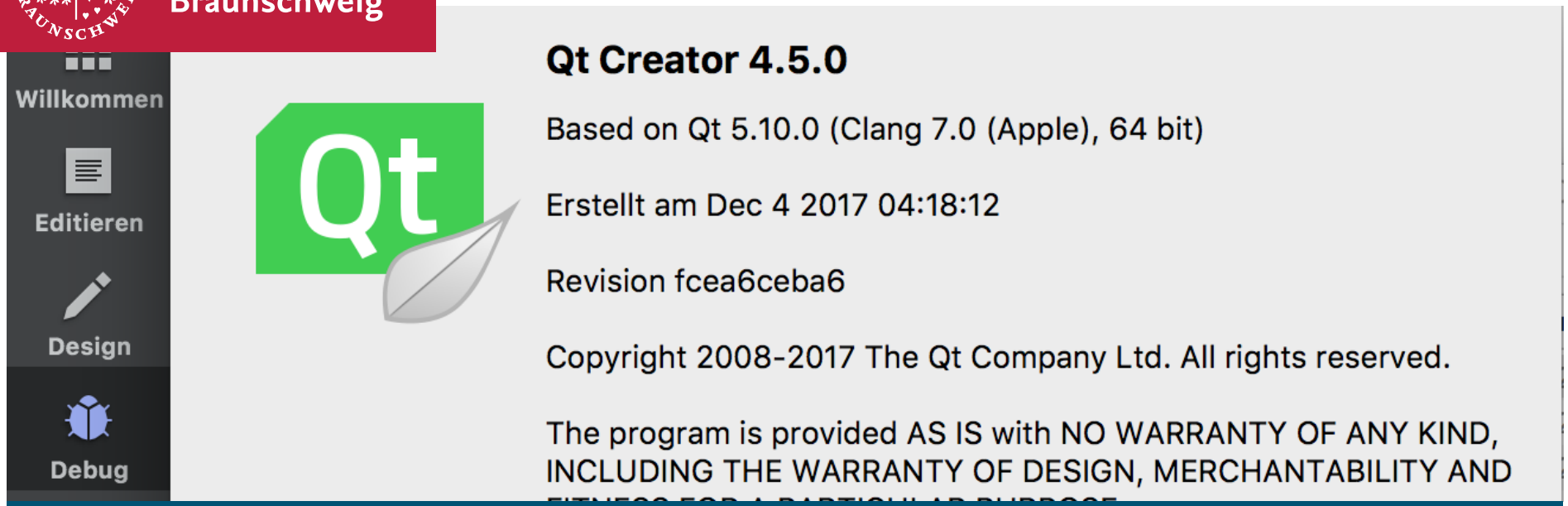
Technische Universität Braunschweig

Institut für Flugführung



GUI-Erstellung mit Qt

Prof. Dr.-Ing. Peter Hecker, Dipl.-Ing. Paul Frost, Andreas Dekiert M. Sc.,
12. Juni 2018

Agenda

03. April	Einführung
10. April	Softwareprojektmanagement
17. April	Entwicklungstools
24. April	GitHub
08. Mai	Einführung Arduino/Funduino
15. Mai	Dateieingabe und -ausgabe
22. Mai	Exkursionswoche
29. Mai	Dokumentation und Bug-Reporting
05. Juni	Einführung von Qt
12. Juni	**GUI-Erstellung und serielle Kommunikation mit Qt**
19. Juni	Anleitung erstellen
26. Juni	Projektarbeit
03. Juli	Vorbereitung der Abgabe
10. Juli	Abgabe

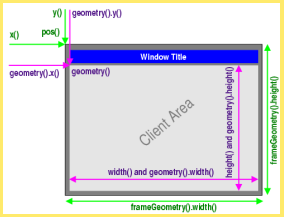
Lehrziele

GUI-Erstellung mit Qt

Als Teilnehmer soll ich am Ende dieser Übung...

- ☐ die QWidget-Klasse kennen
- ☐ GUIs in Qt erstellen können
- ☐ wissen, wie die serielle Schnittstelle angesteuert wird
- ☐ Daten an den Arduino schicken können

GUI



GUI mit Qt

RGB-GUI

Qt GUI/Widgets Module

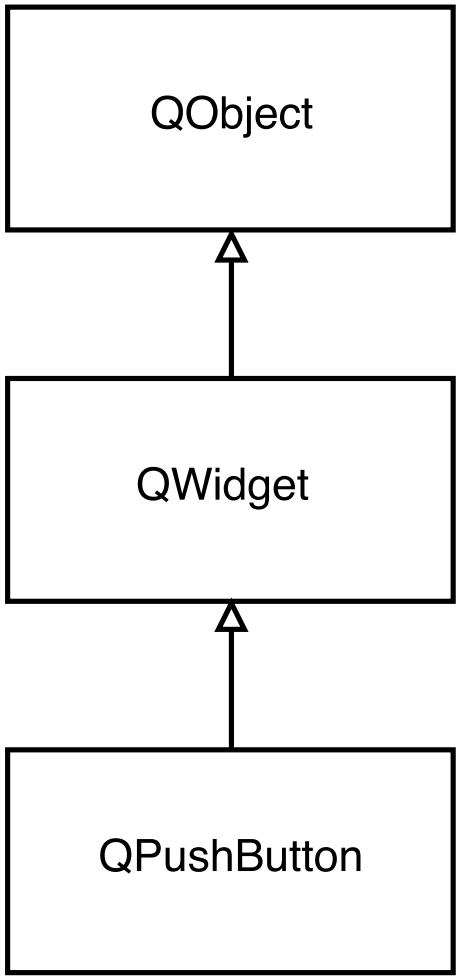
- Qt besitzt eigene Module für GUIs
  - Qt GUI
    Zentrales Modul für graphische Elemente
  - Qt Widgets
    High-Level Objekte, wie z. B. Fenster, Slider und Buttons
- Diese Module enthalten Klassen für eine plattformübergreifende GUI-Programmierung
- Module werden automatisch bei der Erstellung einer Qt-Widgets-Applikation eingebunden

Bei der Konsolenapplikation werden die Module nicht geladen!

