



ShanghaiTech University
上海科技大学

School of Information Science and Technology
信息科学与技术学院

Introduction to Information Science and Technology (IST)

Part IV: Intelligent Machines and Robotics

Intro to Robotics Practice

Sören Schwertfeger / 师泽仁

ShanghaiTech University

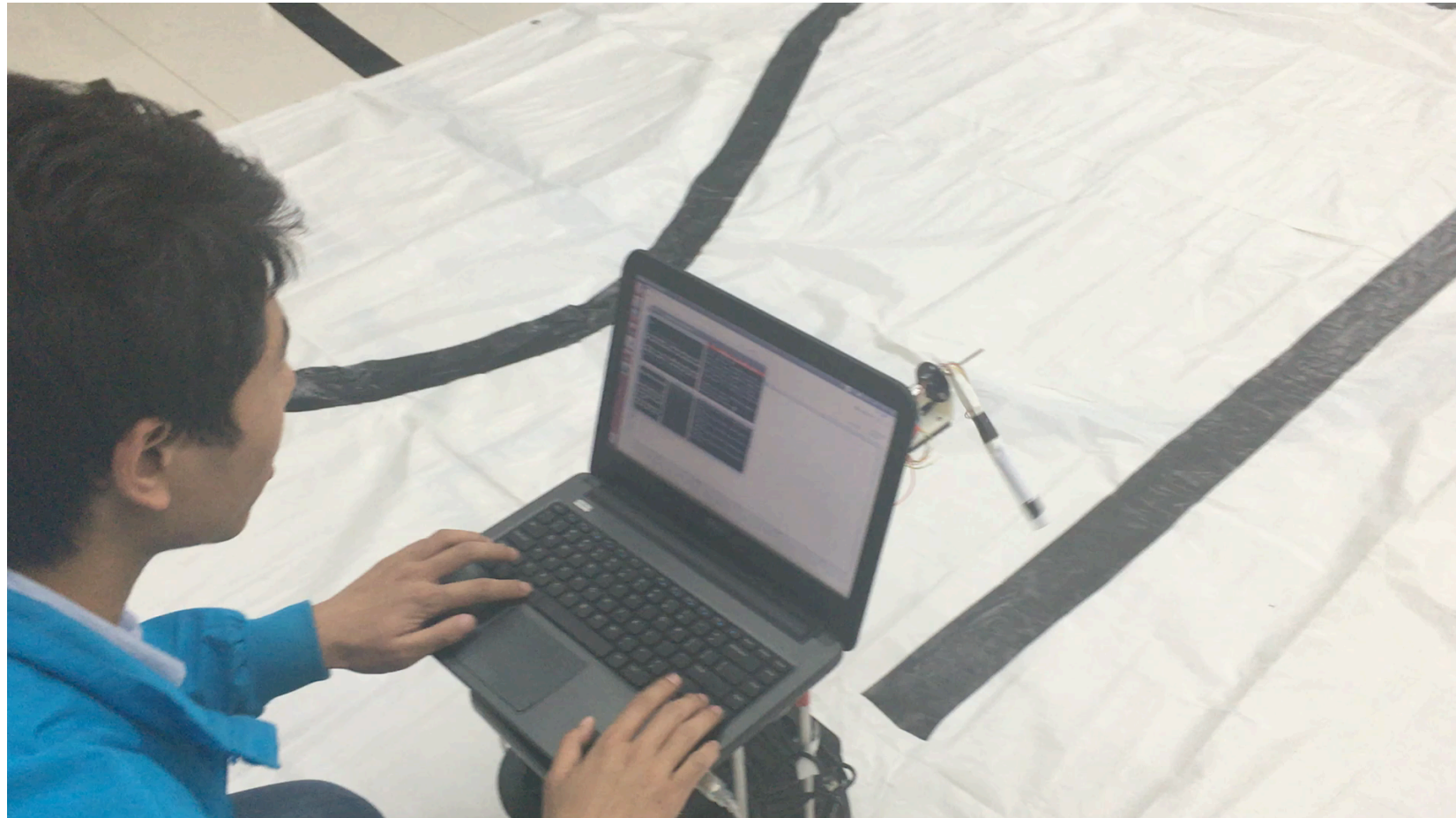
GOALS

Overall Goal

- Gain experience with a real robot
- Practice your
 - Programming skills
 - Problem solving skills
- Be creative
- Have fun!

