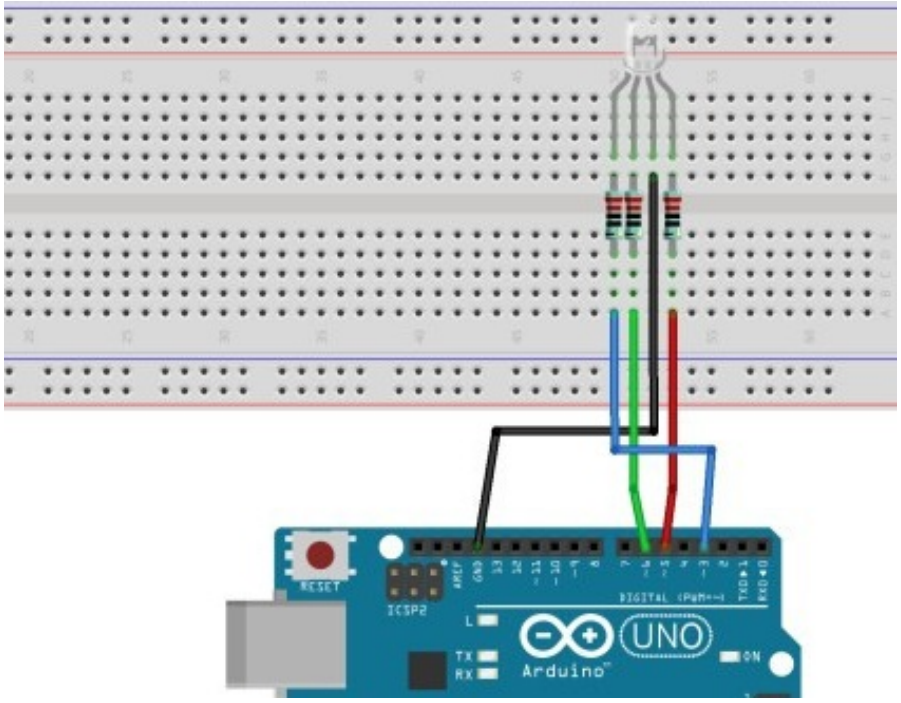
Arduino Initialisierung serielle Schnittstelle

```
// Speicher fuer das ankommende Byte
char incomingByte = '\0';

void setup() {
  // Beim Oeffnen wird die Baudrate festgelegt
  Serial.begin(9600);
}

void loop() {
  if (Serial.available() > 0){
    incomingByte = Serial.read();

    // Zuruecksenden des empfangenen bytes
    Serial.write(incomingByte);
  }
}
```

Übung: GUI zur LED-Farbwahl

Es soll eine GUI programmiert werden, um eine an einen Arduino angeschlossene RGB-LED zu steuern.

Arduino-Schaltplan



Vorwiderstände:
100 Ω für Blau
200 Ω für Grün und Rot

Übung: GUI zur LED-Farbwahl

Es soll eine GUI programmiert werden, um eine an einen Arduino angeschlossene RGB-LED zu steuern.

Übertragungsprotokoll

- Sende 2 Bytes je Farbeinstellung
  `Byte 0: colorIdentifier`
  `Byte 1: colorValue`

`Byte 0`	`Byte 1`	
101	0	→ 0 % Rot
101	100	→ 100 % Rot
102	50	→ 50 % Grün

- Schnittstellenparamter

Parameter	**Einstellung**
Baudrate	9600
Datenset	8 Bits
Paritätsbit	Keins (`none`)
Stop-Bit	1 Bit
⇒	8-N-1

Praxisdemonstration: Arduino-Programm
Der Quellcode wird nach der Veranstaltung hochgeladen.

Lösungsweg: LED-Steuerung per Arduino

1. Erstelle ein Arduino-Projekt
2. Nutze die `setup()`-Funktion, um die Pins für die RGB-LED einzustellen
3. Richte die serielle Schnittstelle mit der passenden Baudrate ein
4. Lese mit jedem Durchgang - sofern vorhanden - ein Byte aus
5. Für die Werte 101 - 103 soll die aktive Farbe geändert werden

   101 $\rightarrow$ rot
   102 $\rightarrow$ grün
   103 $\rightarrow$ blau
6. Die Werte 0 - 100 sollen auf den Bereich 0 - 255 interpoliert und der aktiven LED zugewiesen werden

Abgehakt

Serielle Kommunikation und API-Anleitung

Als Teilnehmer soll ich am Ende dieser Übung...

- [x] die serielle Schnittstelle vom PC ansteuern können
- [x] den Arduino per serieller Schnittstelle steuern können
- [ ] Daten vom Arduino in einer GUI am PC anzeigen können

Übung: Erweiterung GUI um Message Counter

Die GUI soll um einen Zähler der an den Arduino gesendeten und vom Arduino empfangenen Nachrichten erweitert werden.

Vereinbarungen

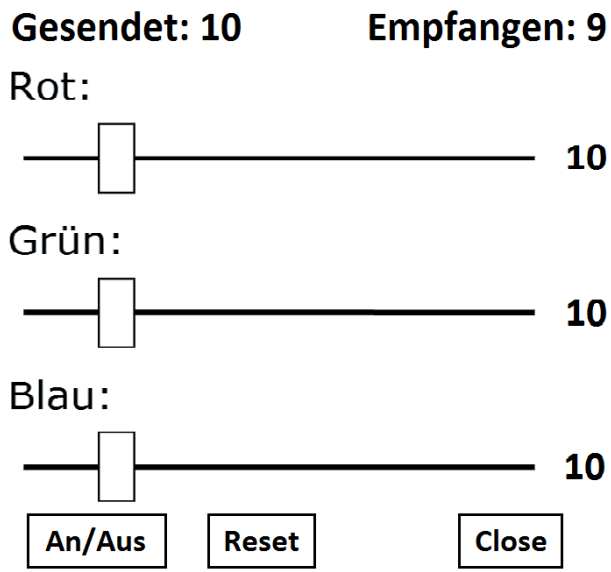
- Definiere GUI-Slots als Schnittstelle

```
void updateCntSend(int iCount);
void updateCntReceived(int iCount);
```

  `iCount` Anzahl der gesendeten bzw. vom Arduino epmfangenen Nachrichten

- Arduino sendet bei Empfang einer Nachricht, Gesamtzahl der empfangenen Nachrichten als `int` (2-Byte) an GUI.

GUI-Skizze



Praxisdemonstration: Erweiterung um Message Counter
Der Quellcode wird nach der Veranstaltung hochgeladen.

Gibt es Fragen oder Anmerkungen zu dem Thema
Serielle Schnittstelle?

Abgehakt

Serielle Kommunikation und API-Anleitung

Als Teilnehmer soll ich am Ende dieser Übung...

- ☑ die serielle Schnittstelle vom PC ansteuern können
- ☑ den Arduino per serieller Schnittstelle steuern können
- ☑ Daten vom Arduino in einer GUI am PC anzeigen können

Ende

Vielen Dank für eure Aufmerksamkeit!