Qt-Widgets

- Elementare Bausteine für alle GUI-Elemente
- Jedes darstellbare GUI-Element ist von der Klasse `QWidget` abgeleitet
- `QWidget` Klassen sind wiederum von der `QObjekt` Klasse abgeleitet

⇒ Verwendung von Signals und Slots möglich



Widgets

Geometrie

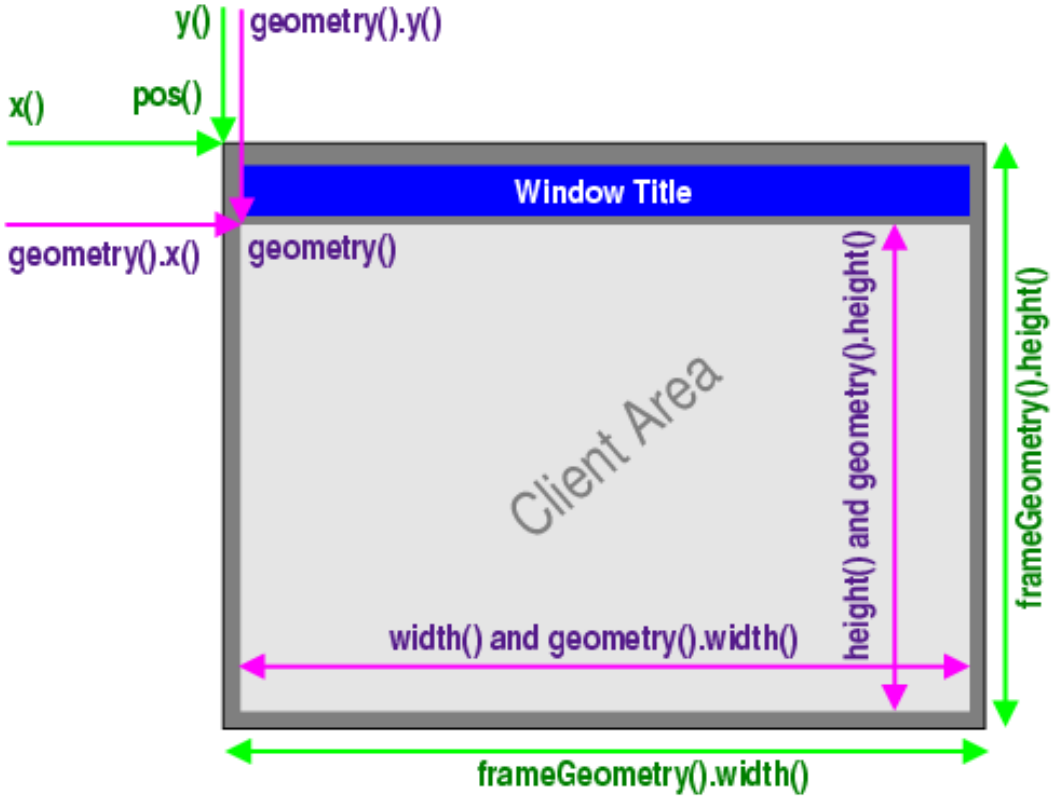


Abbildung 1: Übersicht der Geometriefunktionen für eine `QWidget`-Klasse[1]

[1] http://doc.qt.io/qt-5/application-windows.html

Widgets

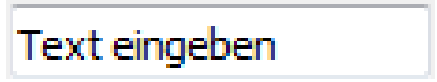
Übersicht

- `QPushButton`

Widgets

Übersicht

- `QPushButton`
- `QLineEdit`



Widgets

Übersicht

- `QPushButton`
- `QLineEdit`
- `QComboBox`

Widgets Übersicht

- `QPushButton`
- `QLineEdit`
- `QComboBox`
- `QSpinBox`

Widgets

Übersicht

- `QPushButton`
- `QLineEdit`
- `QComboBox`
- `QSpinBox`
- `QCheckBox`

Widgets

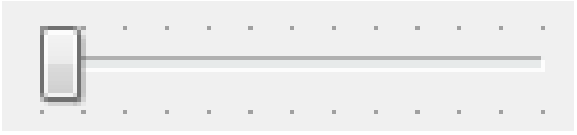
Übersicht

- QPushButton
- QLineEdit
- QComboBox
- QSpinBox
- QCheckBox
- QRadioButton

Widgets

Übersicht

- QPushButton
- QLineEdit
- QComboBox
- QSpinBox
- QCheckBox
- QRadioButton
- QSlider

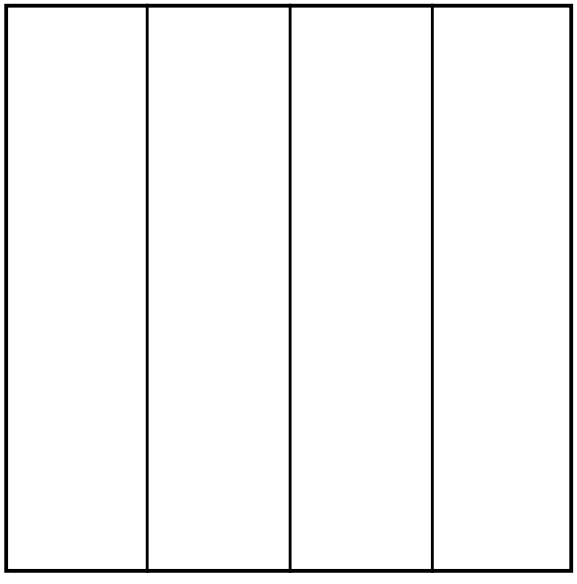


Layout

Übersicht

- Layouts positionieren die enthaltenen Widgets
- Neben Widgets können auch Layouts hinzugefügt werden
  $\rightarrow$ Verschachtelung möglich

- `QHBoxLayout`

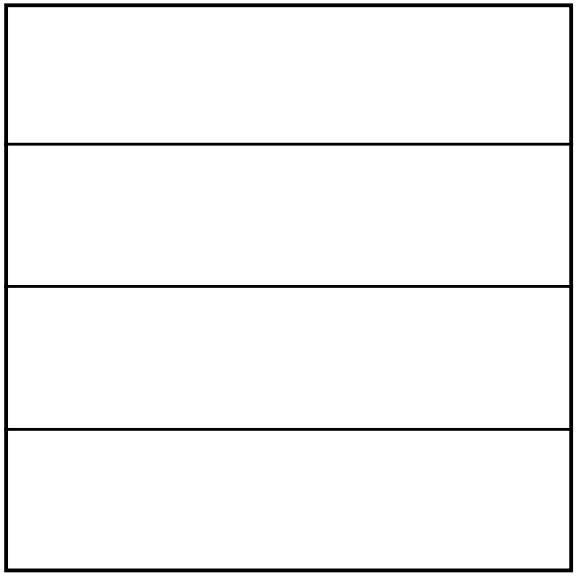


Layout

Übersicht

- Layouts positionieren die enthaltenen Widgets
- Neben Widgets können auch Layouts hinzugefügt werden
  $\rightarrow$ Verschachtelung möglich