Final Goal: Robot Race

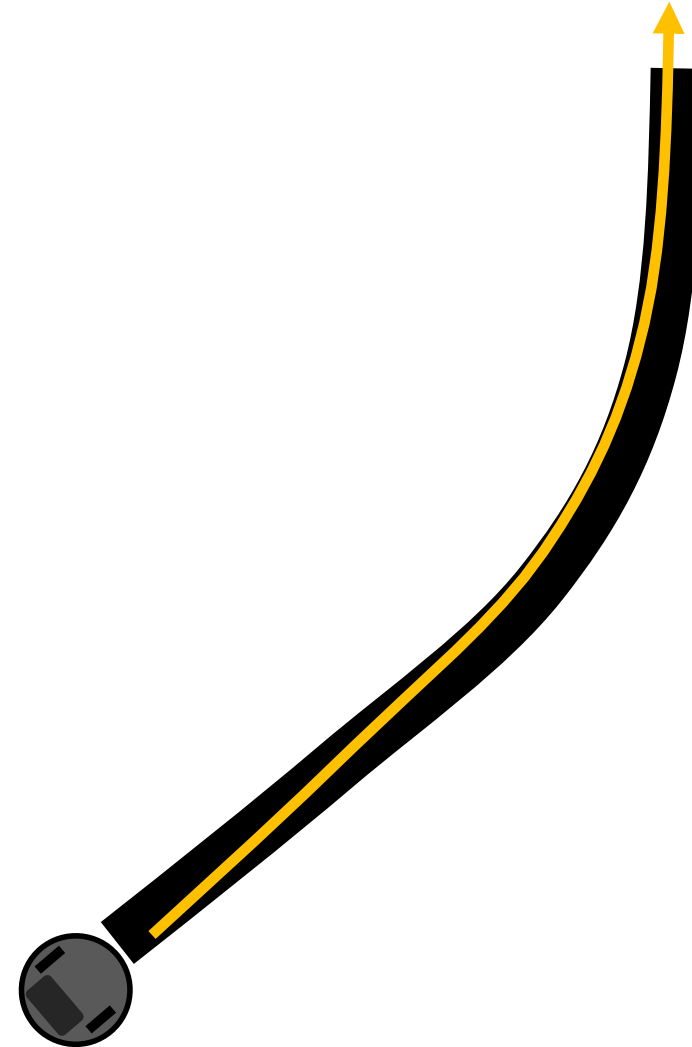
- Who can program the robot to reach the goal line fastest?
- Follow the black line
- Stop when the black line ends



Final Goal: Robot Race

- Who can program the robot to reach the goal line fastest?
- Follow the black line
- Stop when the black line ends

- Rules:
 - On final (3rd) practice you get two tries.
 - Time starts when robot crosses start line
 - Time finishes when robot crosses end line
 - Leave path: disqualified for this run



HARDWARE

TurtleBot

- Based on a vacuum cleaner platform (bottom)
- Kinect camera
- You can place your laptop on top

- Differential drive:
 - Two wheels (+2 passive support wheels)
 - Can rotate on the spot
 - Motions:
 - Translation: Forward/ backward: along x-axis
 - Rotation: Turn: around z-axis

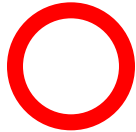



Connectors and Status

- Kinect power goes to 12V 1.5A
- Status LED:
 - Yellow: not ready
 - Green: ready
 - Blinking: charging



