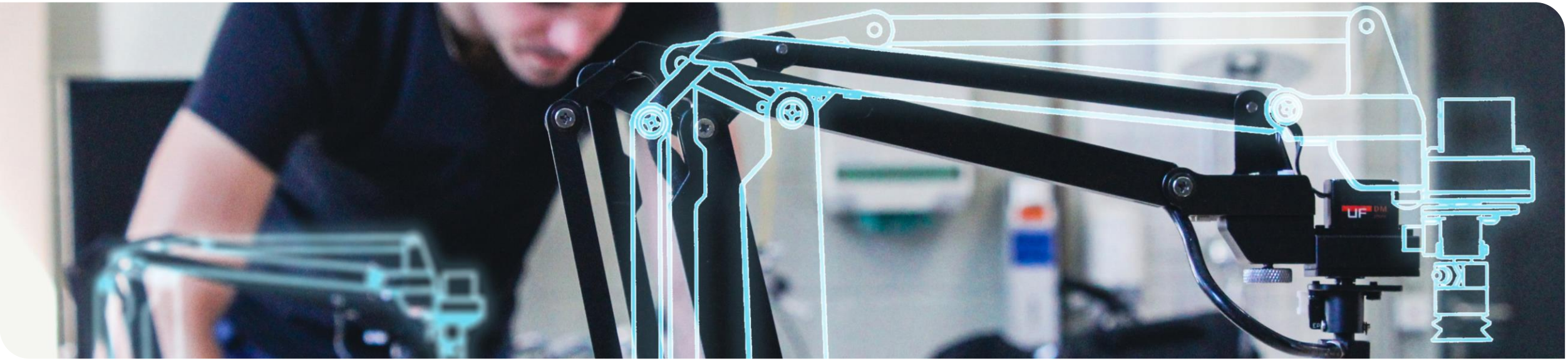
Karlsruher Institut für Technologie

Institut für Technik der Informationsverarbeitung

Seamless Engineering

2. Introduction meeting



Using bwLehrpool

- Virtual machine prepared with required software and provided simulation environment
- Use of SCC pool computers remotely directly in the browser
- Access: https://pool-remote.scc.kit.edu

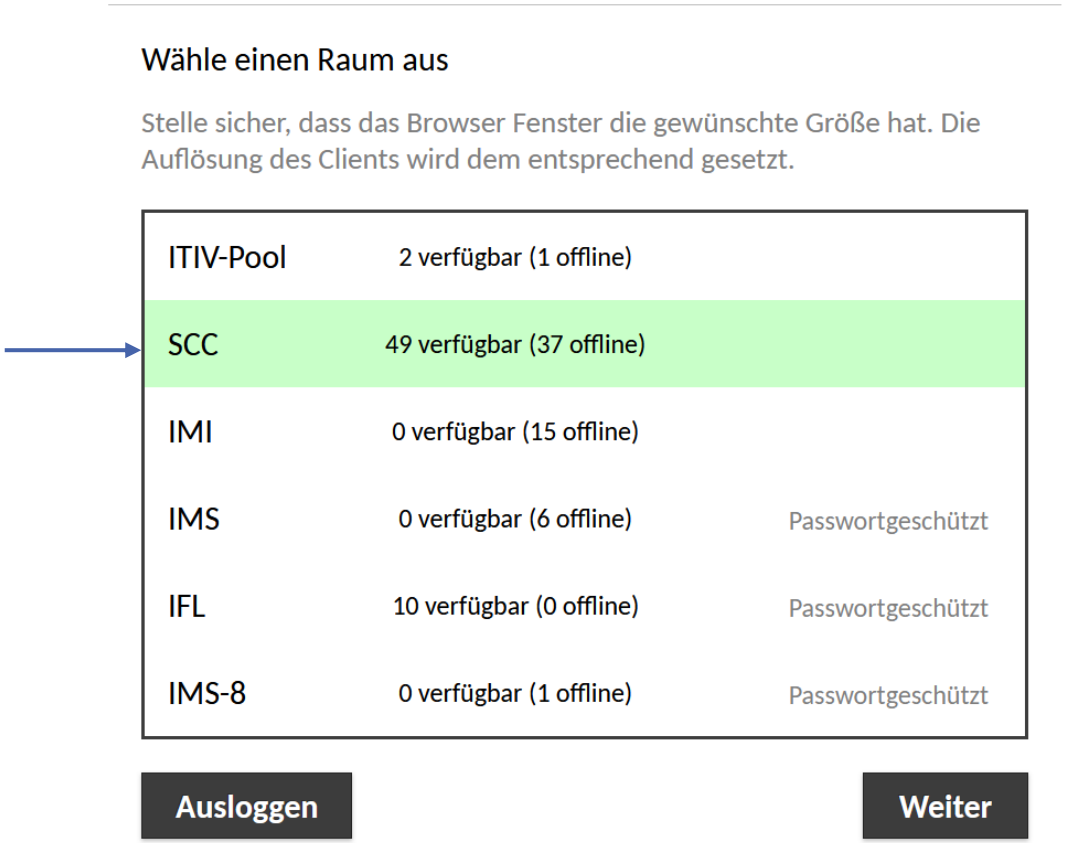
Using bwLehrpool

- Log in with your own abbreviation and password

Using bwLehrpool

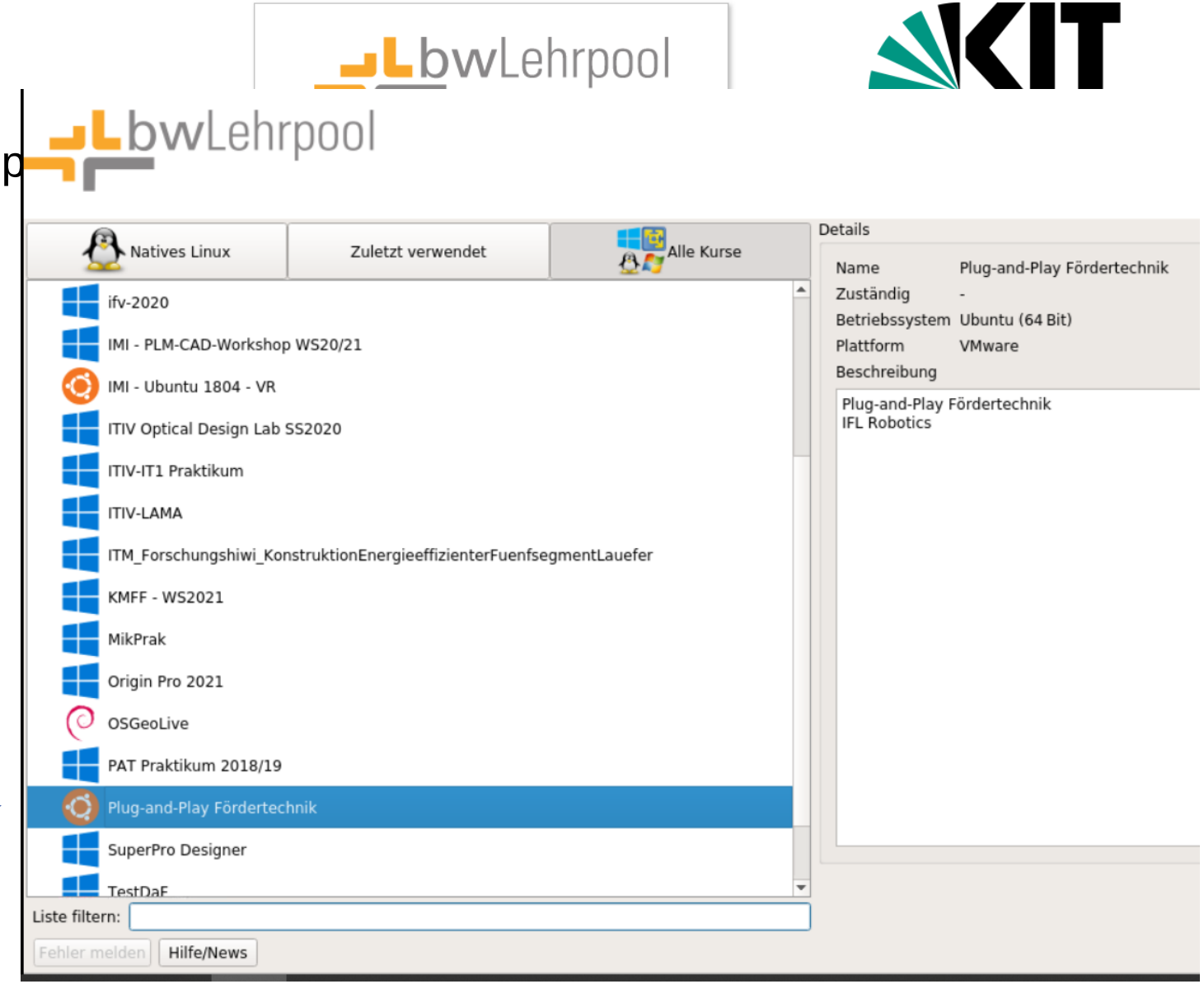
- Log in with your own abbreviation and password