- `QHBoxLayout`
- `QVBoxLayout`

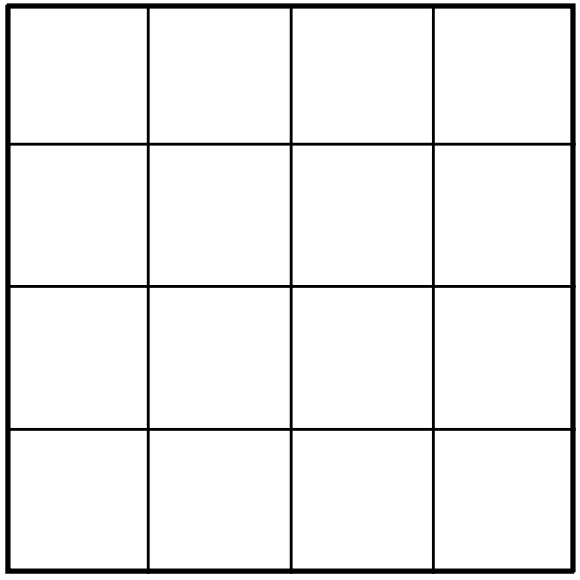


Layout

Übersicht

- Layouts positionieren die enthaltenen Widgets
- Neben Widgets können auch Layouts hinzugefügt werden
  $\rightarrow$ Verschachtelung möglich

- `QHBoxLayout`
- `QVBoxLayout`
- `QGridLayout`

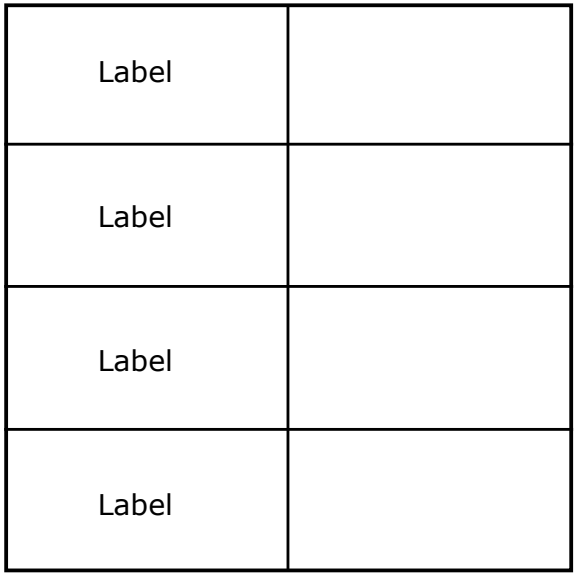


Layout

Übersicht

- Layouts positionieren die enthaltenen Widgets
- Neben Widgets können auch Layouts hinzugefügt werden
  → Verschachtelung möglich

- `QHBoxLayout`
- `QVBoxLayout`
- `QGridLayout`
- `QFormLayout`



Gibt es Fragen oder Anmerkungen zu dem Unterthema
GUI mit Qt - Einführung und Elemente?

Abgehakt

GUI-Erstellung mit Qt

Als Teilnehmer soll ich am Ende dieser Übung...

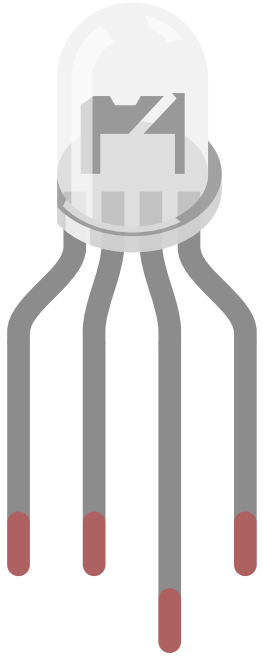
- [x] die QWidget-Klasse kennen
- [ ] GUIs in Qt erstellen können
- [ ] wissen, wie die serielle Schnittstelle angesteuert wird
- [ ] Daten an den Arduino schicken können

Übung: GUI zur LED-Farbwahl

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

RGB-LED

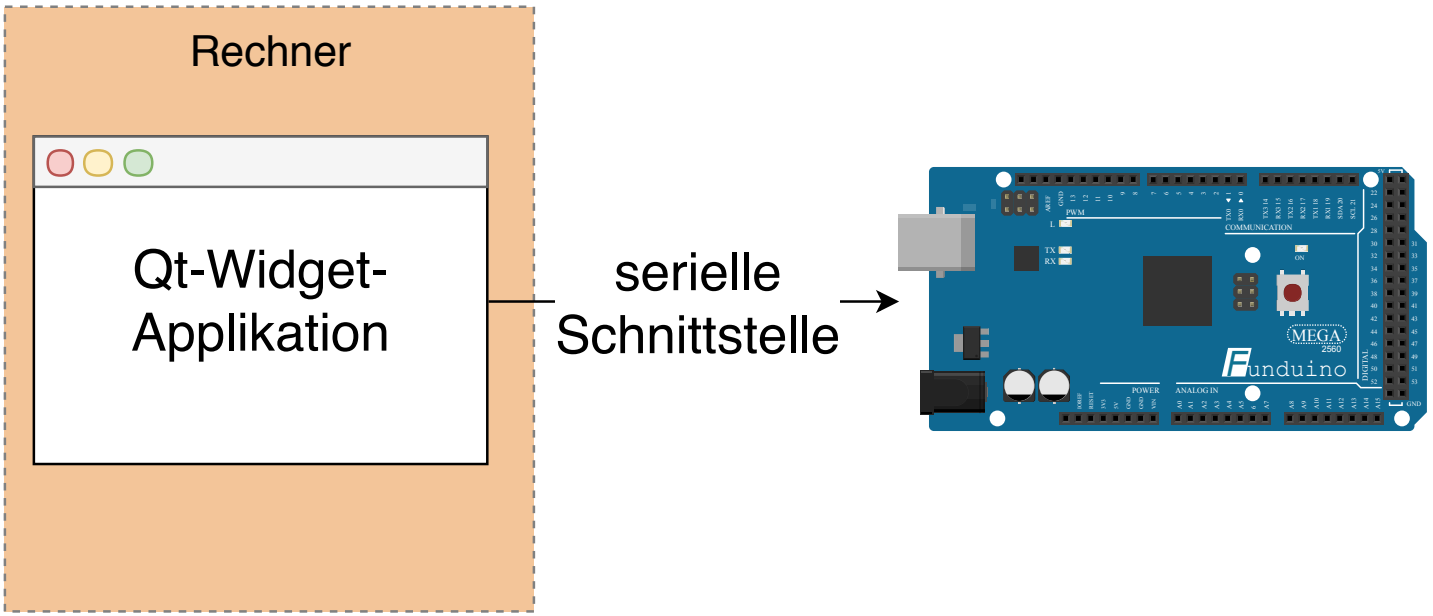
- Ein Bauteil, welches jeweils eine rote, grüne und blaue LED vereint
- Helligkeitssteuerung mittels PWM nach Farbe getrennt möglich
- Beliebige Farben sind durch Farbaddition darstellbar