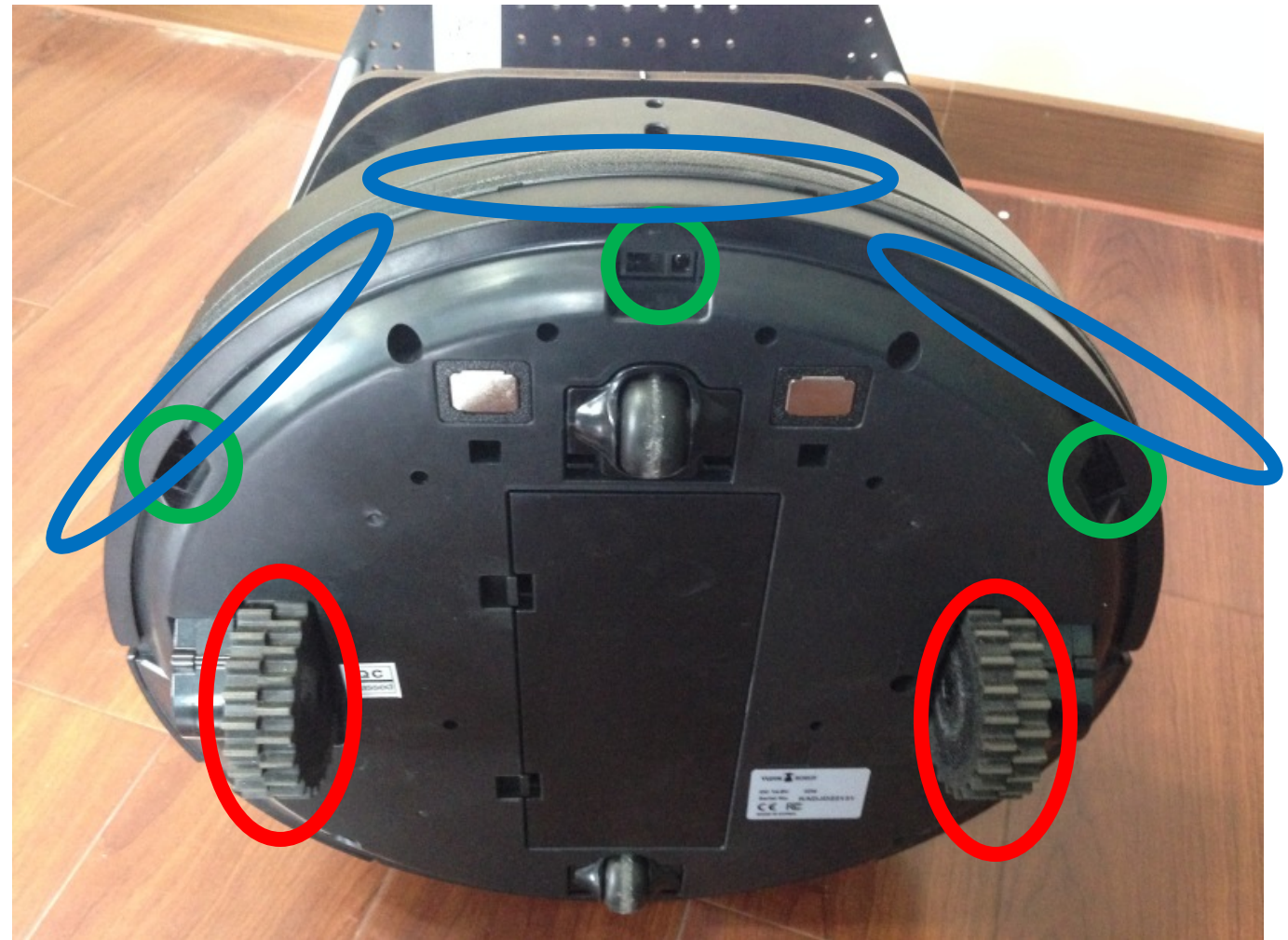
Power Button and Charging Plug

- Power button: 
- Plug for charger: 
- Do not (accidentally) take charger home with you!!!