- Select SCC Pool computers



Using bwLehrpool

- Log in with your own abbreviation and p
- Select SCC Pool computers
- Select „Seamless Engineering“



Using bwLehrpool

- **Note:** After the virtual machine is terminated, it is reset to its original state. **Data that is not backed up is lost!**

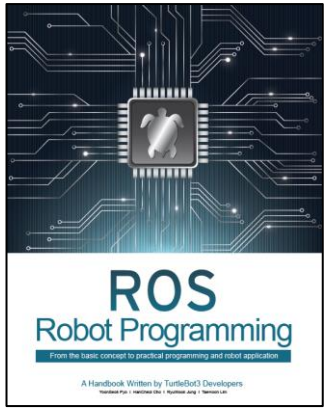
- To save one's own code, the personal memory is mounted on the SCC in the virtual machine.
- Access: /home/student/PERSISTENT_mov

Provided simulation environment

- The required simulation environment is already available in the virtual machine
- Access: /home/student/xxxxx
- Contains code to start the simulation environment as well as to load the hardware

ROS & OPENCV

ROS



- ROS = Robot Operating System
- Middleware not an OS (mainly used with Ubuntu)
- It provides a plattform for developing robot software
- Many robots are based on ROS (research and industry)
- Simulation in virtual environment
- Goal : Reuse Code in Robotics!
- Executables (small programs, e.q. vision system) can be designed individually and are loosely coupled at runtime (communication is done via messages)
- Executables which belong together are grouped into packages which can be easily shipped and shared
- Federated system of code repositories enables collaboration
- Works with Python and C++

ROS.org
Documentation

ROS – Filesystem Level

- **Packages:** main unit for organizing software in ROS. It can contain ROS runtime processes (nodes), libraries, datasets, config files, or anything else that belongs to this operational unit.
- **Metapackages:** (left out)
- **Package manifest** : provide metadata about package*, package.xml*
- **Repositories:** a collection of packages sharing the same Version Control System (not that important for you), a collection of packages in organized in a repo
- **Message types:** message descriptions, define the data structure for messages sent in ROS
- **Service types:** define the request and response data structures for services in ROS.

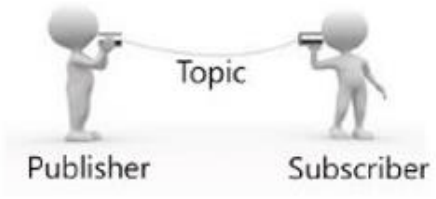
ROS – Computation Graph Level 1/2

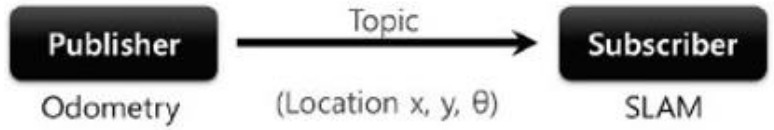
- (The *Computation Graph* is the peer-to-peer network of ROS processes that are processing data together.)
- Its concepts are (this list is not complete!):
  - **Nodes** : Nodes are processes that perform computation. For example, one node controls a laser range-finder, one node controls the wheel motors, one node performs localization, one node performs path planning, one Node provides a graphical view of the system, and so on. Written in roscpp or rospy.
  - **Master:** The ROS Master provides name registration and lookup to the rest of the Computation Graph. Without the Master, nodes would not be able to find each other, exchange messages, or invoke services.
  - **Messages:** Nodes communicate with each other by passing messages. A message is simply a data structure, comprising typed fields.
  - **Topics:** Messages are routed via a transport system with publish / subscribe semantics. A node sends out a message by *publishing* it to a given topic. The topic is a name that is used to identify the content of the message. A node that is interested in a certain kind of data will *subscribe* to the appropriate topic.

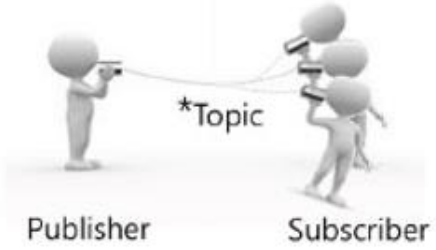
ROS – Computation Graph Level 2/2

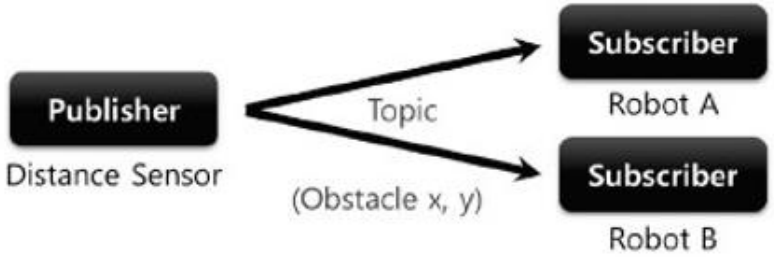
- **Services** : The publish / subscribe model is a very flexible communication paradigm, but its many-to-many, one-way transport is not appropriate for request / reply interactions, which are often required in a distributed system. Request / reply is done via services, which are defined by a pair of message structures: one for the request and one for the reply.
- **Actions** : *http://wiki.ros.org/actionlib_tutorials/Tutorials/SimpleActionServer%28ExecuteCallbackMethod%29*

ROS – Topics, Publisher and Subscriber

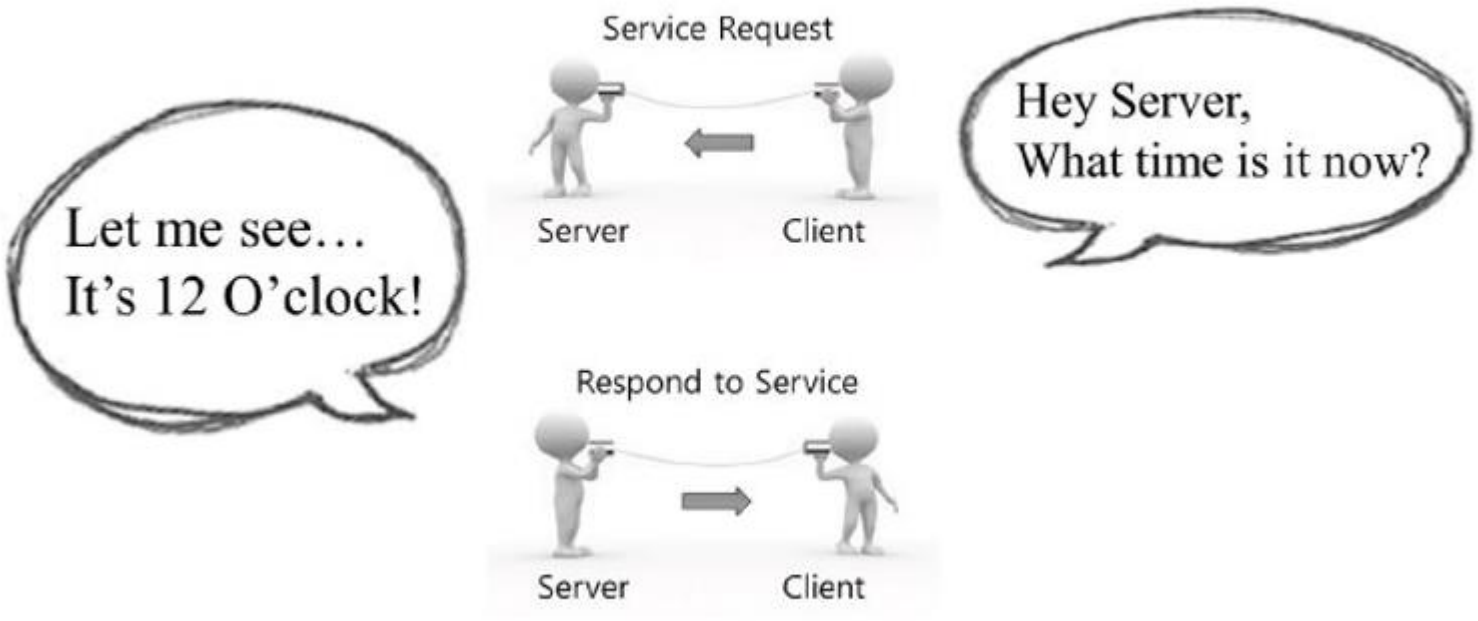








ROS – Services



ROS Message Example

sensor_msgs/Image Message

File: sensor_msgs/Image.msg

Raw Message Definition

```
# This message contains an uncompressed image
# (0, 0) is at top-left corner of image
#

Header header        # Header timestamp should be acquisition time of image
                     # Header frame_id should be optical frame of camera
                     # origin of frame should be optical center of camera
                     # +x should point to the right in the image
                     # +y should point down in the image
                     # +z should point into to plane of the image
                     # If the frame_id here and the frame_id of the CameraInfo
                     # message associated with the image conflict
                     # the behavior is undefined

uint32 height         # image height, that is, number of rows
uint32 width          # image width, that is, number of columns

# The legal values for encoding are in file src/image_encodings.cpp
# If you want to standardize a new string format, join
# ros-users@lists.sourceforge.net and send an email proposing a new encoding.

string encoding       # Encoding of pixels -- channel meaning, ordering, size
                      # taken from the list of strings in include/sensor_msgs
/image_encodings.h

uint8 is_bigendian    # is this data bigendian?
uint32 step           # Full row length in bytes
uint8[] data          # actual matrix data, size is (step * rows)
```