Übung: GUI zur LED-Farbwahl

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

Schema



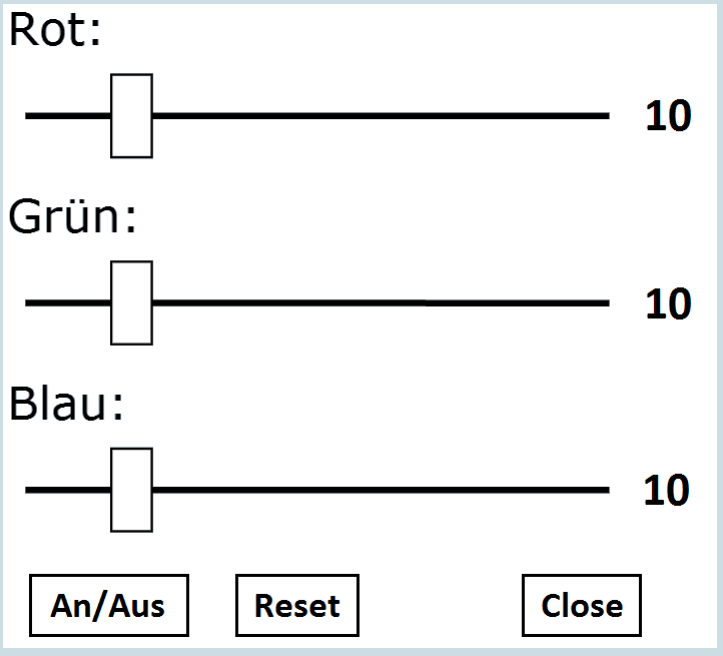
Übung: GUI zur LED-Farbwahl

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

Anforderungen an die GUI

- Steuerung der drei LED-Farben mittels je eines Schiebereglers
- Wertebereich: 0-100
- Knöpfe, um die LED ein- oder auszuschalten, den Anfangszustand wiederherzustellen und das Programm zu beenden

Skizze



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);
```

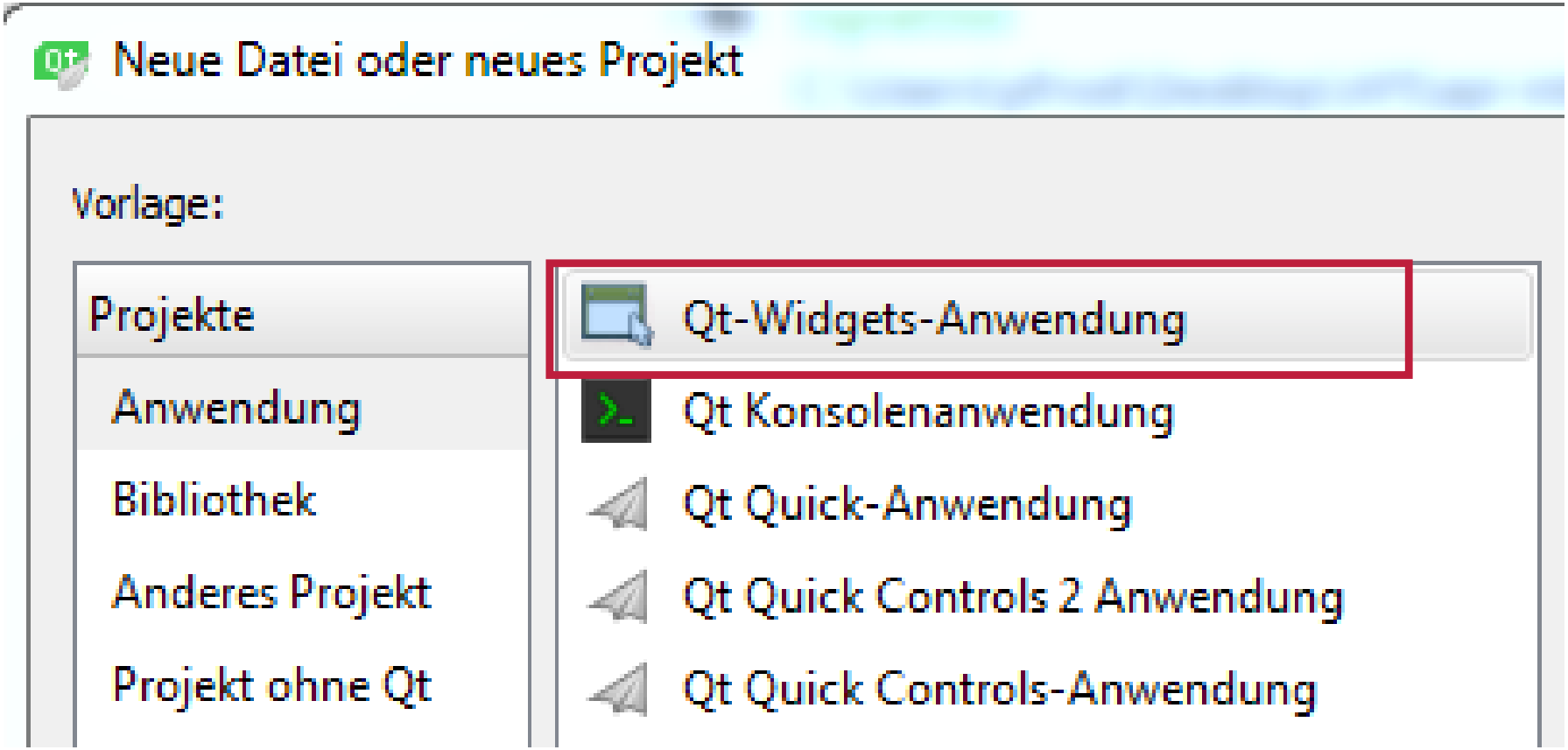
`colorIdentifier` gibt die Farbe an (101: Rot, 102: Grün, 103: Blau)
`colorValue` gibt den Farbwert an (0 - 100)

Graphische Benutzeroberfläche erstellen

1. Qt-Projekt erstellen (Qt-Widgets-Anwendung)
2. GUI-Klasse auswählen
3. Layout im Designer erstellen
4. GUI in Quelltext einbinden

Übung

Projekt erstellen



Übung

Klasse erstellen

Parameter der Klasse

Geben Sie Informationen bezüglich der Klassen ein, für die Sie Quelltexte generieren wollen.

Klassenname:	Widget
Basisklasse:	QWidget
Header-Datei:	widget.h
Quelldatei:	widget.cpp
Form-Datei generieren:	☑
Form-Datei:	widget.ui

Übung

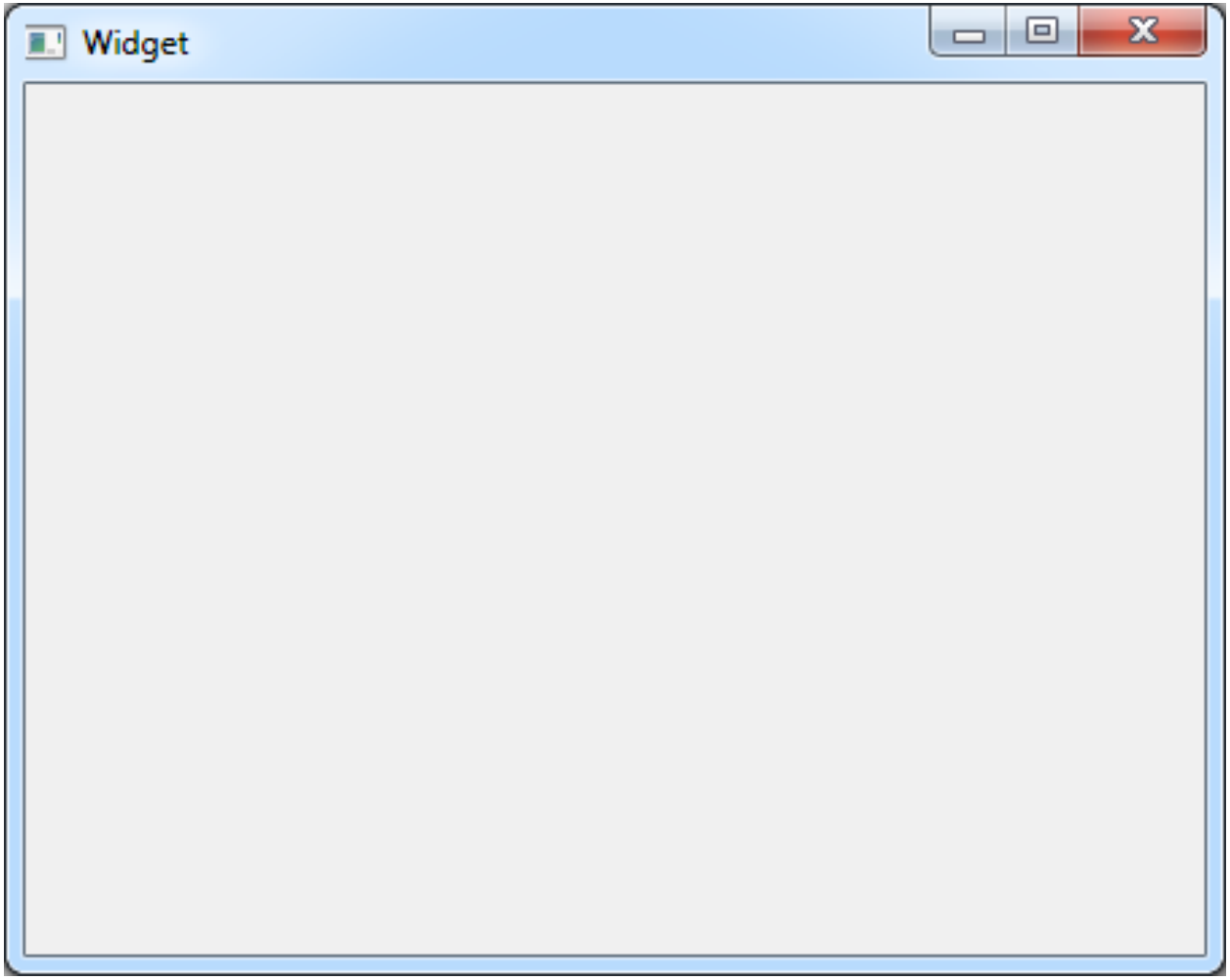
Quelltext

- Quelltext wird automatisch angelegt
- Zugriff auf GUI-Elemente über das Objekt `ui` möglich

