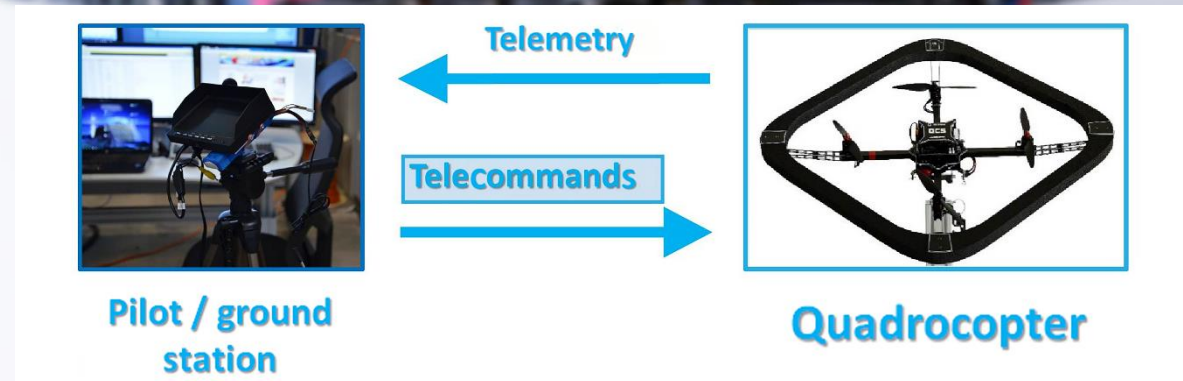


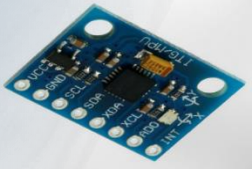
Communication

Telecommands

Lesson 6

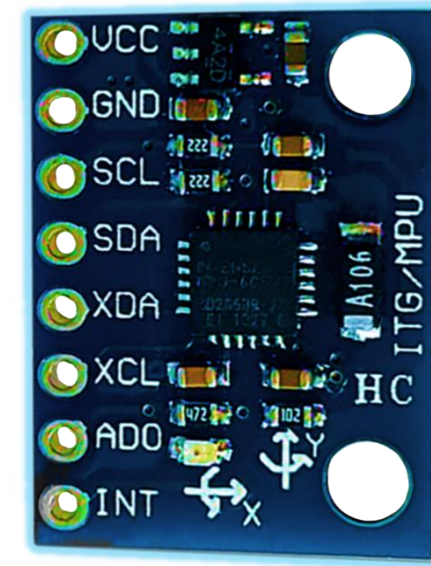


Contents

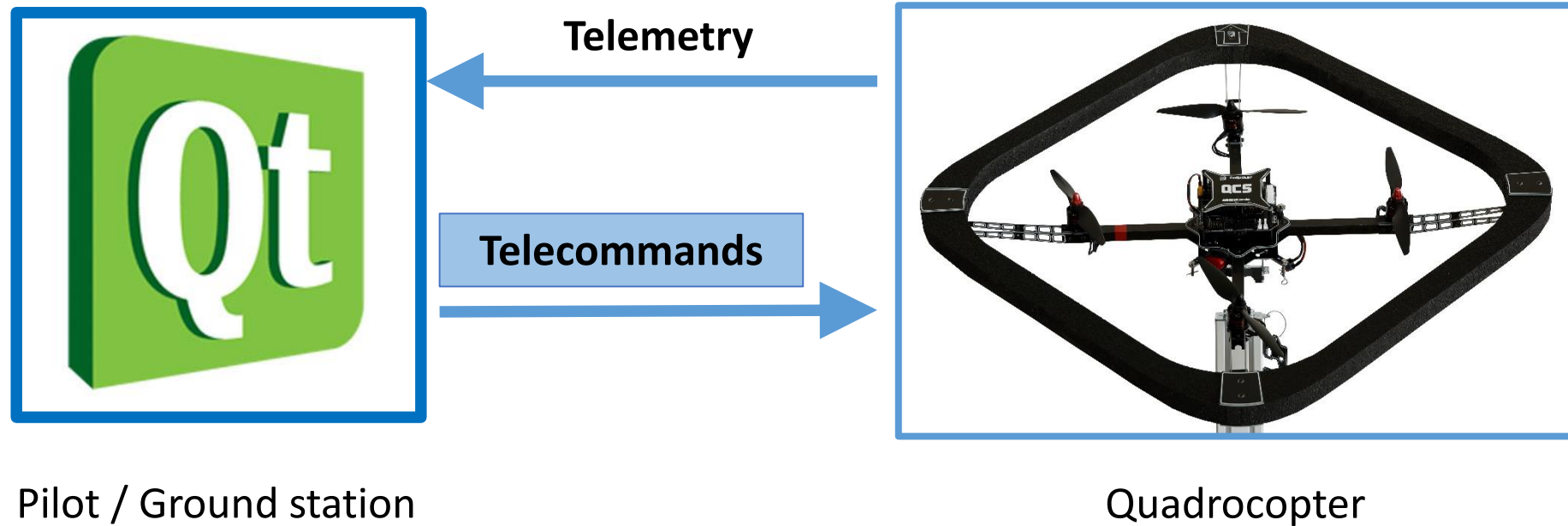


- Send Telecommands
- Receive Telecommands
- Editing of the User Interface
- Exercises

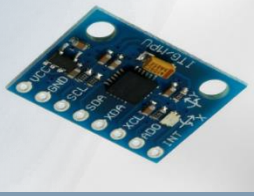
Time scope: approx. 2-4 hours



Telecommanding



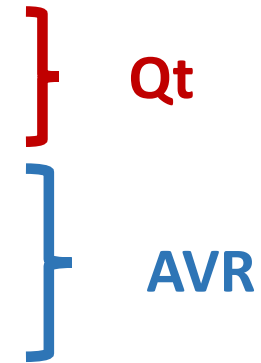
Telecommanding



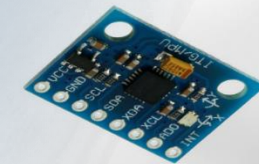
The Qt program is used to steer / control the quadrocopter

General procedure:

1. Formulate data into a message according to a protocol
2. Send the created message via USART
3. Receive message on AVR side
4. Interpret the message and extract data based on the protocol
5. Use data to steer / control quadrocopter



Send Telecommands



1. Formulate data into a message according to a protocol + 2. Send message

- `void createFrameMessageAndSend(int type, int data_length, float data[])` in *mainwindow.cpp* creates a frame according to the communication protocol and sends it.

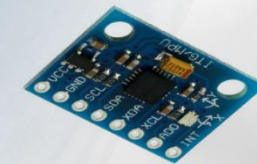
`int type` ID of the desired telecommand (see *protocol.h*)

```
35 // Telecommandos: IDs
36 #define TC_NOTHING -1
37 #define TC_LED 0
38 #define TC_TRANS 1
39 #define TC_AUTOFLUG 2
40 #define TC_MOTOR 3
41 #define TC_IMU 4
```

`int data_length` Number of the data that is sent in this frame.

`float data []` Data that is sent in this frame.

Receive Telecommands



3. Receive message on AVR side

- `int extract_type_and_data_from_telecommand_frame()`
 - In *telecommand.c*
 - Read from the USART and write the data to *my_data*
 - Returns the frame type of the received frame

4. Extract data

- `void my_read_LED_Command()`
 - Extract the data based on the type

5. Use data to steer / control quadrocopter

- `void my_read_telekommand()`
 - In *telecommand.c*
 - Is called every TC_UPDATE_RATE milliseconds
 - Reading and processing the telecommand