Compact Message Definition

```
std_msgs/Header header
uint32 height
uint32 width
string encoding
uint8 is_bigendian
uint32 step
uint8[] data
```

autogenerated on Fri, 15 Jan 2021 03:18:41

ROS – Learning by doing

http://wiki.ros.org/ROS/Tutorials

1. Core ROS Tutorials

1.1 Beginner Level

1. Installing and Configuring Your ROS Environment
   This tutorial walks you through installing ROS and setting up the ROS environment on your computer.
2. Navigating the ROS Filesystem
   This tutorial introduces ROS filesystem concepts, and covers using the roscd, rosls, and rospack commandline tools.
3. Creating a ROS Package
   This tutorial covers using roscreate-pkg or catkin to create a new package, and rospack to list package dependencies.
4. Building a ROS Package
   This tutorial covers the toolchain to build a package.
5. Understanding ROS Nodes
   This tutorial introduces ROS graph concepts and discusses the use of roscore, rosnode, and rosrun commandline tools.
6. Understanding ROS Topics
   This tutorial introduces ROS topics as well as using the rostopic and rqt_plot commandline tools.
7. Understanding ROS Services and Parameters
   This tutorial introduces ROS services, and parameters as well as using the rosservice and rosparam commandline tools.
8. Using rqt_console and roslaunch
   This tutorial introduces ROS using rqt_console and rqt_logger_level for debugging and roslaunch for starting many nodes at once. If you use `ROS fuerte` or ealier distros where rqt isn't fully available, please see this page with this page that uses old `rx` based tools.
9. Using rosed to edit files in ROS
   This tutorial shows how to use rosed to make editing easier.
10. Creating a ROS msg and srv
    This tutorial covers how to create and build msg and srv files as well as the rosmsg, rossrv and roscp commandline tools.
11. Writing a Simple Publisher and Subscriber (C++)
    This tutorial covers how to write a publisher and subscriber node in C++.
12. Writing a Simple Publisher and Subscriber (Python)
    This tutorial covers how to write a publisher and subscriber node in python.
13. Examining the Simple Publisher and Subscriber
    This tutorial examines running the simple publisher and subscriber.
14. Writing a Simple Service and Client (C++)
    This tutorial covers how to write a service and client node in C++.
15. Writing a Simple Service and Client (Python)
    This tutorial covers how to write a service and client node in python.
16. Examining the Simple Service and Client
    This tutorial examines running the simple service and client.
17. Recording and playing back data
    This tutorial will teach you how to record data from a running ROS system into a .bag file, and then to play back the data to produce similar behavior in a running system
18. Reading messages from a bag file
    Learn two ways to read messages from desired topics in a bag file, including using the `ros_readbagfile` script.
19. Getting started with roswtf
    Basic introduction to the roswtf tool.
20. Navigating the ROS wiki
    This tutorial discusses the layout of the ROS wiki (wiki.ros.org) and talks about how to find what you want to know.
21. Where Next?
    This tutorial discusses options for getting to know more about using ROS on real or simulated robots.

ROS – Learning by doing

- General

  http://wiki.ros.org/tf/Tutorials

- TF – Managing coordinate transformations

  http://wiki.ros.org/navigation/Tutorials/RobotSetup/TF

- Video tutorials on ROS (3 x approx. 50min videos)

  https://www.youtube.com/watch?v=0BxVPCInS3M&t=1106s

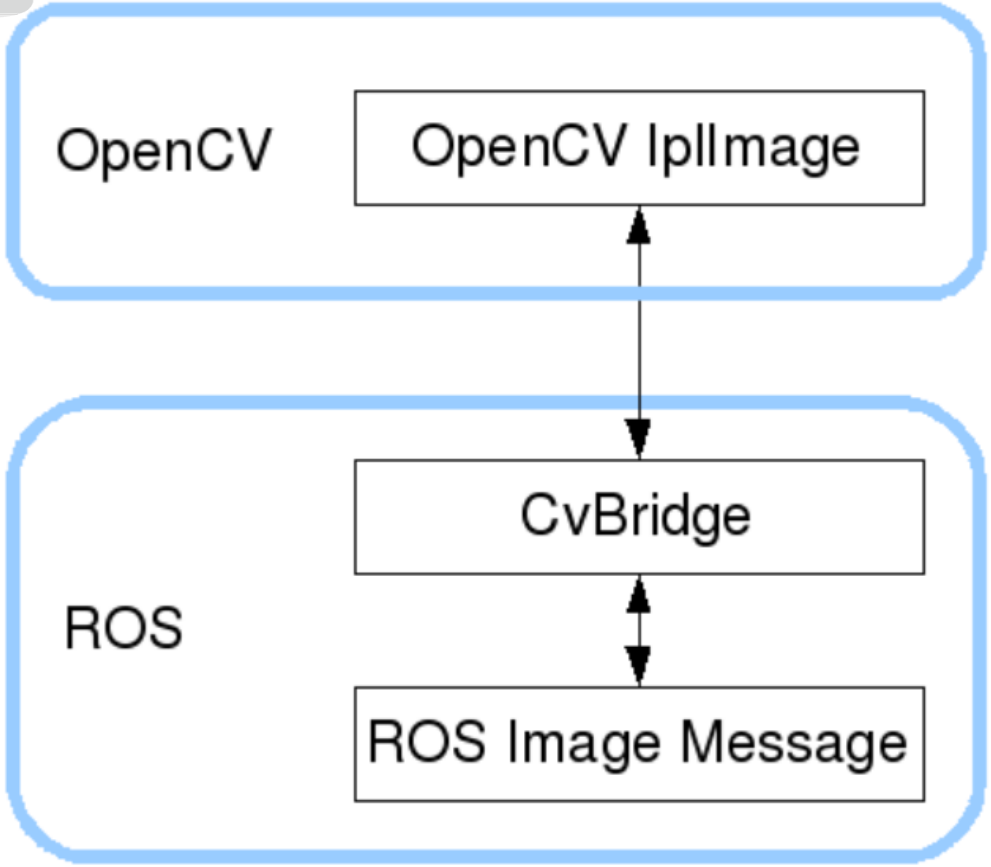
OpenCV

- Developed by Intel in 2011
- Free library for computer vision
- Strong background in research and industry
- C++ and Python supported
- Alternatives : SciPy and MATLAB