```
#include <QWidget>

namespace Ui {
class Widget;
}

class Widget : public QWidget
{
   Q_OBJECT

private:
   Ui::Widget *ui;
```

Übung

Programm starten

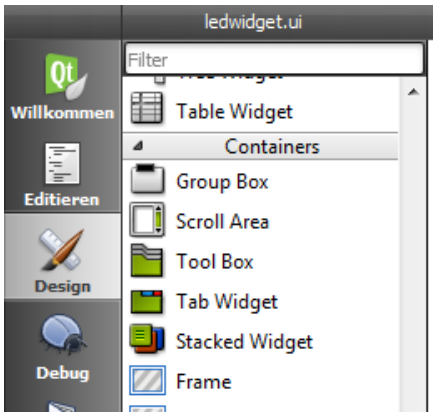


Übung

Benötigte Elemente

Qt Designer

Elemente können über den Qt Designer hinzugefügt werden

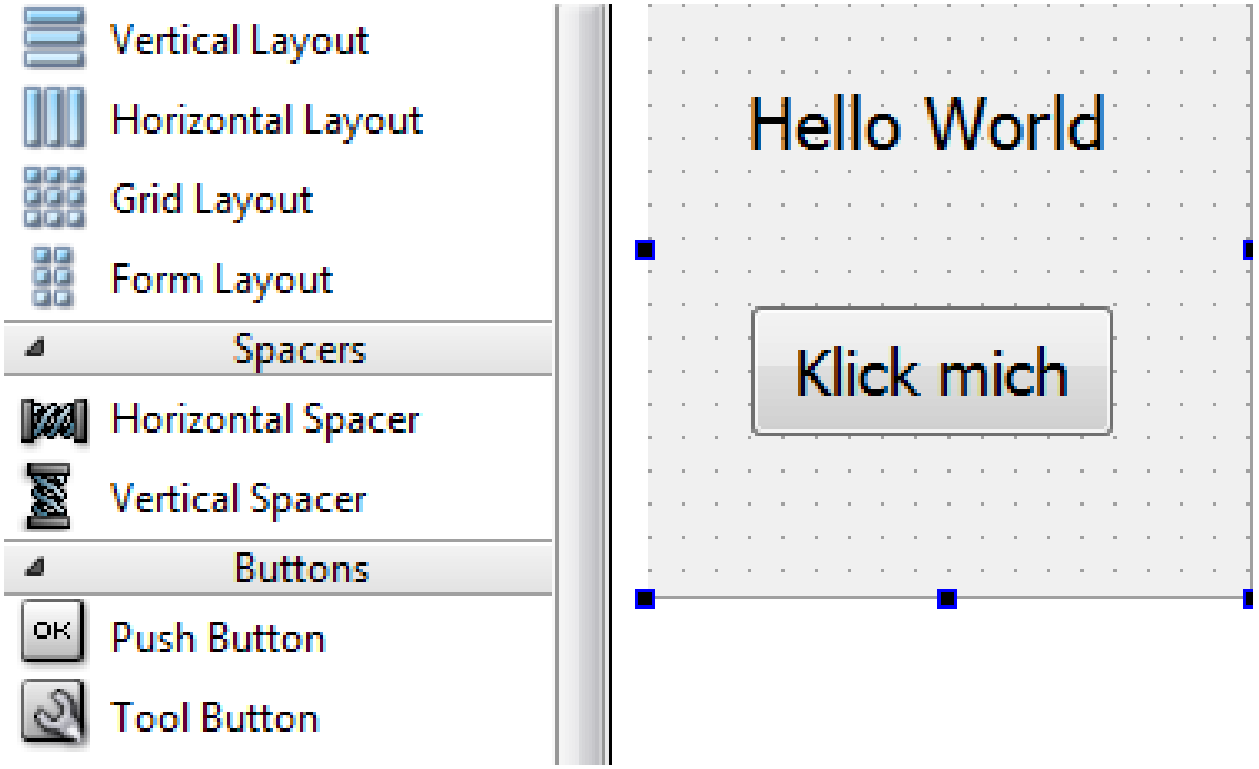


Elemente

`QWidget`	Fensterelement
`QPushButton`	Schaltfläche in Qt
`QLabel`	Element für Beschriftungen
`QSlider`	Schieberegler
`QLineEdit`	Einzeilige Textbox
`QGridLayout`	Rasterlayout
`QHBoxLayout`	Horizontales Layout
`Spacer`	Puffer zum „Auffüllen“ eines Layouts

Übung Layout im Designer erstellen