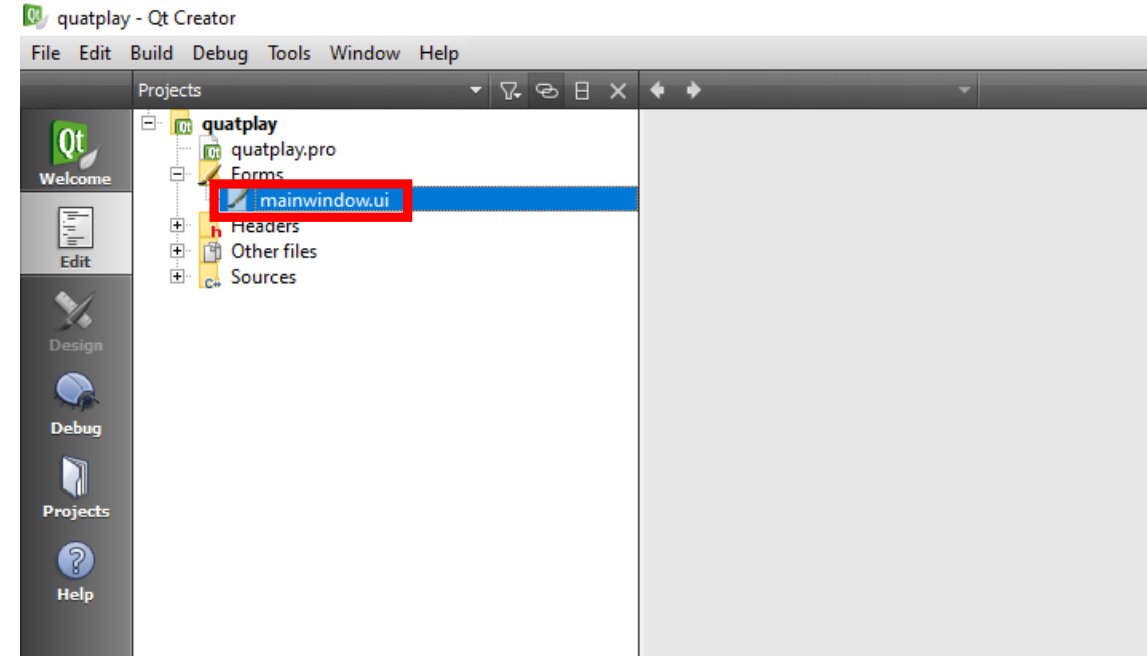
Editing the User Interface



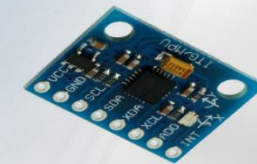
Adding and editing control elements

Double-click on *Formulardateien/mainwindow.ui*

→ Opens the editing view of the user interface

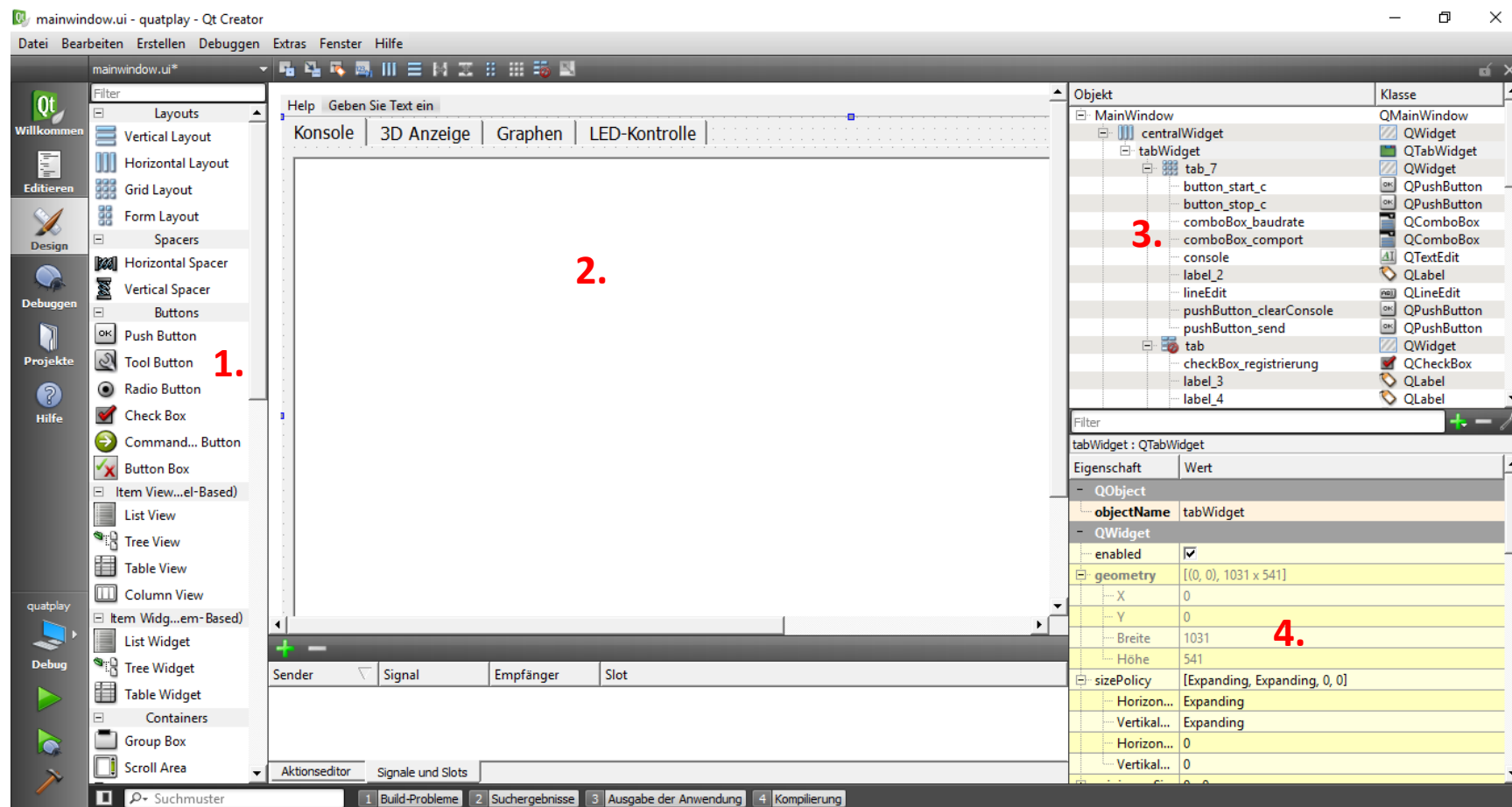


Editing the User Interface

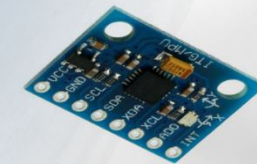


The editing view

1. Element library
2. UI - Preview
3. UI - Elements
4. Element - Details



Editing the User Interface



Add element:

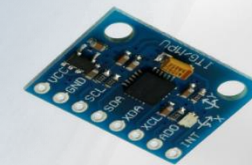
- Simply insert via drag and drop from the element library
- **Important:** After inserting, immediately adapt the name under **Element - Details / objectName** to make it meaningful:
- All elements of the user interface can be called in **mainwindow.cpp** with `ui->objectName`
- The Functions of methods can be called via `ui->objectName-> ...`

Here e.g.: `ui->checkBox_LED_1->isChecked()`

- Press **<Strg> + <Space>** after the `,->` to open a list of possible additions. This allows you to find the required functions quickly.

checkBox_LED_1 : QCheckBox	
Property	Value
- QObject	
objectName	checkBox_LED_1
- QWidget	
enabled	☒
geometry	[(40, 30), 111 x 18]
X	40
Y	30
Width	111
Height	18
sizePolicy	[Minimum, Fixed, 0, 0]
Horizon...	Minimum
Vertical ...	Fixed
Horizon...	0
Vertical ...	0