Sensors

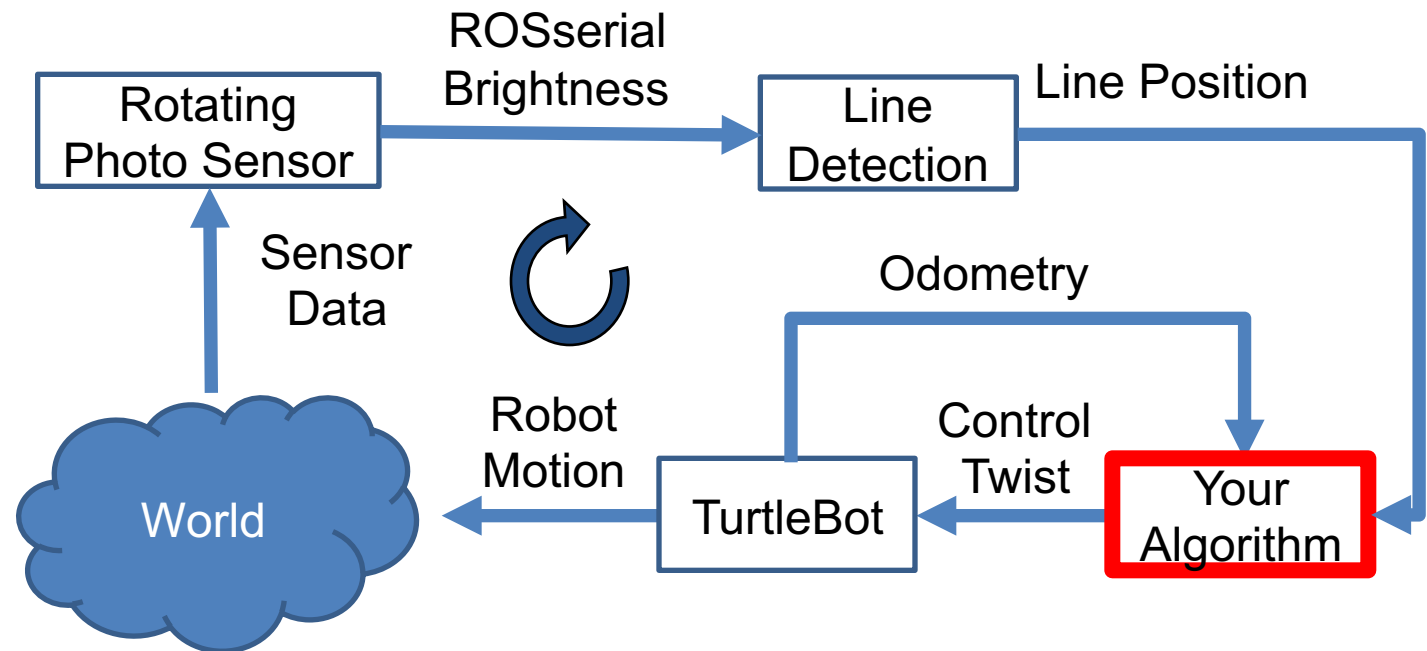
- Encoders (wheel speeds)
 - Wheel drop
- 3x cliff sensor (IR distance to ground)
- 3x bumpers
- Buttons
- Analog + digital input
- Current (power) of motors
- Charger connected



APPROACH

Control Loop

- Perception of the environment
- Create a model of the environment
- Decide where to drive next
- Actually drive