- Das Zusammenstellen der GUI erfolgt per *Drag and Drop*
- Widgets können auch nächträglich in Layouts einsortiert werden



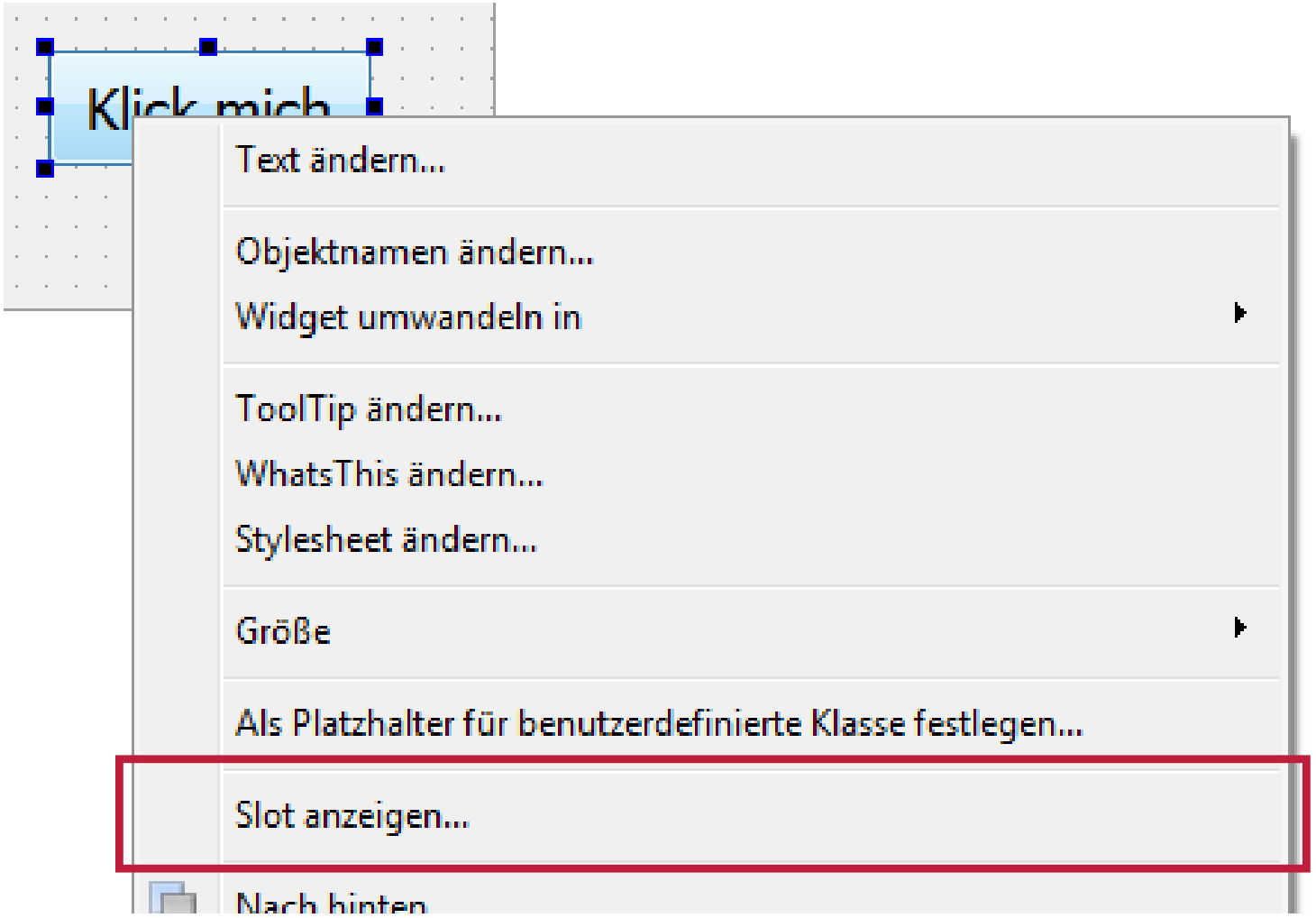
Übung Objektnamen anpassen

- Anordnung der Eigenschaften nach Klassenhierarchie

Eigenschaft	Wert
QObject	
objectName	hwButton
QWidget	
QAbstractButton	
text	Klick mich
icon	
iconSize	16 x 16
shortcut	
checkable	☐

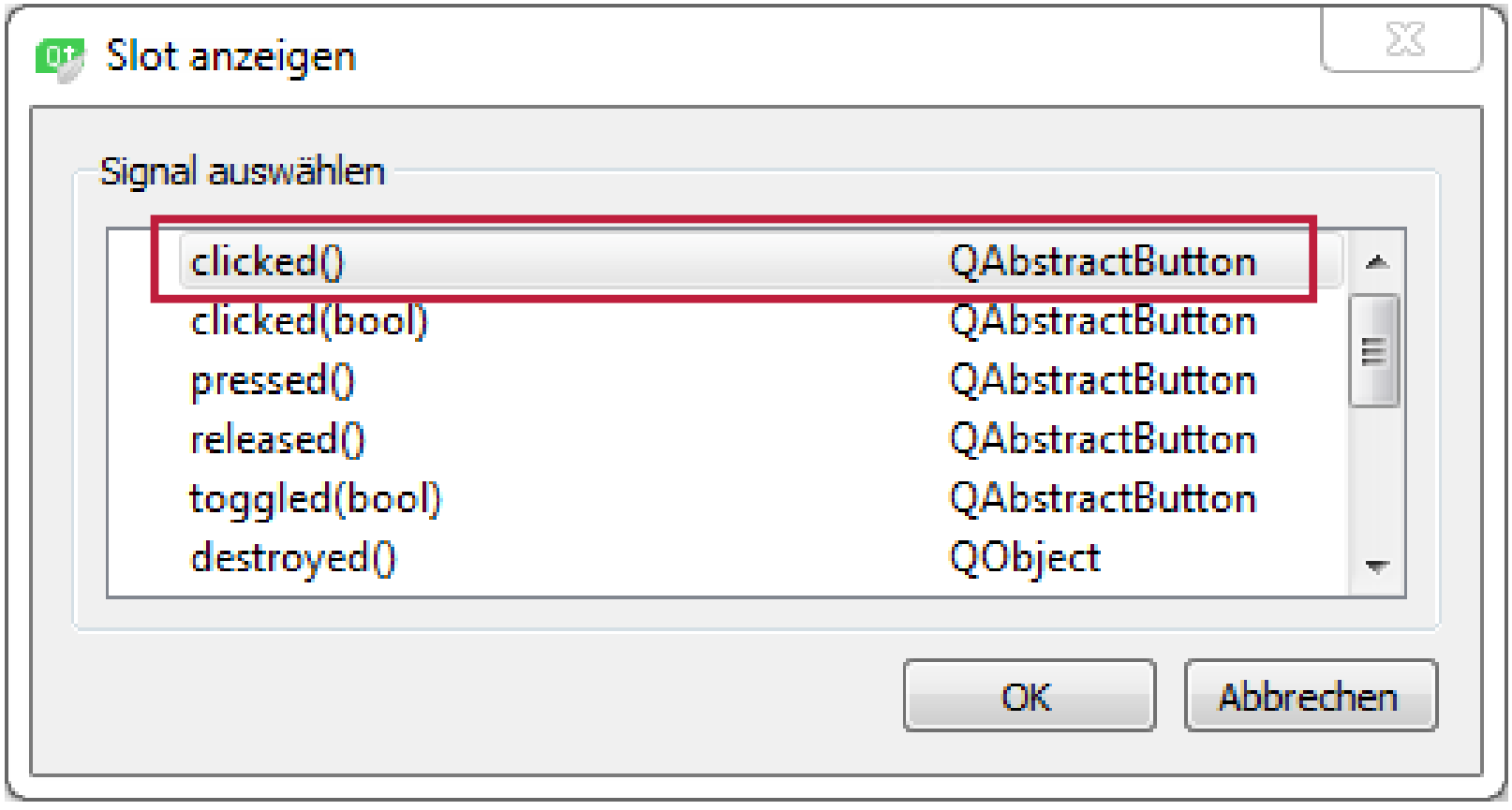
Übung

Slot erstellen



Übung

Slot erstellen



Übung

Quelltext

- Slot wird automatisch über den Dialog „Slot anzeigen“ angelegt
- Über den richtigen Namen wird der Slot automatisch mit dem Signal verbunden

  `on_<ObjektnameDesElements>_clicked()`