Tutorial 1

From ROS to OpenCV

- OpenCV : BGR channel
- OpenCV : image is numpy array

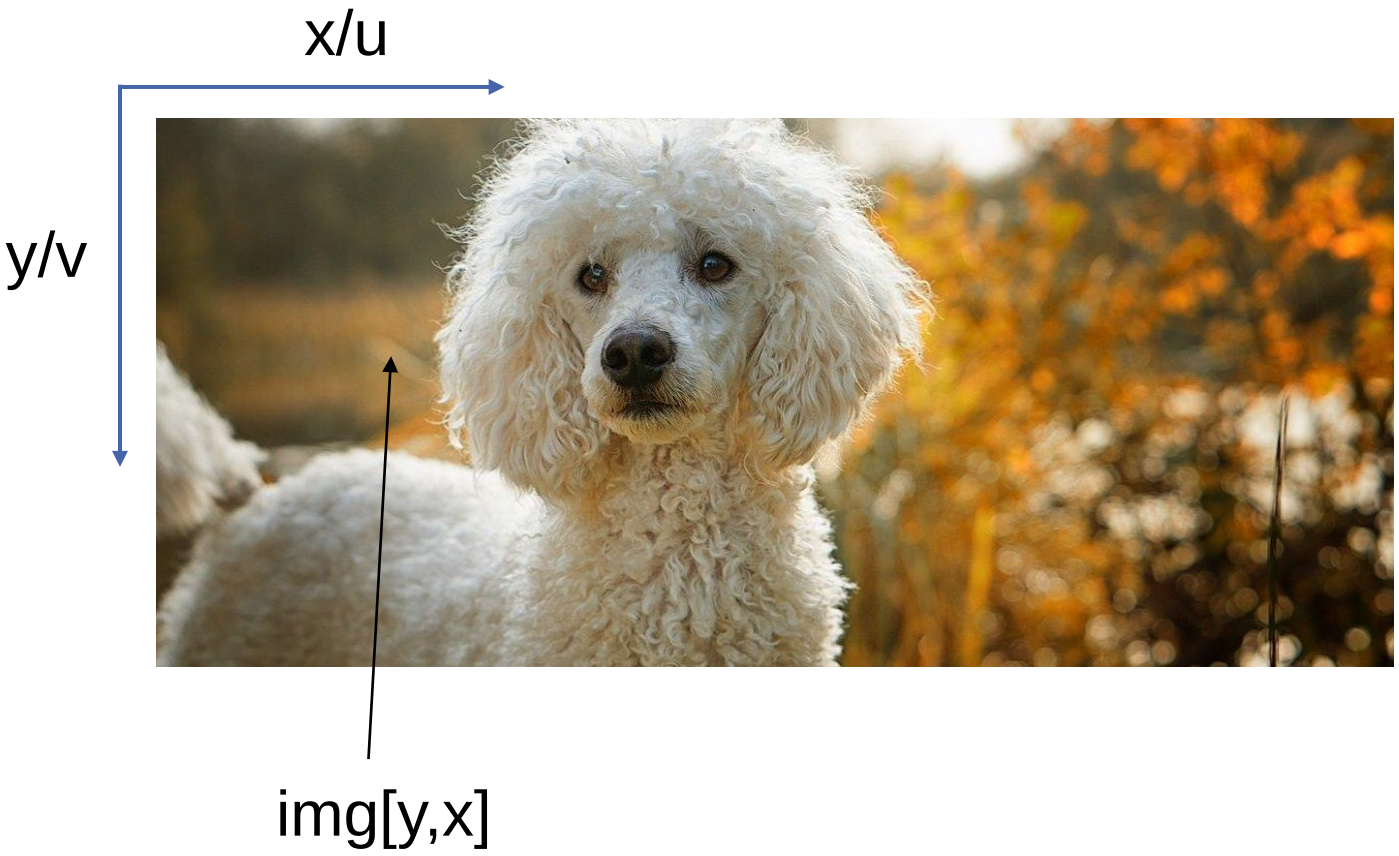


http://wiki.ros.org/cv_bridge/Tutorials/ConvertingBetweenROSImagesAndOpenCVImagesPython

Tutorial 2

Working with images

- Show dimensions
- Manipulating Pixels
- Segmenting image



Python

- What is Python?
  - Interpreted high-level and general-purpose programming language
  - Object-oriented approach
  - Large community
- Main advantages of Python (compared to Java or C++):
  - Easy to use and to read
  - Compact
- Main disadvantages of Python:
  - Speed

- Python Tutorials
  - https://www.w3schools.com/python/default.asp

Java

```java
File dir = new File("."); // get current directory
File fin = new File(
    dir.getCanonicalPath() + File.separator + "Code.txt"
);

FileInputStream fis = new FileInputStream(fin);

// Construct the BufferedReader object
BufferedReader in = new BufferedReader(new InputStreamReader(fis));

String aLine = null;
while ((aLine = in.readLine()) != null) {
    // //Process each line, here we count empty lines
    if (aLine.trim().length() == 0) {}
}

// do not forget to close the buffer reader
in.close();
```

Python

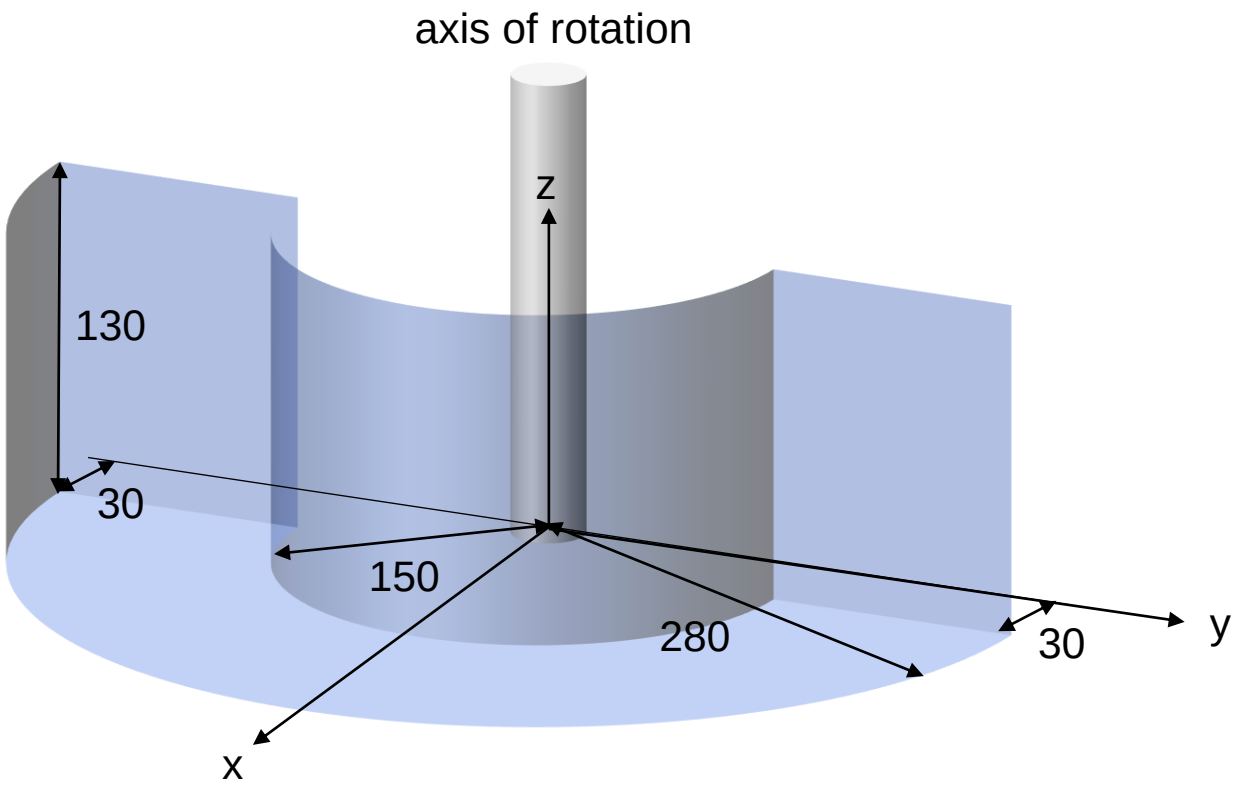
```python
my_file = open("/home/xiaoran/Desktop/test.txt")

print(my_file.read())
my_file.close()
```

HARDWARE

Working Space of uArm [mm]

- 30 <= x <= 280
- -278 <= y <= 278
- -120 <= z <= 130
- 150 <= r <= 280



Most important uArm Topics, Actions & Services

Actions:

- NAMESPACE/uarm_move — moves to target position
- NAMESPACE/uarm_reset — reset to position [180, 0, 130]

Services

- NAMESPACE/uarm_set_pump — pump of vacuum gripper true / false
- NAMESPACE/uarm_get_state — gives position, busy and pump status