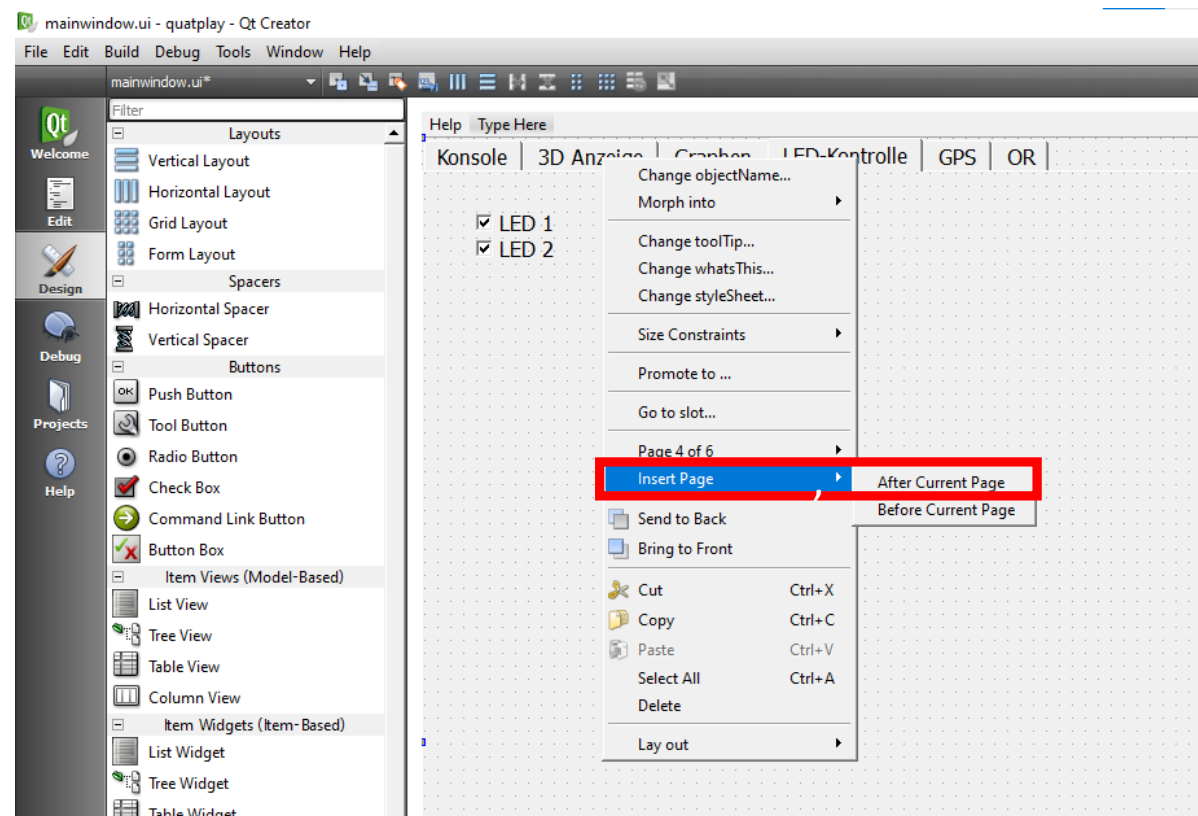
Editing the User Interface

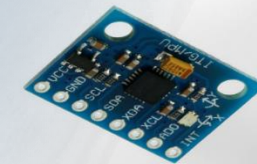


Special case: Adding tabs:

1. Right-click on an existing tab
2. Insert page → Then select
3. Enter *currentTabText* under *Element details*
4. Change *currentTabName* under *Element details*