```
void Widget::on_hwButton_clicked()
{
   ui->hwLabel->setText("Hallo World");
}
```

Gibt es Fragen oder Anmerkungen zu dem Thema **GUI**?

Abgehakt

GUI-Erstellung mit Qt

Als Teilnehmer soll ich am Ende dieser Übung...

- [x] die QWidget-Klasse kennen
- [x] GUIs in Qt erstellen können
- [ ] wissen, wie die serielle Schnittstelle angesteuert wird
- [ ] Daten an den Arduino schicken können

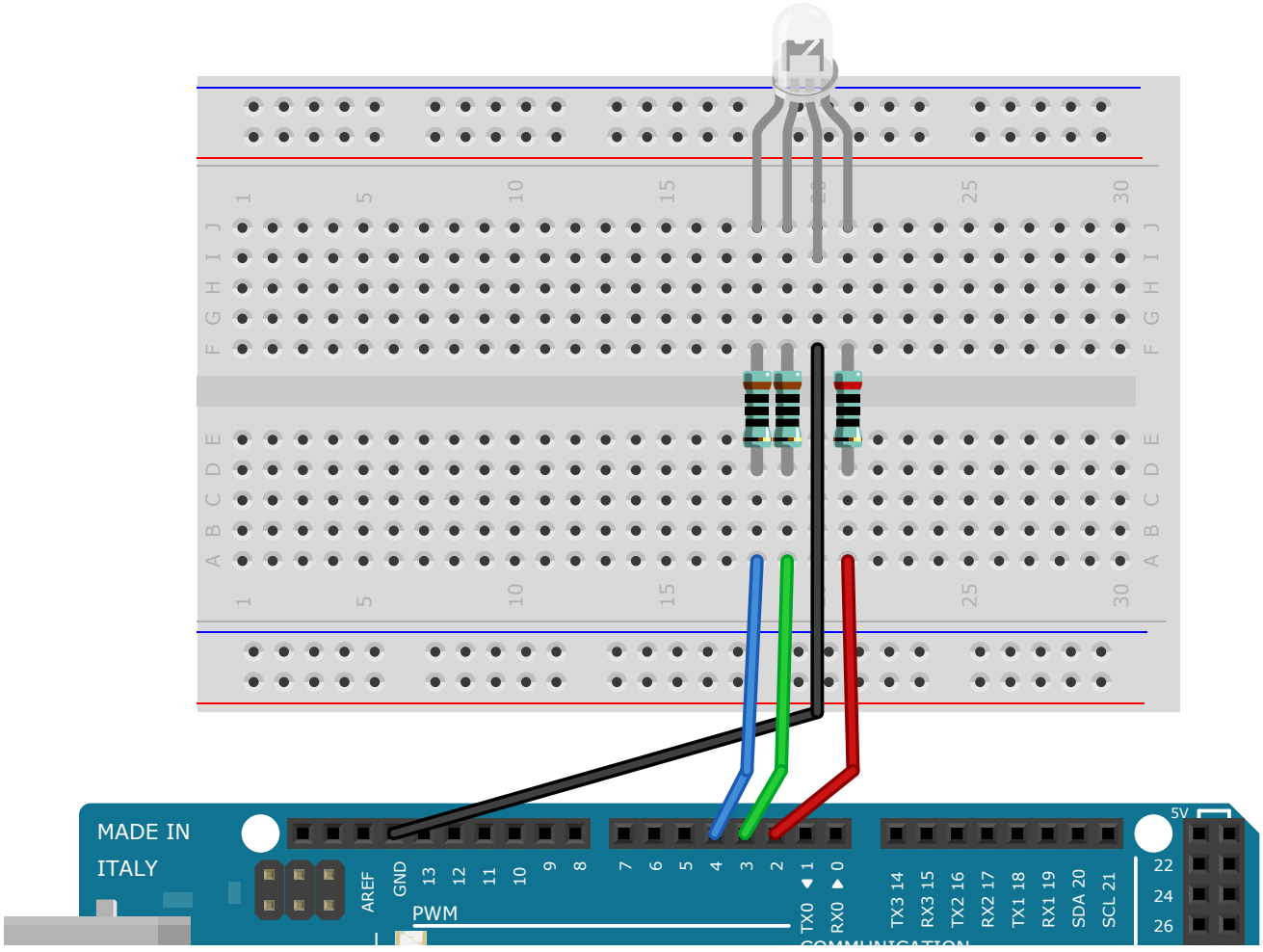
LED-Farbwahl

Qt-Designer

Praxisdemonstration Qt-Designer

LED-Farbwahl

Arduino-Aufbau



LED-Farbwahl Arduino-Programmierung

Praxisdemonstration Arduino-Programmierung

LED-Farbwahl Qt-Schnittstelle

Praxisdemonstration Qt-Schnittstelle

Gibt es Fragen oder Anmerkungen zu dem Thema **GUI**?

Abgehakt

GUI-Erstellung mit Qt

Als Teilnehmer soll ich am Ende dieser Übung...

- [x] die QWidget-Klasse kennen
- [x] GUIs in Qt erstellen können
- [ ] wissen, wie die serielle Schnittstelle angesteuert wird
- [ ] Daten an den Arduino schicken können

Serielle Schnittstelle

Einführung

QtSerial

Arduino

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 $\frac{\text{Bits}}{\text{Sekunde}}$

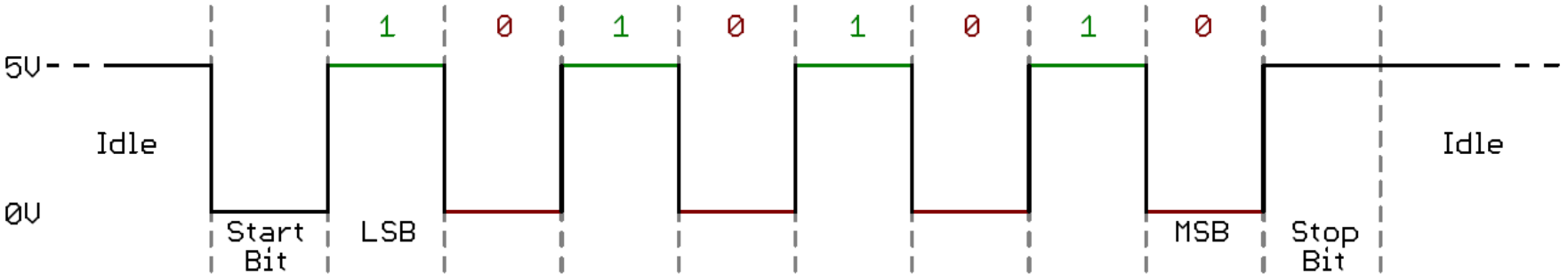
Baud-Rate	max. Länge
2.400	900 m
4.800	300 m
9.600	152 m
19.200	15 m
57.600	5 m
115.200	< 2 m

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
- Häufig wird eine Länge von 8 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**

Gibt es Fragen oder Anmerkungen zu dem Unterthema
Serielle Schnittstelle?

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 serial