Topics

- NAMESPACE/busy — true if currently working on task

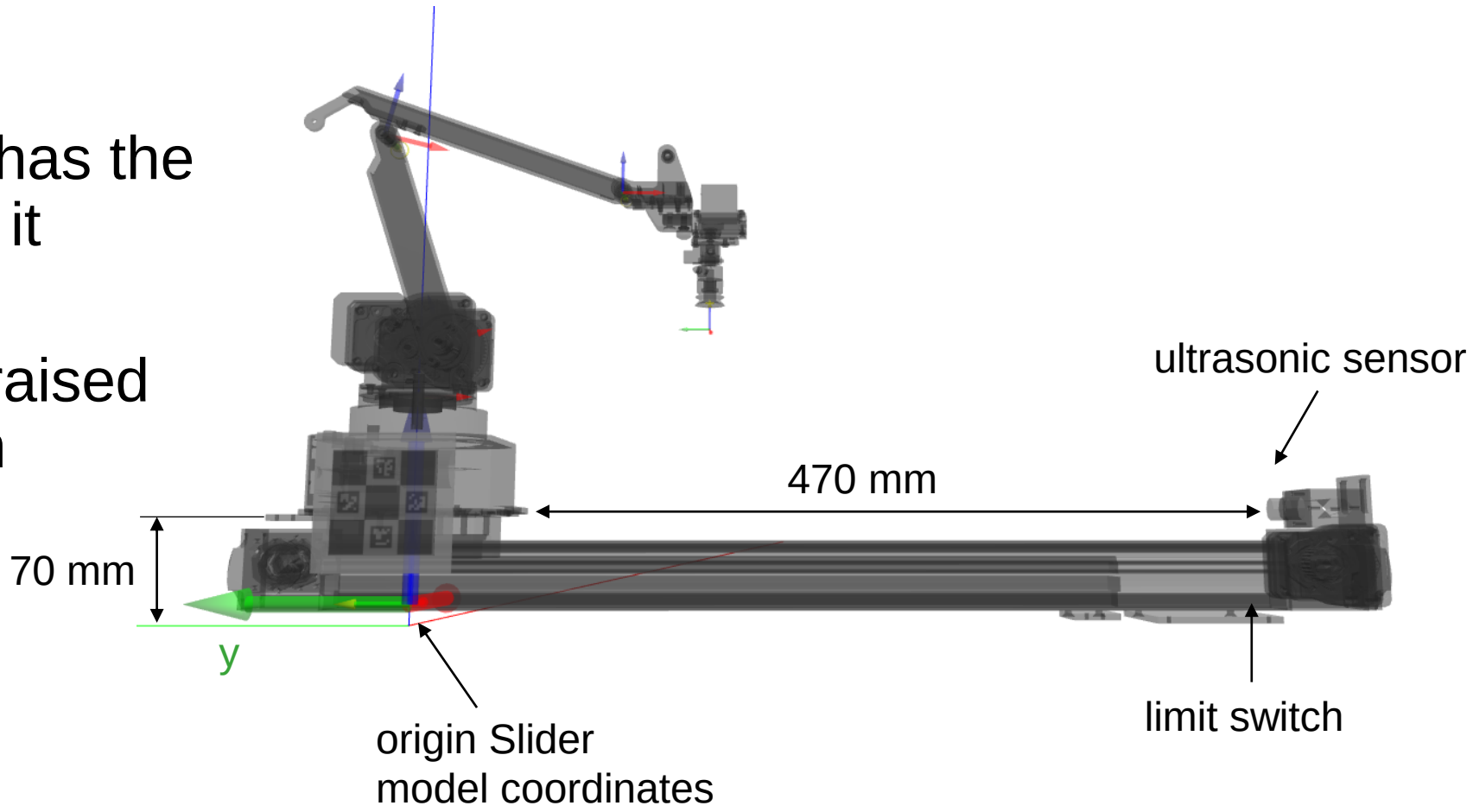
Note: Only one action / service can be actively used at a time. All other requests are blocked during the execution of a task and get "success = False" (or in the case of uarm_get_state "ready = False") as result.

uArm with Slider

- Slider Working Space [mm]:
  -470 <= y <= 0

- only the real model has the sensors attached to it

- in our environment raised by additional 70 mm (so uArm's platform on 140 mm)



Most important uArm Slider Topics, Actions & Services

Karlsruher Institut für Technologie

Actions:

- NAMESPACE/slider_move — moves to target position y in range [-470, 0]
- NAMESPACE/drive_to_limit_switch — move until switch gets triggered (real model)

Services

- NAMESPACE/uarm_set_pump — pump of vacuum gripper true / false
- NAMESPACE/uarm_get_state — gives position, busy and pump status

Topics

- NAMESPACE/uarm/busy — true if uArm currently working on task
- NAMESPACE/slider/busy — true if slider currently working on task
- NAMESPACE/state_raw — raw data from slider (real model)

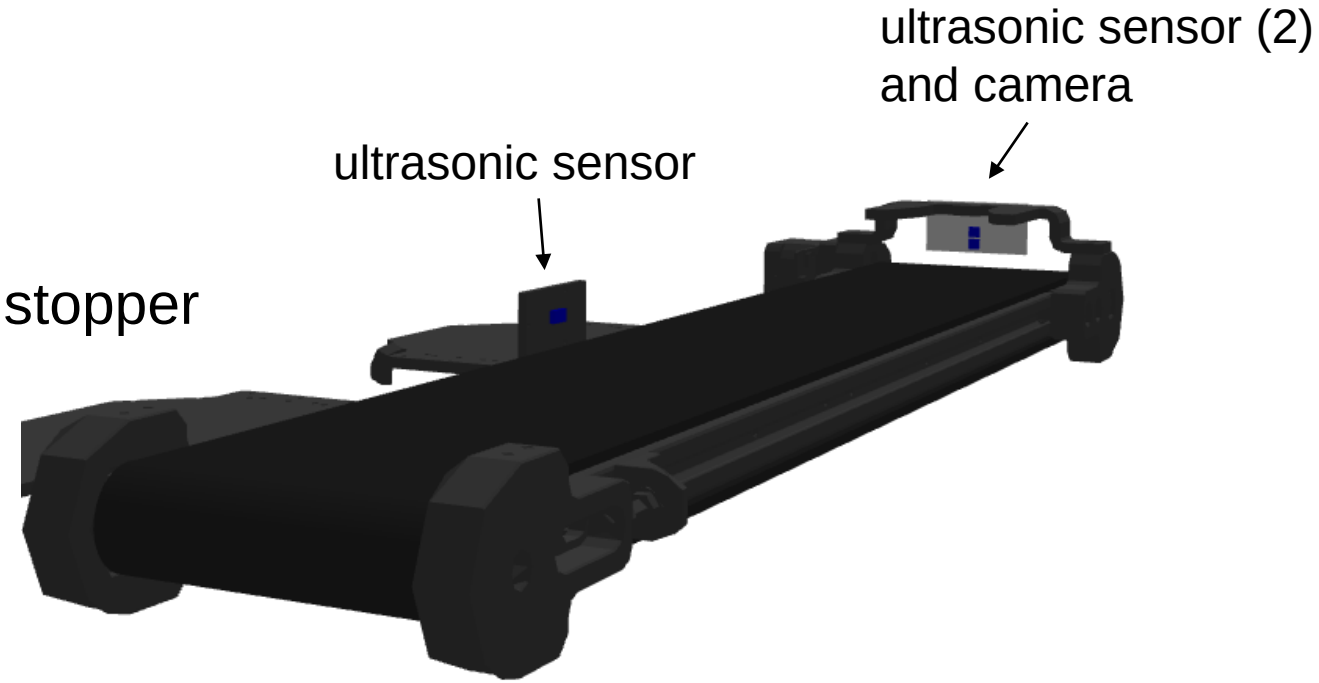
uArm Conveyor

Sensors – real Model:

- ultrasonic sensor
- color sensor at the stopper

Sensors – Simulation:

- ultrasonic sensor
- camera & ultrasonic sensor at the stopper



Most important uArm Conveyor Services

Services

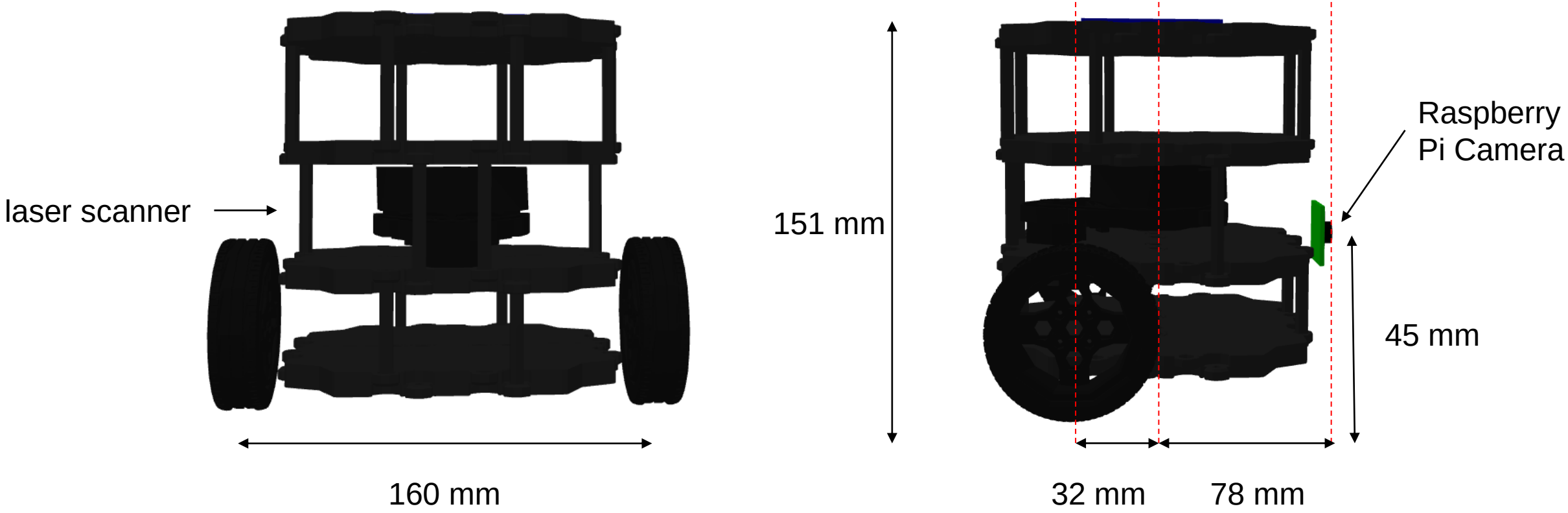
conveyor/conveyor/control	move conveyor belt (-15.0,0,15.0) (simulation)
NAMESPACE/move_forward (Int16 data) NAMESPACE/move_backward (Int16 data)	moves belt with the duration of data (defaults to 0 – 0 means infinite) (real model)
NAMESPACE/stop	Stop conveyor belt (real model)
NAMESPACE/start_search NAMESPACE/stop_search	stops the conveyor when the ultrasonic sensor detects an object. Deactivates itself when object detected (real model)