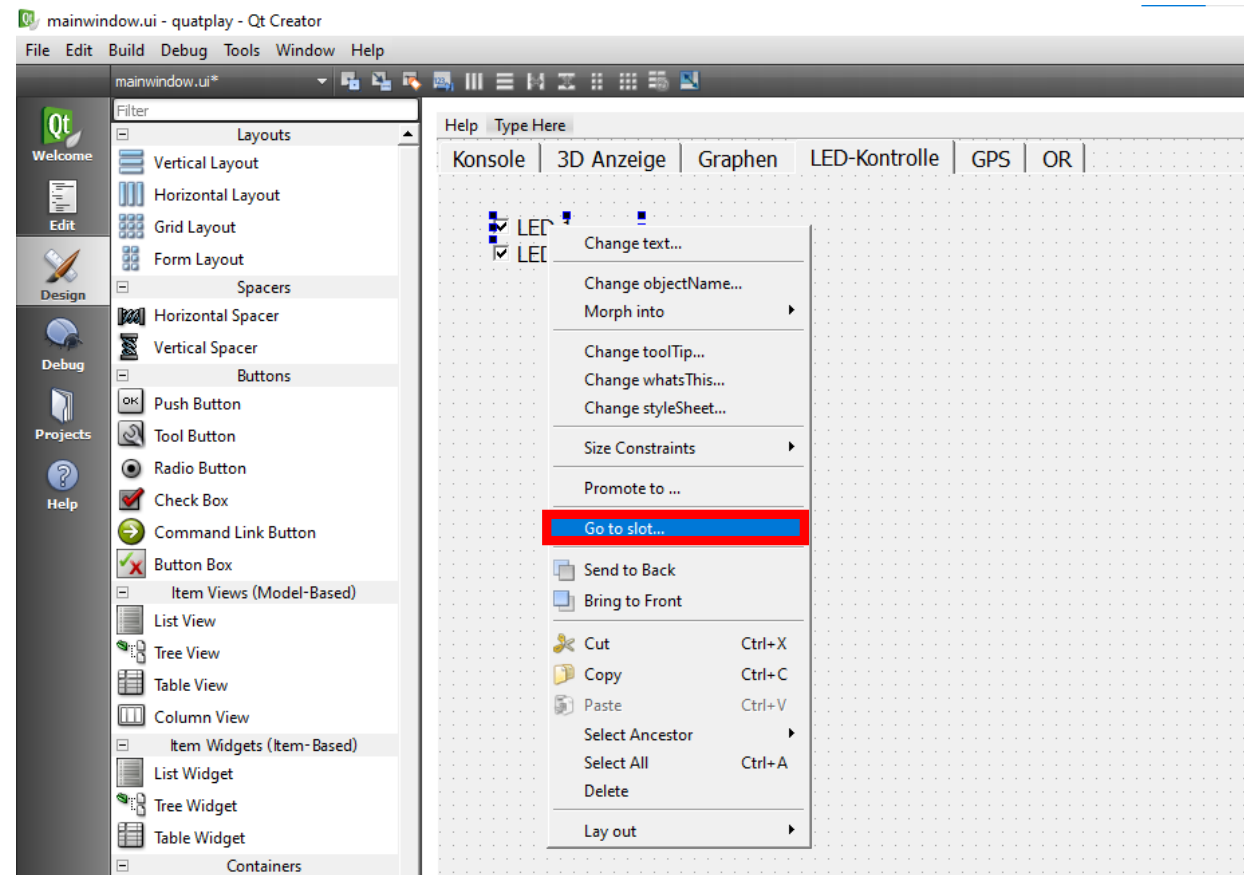
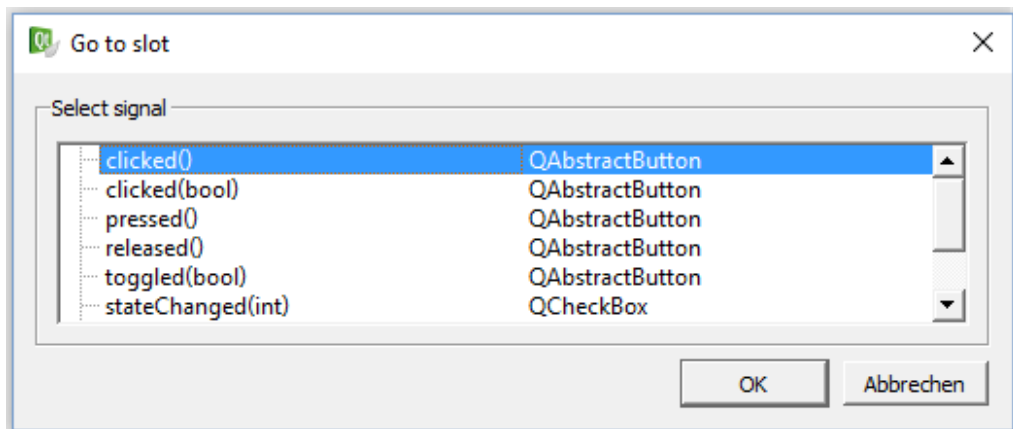
Eigenschaft	Wert
tabPosition	North
tabShape	Rounded
currentIndex	4
iconSize	16 x 16
elideMode	ElideNone
usesScrollButtons	☒
documentMode	☐
tabsClosable	☐
movable	☐
currentTabText	Seite
currentTabName	tab_3
currentTabIcon	
currentTabToolTip	
currentTabWhats...	



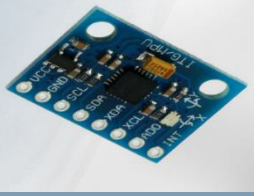


Program functionality:

1. Right-click on element
2. Select ***Slot anzeigen...***
3. Select the desired action and confirm with **OK**
 → The corresponding function is created under ***mainwindow.cpp***



Exercises



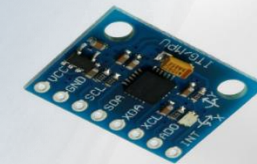
Necessary hardware:

- EMQ3000
- Mini-USB cable for power and flashing
- Mini-USB cable for communication: USART to PC
- QCS / QCSF

Necessary software:

- AVR Studio 32 installed (with toolchain and flip driver)
- Qt Creator 2.0.1 and Qt 4.7.0 (32bit)
- EMQt_Framework\quatplay\quatplay.pro (code)

Exercises



Exercise 1:

Create a new tab next to the „*Graphs*“ with the label „*LED control*“.

Add two check boxes LED 1 and LED 2 to the newly created area, which are activated when the program starts.

Exercise 2:

Program the new check boxes so that you can use them to control LEDs 1 and 2 of the *EMQ3000*.

`void createFrameMessageAndSend(int type, int data_length, float data [])` in *mainwindow.cpp* sends a frame to the *EMQ3000* via *USART*. To process the telecommands, the functions `void my_read_telecommand()` and `void my_read_LED_Command()` must be implemented in *telecommand.c*.

The function `int extract_type_and_data_from_telecommand_frame()` helps to read out the type of telecommands. Compare the *defines* in *protocol.h*.