```
Qt += serial
```

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 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.

Übung: Öffnen der seriellen Schnittstelle

1. Füge in der Projektdatei das Modul serial 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 Öffne über die Methode `open()` den Port zum Schreiben von Daten

Übung: Schreiben von Daten

1. Erstelle in der Klasse ArduinoInterface den Slot `writeData()` mit einem Eingangsparameter:
   - `dataToWrite` des Typs `const 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 `colorChanged()` mit dem hier angelegten Slot `writeData()`

Gibt es Fragen oder Anmerkungen zu dem Unterthema
QtSerial?

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 chars
    Serial.print(incomingByte);
  }
}
```

LED-Farbwahl

Übertragungsprotokoll

- Übertragung erfolgt seriell
- 0-100 Sind reserviert für den Farbwertbereich
- 101-103 Wählen die Farbe aus

  101 Rot
  102 Grün
  103 Blau
- 2 Bytes werden für eine Farbeinstellung gesendet

	Byte 0	Byte 1	
Wert	101	0	→ 0 % Rot
Wert	101	100	→ 100 % Rot
Wert	102	50	→ 50 % Grün
Wert	103	20	→ 20 % Blau

Übung: LED-Einstellung 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 ein Byte aus
5. Für die Werte 101 - 103 soll die aktive Farbe geändert werden
   - 101 → rot
   - 102 → grün
   - 103 → blau
6. Die Werte 0 - 100 sollen auf den Bereich 0 - 255 interpoliert und der aktiven LED zugewiesen werden

Gibt es Fragen oder Anmerkungen zu dem Thema
Serielle Schnittstelle?

Abgehakt

GUI-Erstellung mit Qt

Als Teilnehmer soll ich am Ende dieser Übung...

- ☑ die QWidget-Klasse kennen
- ☑ GUIs in Qt erstellen können
- ☑ wissen, wie die serielle Schnittstelle angesteuert wird
- ☑ Daten an den Arduino schicken können

Sprintmeeting 15 Min

Jetzt besteht die Möglichkeit, das Sprintmeeting durchzuführen.

Protokolliert bitte

- die bearbeiteten Aufgaben der Vorwoche.
- die Zwischenstände der geplanten Aufgaben.
- die in der kommenden Woche zu bearbeitenden Aufgaben.

Ende

Vielen Dank für eure Aufmerksamkeit!