Most important uArm Conveyor Topics

Topics

- NAMESPACE/busy — True, while search mode activated (real)
- NAMESPACE/ultrasonic/scan — ultrasonic sensor beside belt [m] (sim)
- NAMESPACE/ultrasonic_2/scan — ultrasonic sensor at stopper [m] (sim)
- NAMESPACE/camera/image_raw — camera at stopper (sim)
- NAMESPACE/state — information about belt, mode, color sensor, time moving since last start command and (feed – not attached here) (real)

Turtlebot3 Burger



Most important Turtlebot 3 Topics, Actions & Services

Actions:

- NAMSPACE/move_base_simple — moves to target position

Topics

- NAMSPACE /camera/image — image of camera (sim)
- raspicam_node/image — image of camera (real)
- NAMSPACE/camera/image_charuco_pose — pose of recognized marker relative to camera (origin in bottom right corner of marker) (sim)
- raspicam_node /image_charuco_pose — ~ (real)
- NAMSPACE/scan_filtered — array of distances measured by laser scanner

Example Nodes in Python code

uArm

- roslaunch uarm spawn.launch start_gazebo:=true box:=true — to start gazebo with a uArm and a box
- rosrun uarm very_simple_action_client_example.py (path to code: ~/2021_pnp_ws/src/models/uarm/scripts/) — to start example script in new terminal

Slider

- roslaunch spawn.launch start_gazebo:=true — use spawn command in new terminal
- rosrun slider very_simple_action_client_example.py — to start example script in new terminal

uarm_msgs

uarm_move.action

```
#goal definition
int64 x
int64 y
int64 z
---
#result definition
bool success
---
#feedback
#int32[] sequence
```

uarm_reset.action

```
#goal definition
---
#result definition
bool success
---
#feedback
```

uarm_pump.srv

```
bool run
---
bool success
```

uarm_get_state.srv

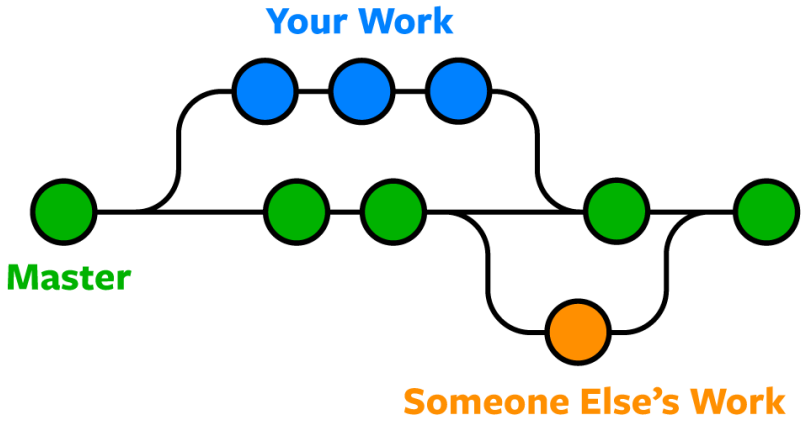
```
---
bool ready
bool pump
int64 x
int64 y
int64 z
```

GIT

Git structure (1)

- Entire source code is in a repository

- Code can be developed in any number of branches in parallel
- Example: Master-Branch: last stable program version
  - Test-Branch: current development status, not yet tested („Alpha-Version“)



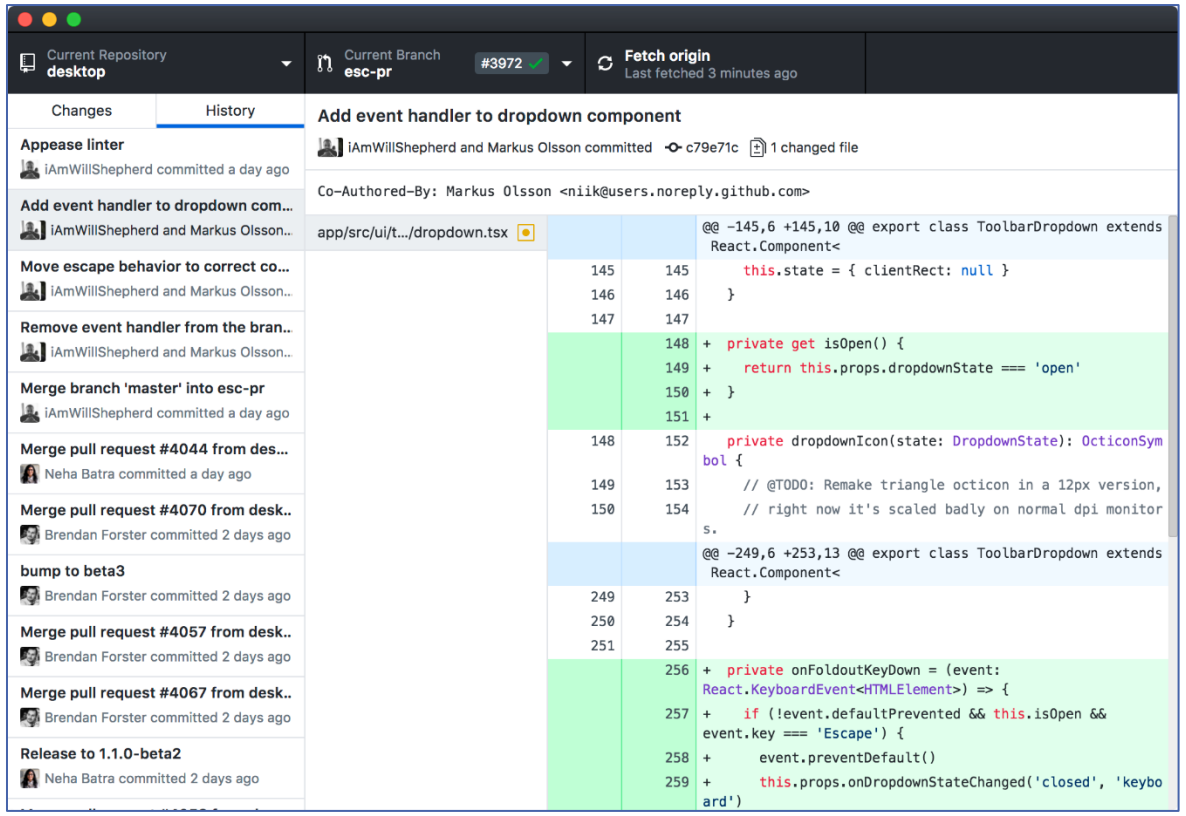
- Developers add new (improved) code via commit to the branch
  - Every commit is saved and can be undone

Git Structure (2)

- Merging of branches
  - Automatic comparison of the code in the different branches („ always take the latest version“)
  - Most of the time this works without problems, otherwise solve Merge conflict.
- Developers and program users can communicate with each other about issues
  - Report found bugs
  - Specify suggestions for improvement and wishes
- Respository (Code) + Issues = Project

What is GitHub Desktop?