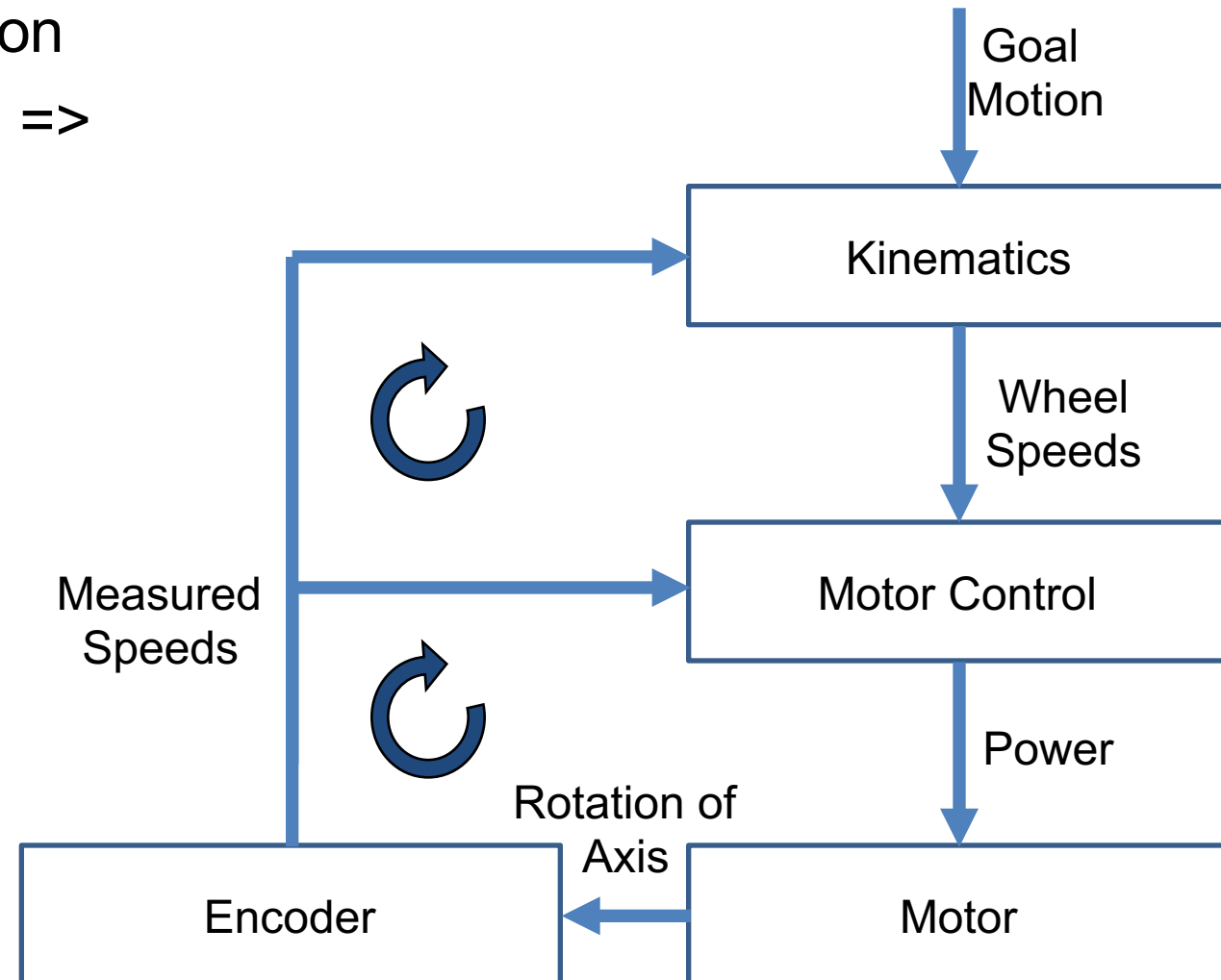


Low Level Control: Done!

- Assume we have a goal Motion
- Calculate Inverse Kinematics =>
- Desired wheel speeds

- Motor control loop
- Pose control loop

- All Done!



Your Algorithm

- Input:
 - Odometry:
 - Current global pose = position + orientation
 - Brightness and distance sensor value from Arduino
- Algorithm:
 - Find out where the line is
 - Find out that the line has ended
 - Compute the control output
- Output:
 - Where to drive next in terms of:
 - How fast to drive forward
 - How much to turn (and direction)

DATA TO USE

Data from Arduino

- Topic:
/ir_data
- Type: std_msgs/Int32
 - Encodes the values of both sensors in one 32 bit integer

```
rosmmsg show std_msgs/Int32  
int32 data
```


Command the Robot

- Topic:
/mobile_base/commands/velocity
- Type: geometry_msgs/Twist
 - Linear translation: only in x possible
 - Rotation: only around z possible
- Try to send command in console (use tab completion!)

```
roscat show geometry_msgs/Twist
geometry_msgs/Vector3 linear
float64 x
float64 y
float64 z
geometry_msgs/Vector3 angular
float64 x
float64 y
float64 z
```

```
schwerti@schwerti-ws:~ $ rostopic pub /mobile_base/commands/velocity geometry_msgs/Twist
"linear:
  x: 0.4
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0"
```

Odometry (Robot Pose)