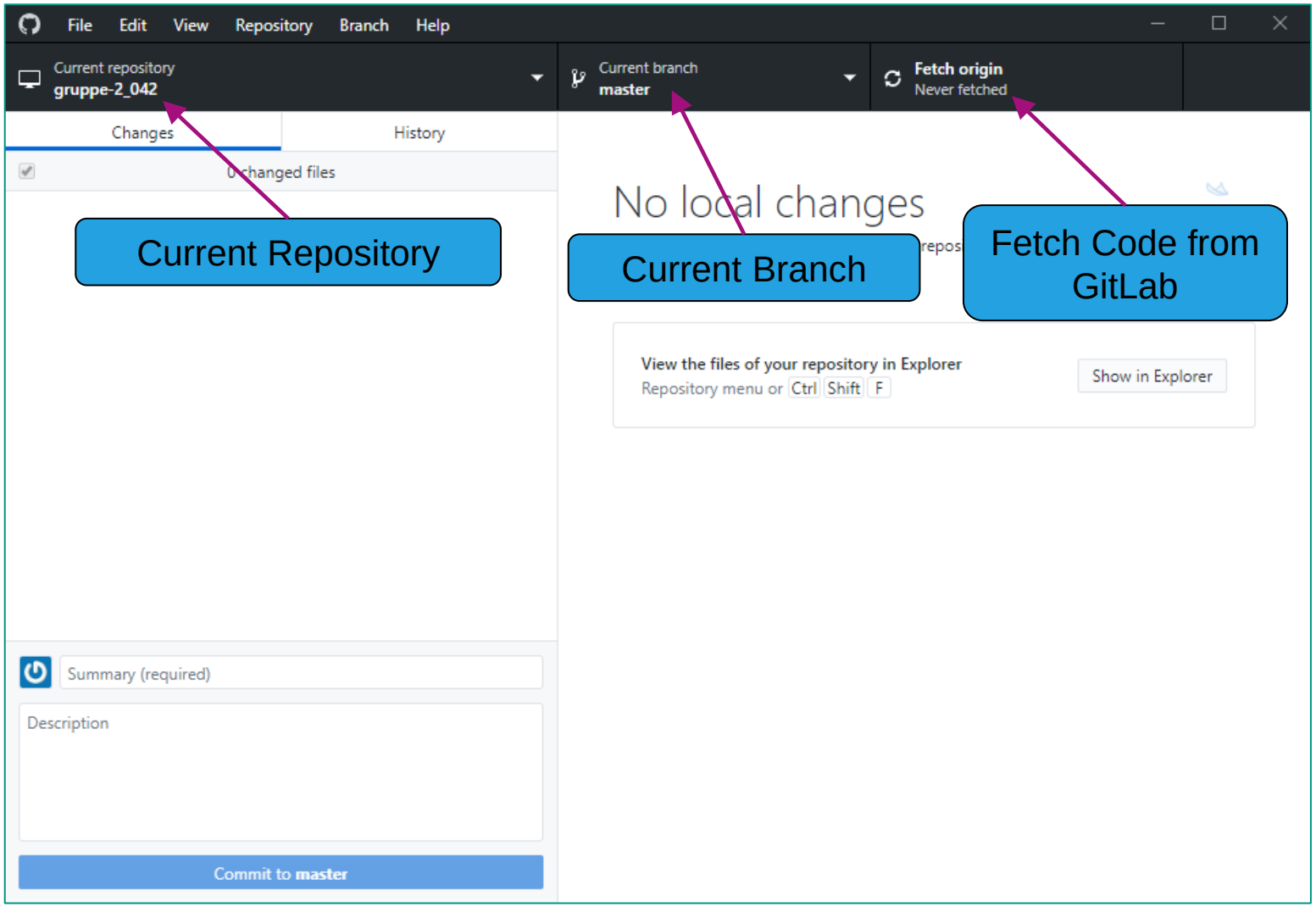
- Service to synchronize Git with your computer with a GUI
- Contents of the Git repository are stored locally on your computer
- Available for Windows and MacOS

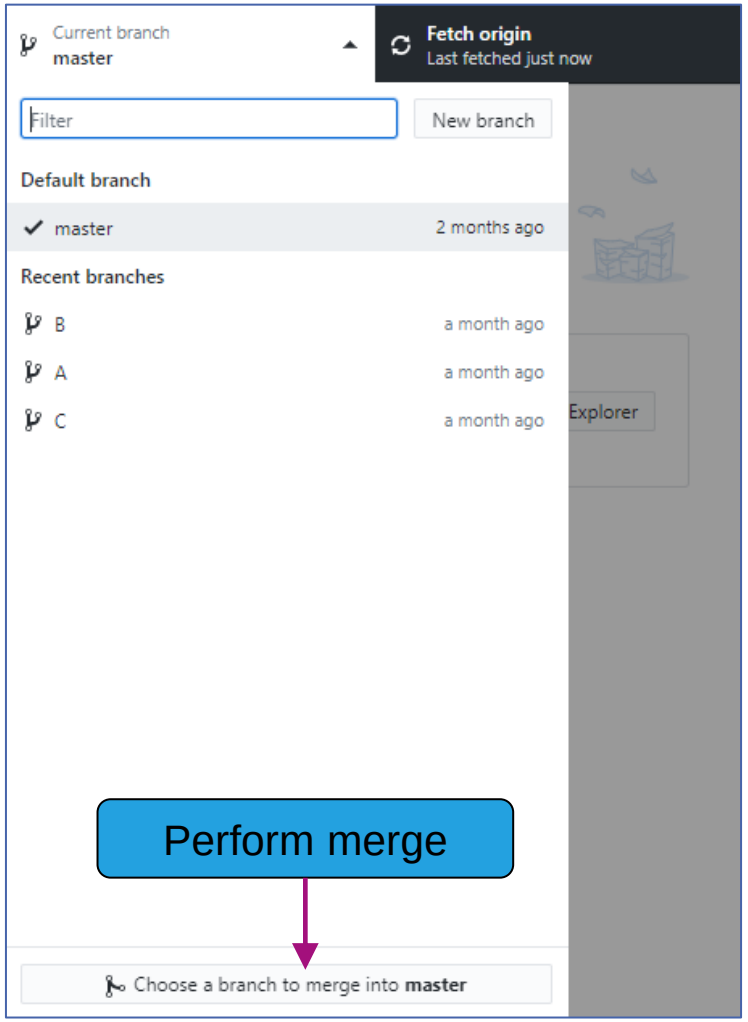


Setting up GitLab and GitHub Desktop

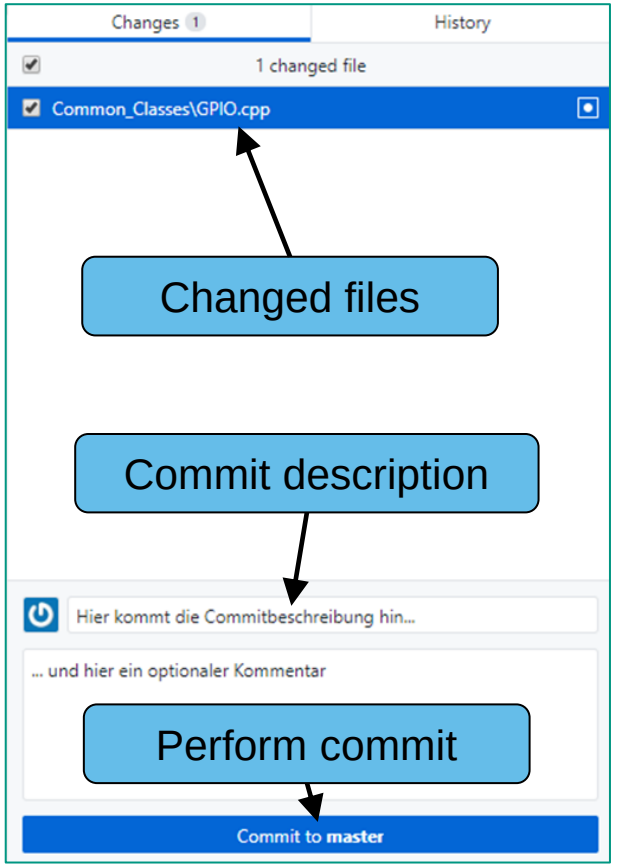
- Step-by-step instructions available in the "Seamless Engineering Manual”
- Be sure to follow 1:1 if you did not set up GitLab in the introductory session!
- Access to GitLab Seamless Engineering: https://git.scc.kit.edu/seamless-engineering
- Download GitHub Desktop:
  - Windows and macOS: https://desktop.github.com/

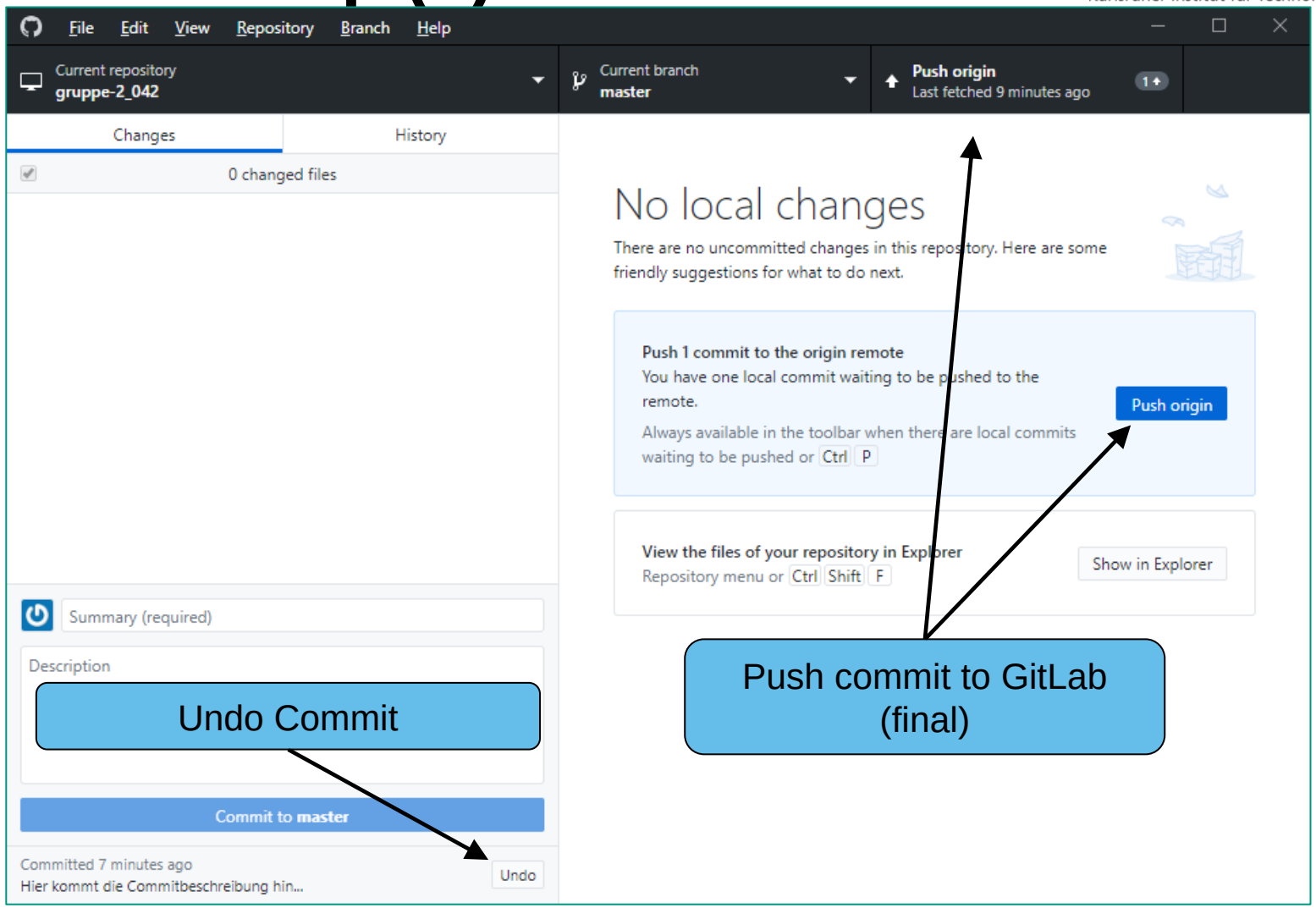
Working with GitHub Desktop (1)





Working with GitHub Desktop (2)

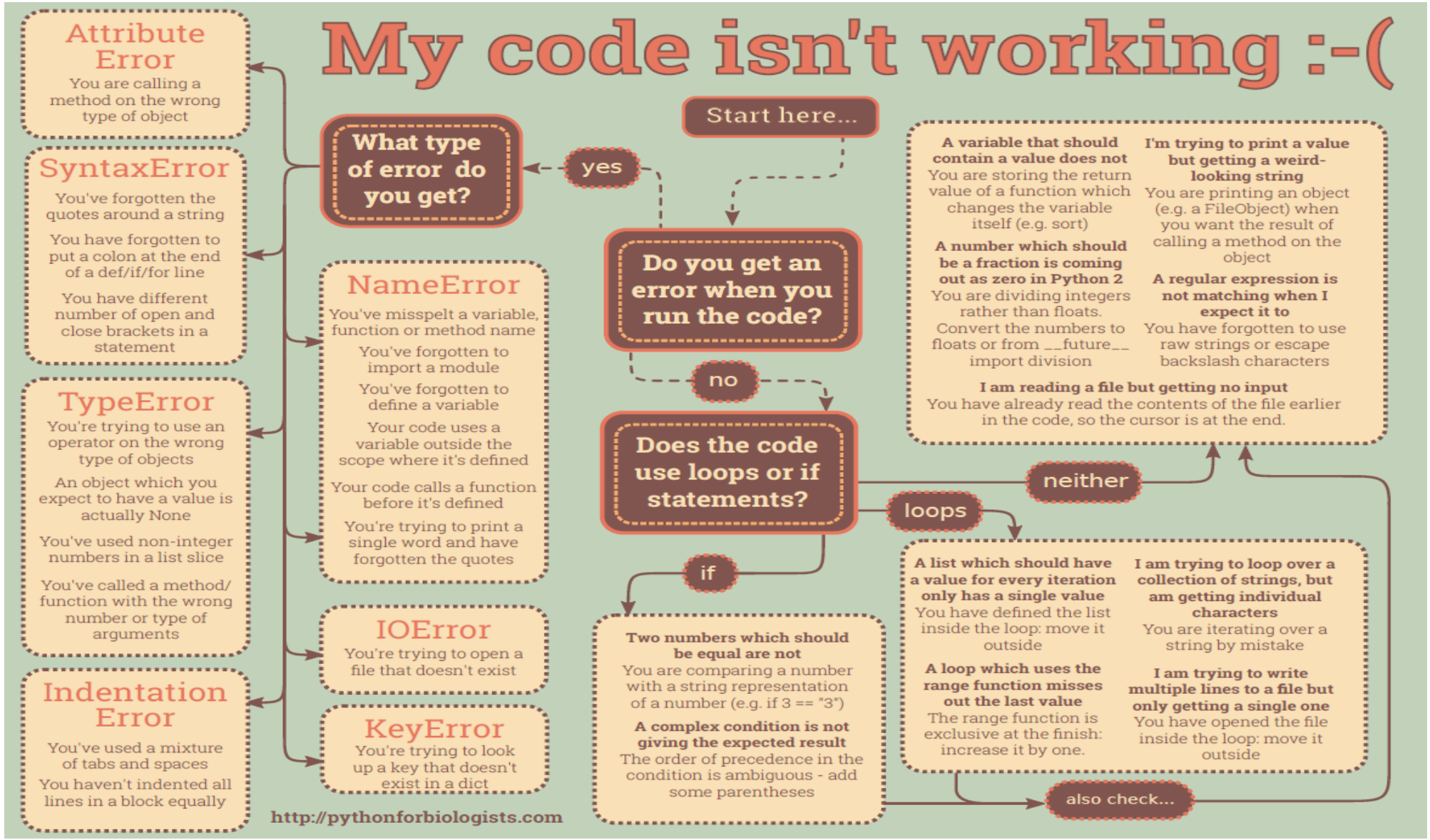




Git Bash

- Alternatively, to the graphical variant there is also the possibility to use git directly from the terminal
- To clone a repository from git type:
  - *git clone <repo> <directory>*
  - This will clone the repository at *<repo>* into the ~＜directory＞! folder on the local machine

- Example: *git clone ssh://john@example.com/path/to/my-project.git cd my-project*
- Important git commands can be found here: https://dzone.com/articles/top-20-git-commands-with-examples


My code isn't working :-(
Start here...
Do you get an error when you run the code?
yes
no
What type of error do you get?
Does the code use loops or if statements?
neither
loops
if
also check...
Attribute Error
You are calling a method on the wrong type of object
SyntaxError
You've forgotten the quotes around a string
You have forgotten to put a colon at the end of a def/if/for line
You have different number of open and close brackets in a statement
TypeError
You're trying to use an operator on the wrong type of objects
An object which you expect to have a value is actually None
You've used non-integer numbers in a list slice
You've called a method/ function with the wrong number or type of arguments
Indentation Error
You've used a mixture of tabs and spaces
You haven't indented all lines in a block equally
NameError
You've misspelt a variable, function or method name
You've forgotten to import a module
You've forgotten to define a variable
Your code uses a variable outside the scope where it's defined
Your code calls a function before it's defined
You're trying to print a single word and have forgotten the quotes
IOError
You're trying to open a file that doesn't exist
KeyError
You're trying to look up a key that doesn't exist in a dict
http://pythonforbiologists.com
Two numbers which should be equal are not
You are comparing a number with a string representation of a number (e.g. if 3 == "3")
A complex condition is not giving the expected result
The order of precedence in the condition is ambiguous - add some parentheses
A variable that should contain a value does not
You are storing the return value of a function which changes the variable itself (e.g. sort)
A number which should be a fraction is coming out as zero in Python 2
You are dividing integers rather than floats. Convert the numbers to floats or from __future__ import division
I'm trying to print a value but getting a weird-looking string
You are printing an object (e.g. a FileObject) when you want the result of calling a method on the object
A regular expression is not matching when I expect it to
You have forgotten to use raw strings or escape backslash characters
I am reading a file but getting no input
You have already read the contents of the file earlier in the code, so the cursor is at the end.
A list which should have a value for every iteration only has a single value
You have defined the list inside the loop: move it outside
A loop which uses the range function misses out the last value
The range function is exclusive at the finish: increase it by one.
I am trying to loop over a collection of strings, but am getting individual characters
You are iterating over a string by mistake
I am trying to write multiple lines to a file but only getting a single one
You have opened the file inside the loop: move it outside

Debugging

- The following website gives a very good overview for debugging Python code:
  - https://medium.com/techtofreedom/six-debugging-techniques-for-python-programmers-cb25a4baaf4b

- The following slides can help you as well:
  - https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec7.pdf

What happens next?

- Share ideas in your group
  - Find your strengths
  - Distribute the tasks
- We help you with problems
  - In the consultation hours
  - Or in the ILIAS Forum
- We wish you a lot of fun and success!

- Linux Commands: https://files.fosswire.com/2007/08/fwunixref.pdf