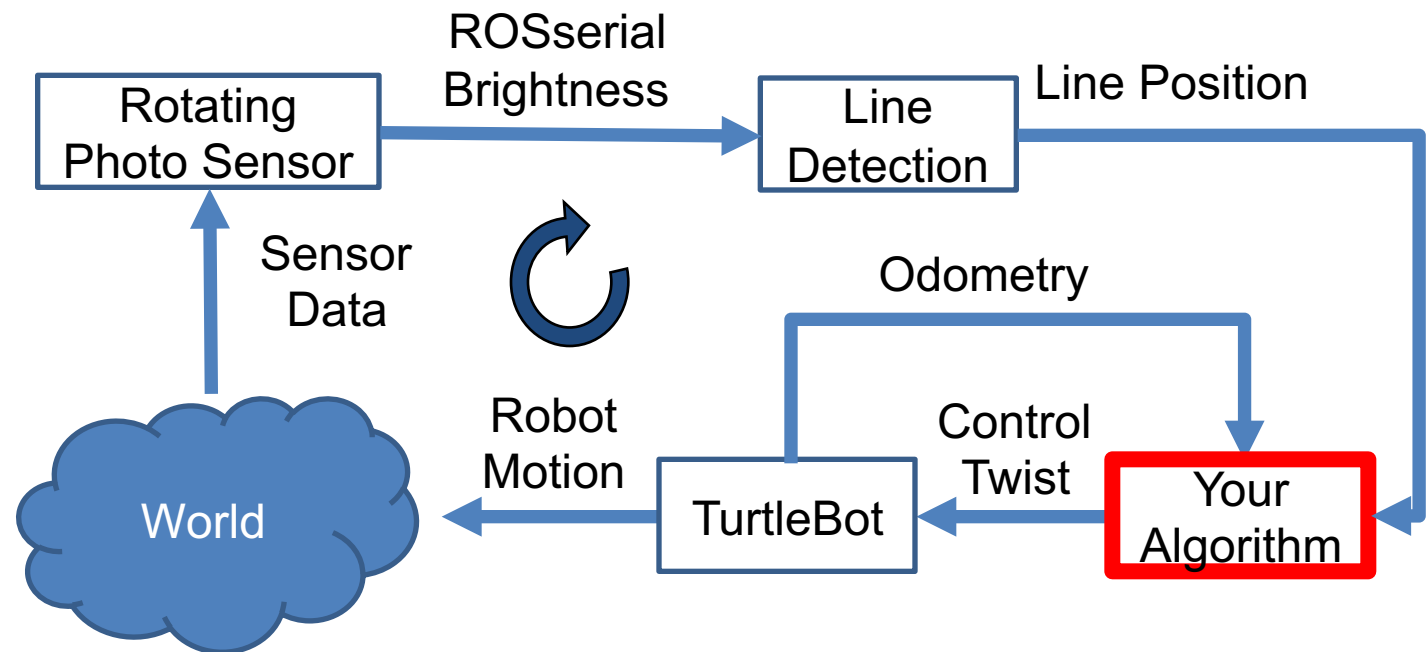
- Topic: /odom
- Type: nav_msgs/Odometry
- Current pose (position and orientation) of the robot
 - x, y: position (ignore z)
 - Orientation as a Quaternion:
 - Representation of 3D rotation
 - Code to extract the yaw can be provided

```
rosmmsg show nav_msgs/Odometry
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
string child_frame_id
geometry_msgs/PoseWithCovariance pose
  geometry_msgs/Pose pose
    geometry_msgs/Point position
      float64 x
      float64 y
      float64 z
    geometry_msgs/Quaternion orientation
      float64 x
      float64 y
      float64 z
      float64 w
float64[36] covariance
```

ALGORITHMS

Algorithm

- Detect Line Position
 - With respect to the robot
 - Maybe:
 - Smooth the signal
 - Use the distance sensor to find out where "down" is
- Behavior based control
 - The more the line is to the left the more turn left
 - The more the line is to the right the more turn right
 - Always drive forward – until you loose the line



Algorithm

- You can also implement an own idea!
- Algorithms look simple
- But the details are difficult
- Testing is very much needed =>
- Testing during practices 2. and 3.

TimeLine

- Practice 1: Get to know the robot and ROS; Try the perception software
- Homework 2: Implement your algorithm!
- Practice 2: Test your algorithm!
- Homework 3: Debug and improve your algorithm
- Practice 3: Final testing and RACE!