Project Ballbot

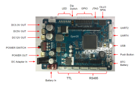
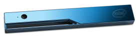
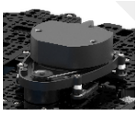
Markus Lamprecht
Florian Müller
Michael Suffel

TECHNISCHE UNIVERSITÄT DARMSTADT

TU Darmstadt, RTM

Inhaltsverzeichnis

1 Item - List

Item	#	W.[g]	Weblink	Picture
OpenCR Board (Controlling the motors, IMU)	1	60	*github_wiki*	
UpBoard (Main PC)	1	96	*127€*	
Intel RealSense R200	1	9.4	*datasheet, 84.15€*	
Laser Distance Sensor	1	124	*specs, 100€*	
Battery: LI-PO 11.1 1800mAh LB-12 19	1	132	*44.90€*	
Turtlebot3 Layers(125cmx125cm)	4			
XM430-W350-R Dynamixel (Motors)	3	82	*robotis,250€*	
Ball(alum., dia.: 140mm, material thickness 2.5mm)	1	400	*ball-tech gmbh,40€.*	
Omni wheels(dia: 60mm, thickness:25mm)	3	51.46	*10.38€*	
Kreisring (PLA, 3D printeted)	1	28		
Halterung (PLA, 3D printeted)	3	18		
Mitnehmer (PLA, 3D printeted)	3	8		
Plain washer (Beilagscheibe),(PLA, 3D printeted)	3	0.45		
M3 (Mutter-Halterung-Kreisring-Layer)	9			
M2.5 (Kreisring-Layer)	2			
M3x8mm Halterung	6		Zylinderkopf (Imbus)	
M3x22mm Layer	3	1.34	Zylinderkopf (Imbus)	
M2.5x22 (Motoren-Halterung)	12		Sechskant	
M2.5x38 (Motoren-Rad)	3		Zylinderkopf (Imbus)	
M2.5x24 (Layer)	2		Zylinderkopf (Imbus)	
M2x6mm (Mitnehmer-Motor)	12		Zylinderkopf (Imbus)	
Distanzbolzen	???		???	

Total Cost: 1176€ + Cost of opencr board and all plastic (incl. tb3 structure) and scrwes

Tabelle 1.1: Screws:

Type	Size	Amount	Place
Cylinderhead screw	M3 x 11mm	8	Motor mounts
Cylinderhead screw	M2,5 x 22mm	16	Motor plate
Cylinderhead screw	M2 x 6 mm	18	Wheel shaft
Cylinderhead screw	M2,5 x 36 mm (38 mm)	5	Wheel shaft cover
Cylinderhead screw	M3 x 20 mm (21mm)	4	Layer mounting
Nut	M2	5	Layer mounting
Cylinderhead screw	M2,5 x 22mm (23mm)	4	Layer mounting

2 Simulation

TODO: check if controller works
check why imu fails

2.1 Launch

These files are executed one after another:

1. bb_simulation: ballbot.launch
2. bb_description: bb_description.launch
3. bb_description -> urdf: bb.xacro
4. bb_description -> urdf: bb.urdf.xacro
5. bb_description -> urdf: common_properties.xacro
6. bb_description -> urdf: bb.gazebo.xacro

2.2 Gazebo - Controller Synchronization

The calculation of the 2D torques takes around: 0.000187 s.

Consider to take the right joint states as also all subwheel states are published.

Erst wenn das Motordrehmoment grosser als die wheel joint friction ist, bewegt sich das rad!.

Denke daran die joint states position und velocity in einen winkel und winkelgeschwindigkeit umzurechnen.

The state of each joint (revolute or prismatic) is defined by:
the position of the joint (rad or m),
the velocity of the joint (rad/s or m/s) and
the effort that is applied in the joint (Nm or N).

Muss ich nun in winkelgeschwindigkeit umrechnen oder nicht?!

Das seltsame: Ich kann keinen kontinuierlichen effort draufgeben. UnabhÃďngig von der publish rate, es wird of 3 mal eine 1 drauf gegeben und anschlieÃ§end eine 0.

Groesstes Problem bleibt: Die motor commands (torques) koennen nicht kontinuierlich rausgeschickt werden. Es werden immer wieder 0 rausgschickt ..... - warum ist das so?

use_sim_time parameter: ros time is the same as simulation time when the use_sim_time parameter is enabled.

In order for a ROS node to use simulation time according to the /clock topic, the use_sim_timeparameter must be set to true before the node is initialized. This can be done in a launchfile or from the command line.

If the use_sim_time time parameter is set, the ROS Time API will return time=0 until it has received a value from the /clock topic. Then, the time will only be updated on receipt of a message from the /clock topic, and will stay constant between updates.

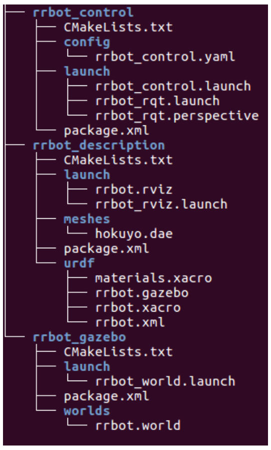
For calculations of time durations when using simulation time, clients should always wait until the first non-zero time value has been received before starting, because the first simulation time value from /clock topic may be a high value.

Note: Prior to ROS C Turtle, nodes were automatically subscribed to the /clock topic, and would use simulation time if there was anything published to the /clock topic.

2.3 Simulation design

Ballbot SDF Reference: *Ballbotmodel*

We use not the sdf but the xacro description as in this example *here*.



Gazebo uses different physics engines:

- Open Dynamics Engine (ODE) (Default)
- Bullet
- Dynamic Animation and Robotics Toolkit (DART)
- Simbody

which all have different friction etc. models.

Files:

- bb.urdf.xacro: Link's: Visual description of the Robot and its collision model(STL file). Pose Mass and Inertias. Joint's: Pose,axis,effort and velocity limits, friction.
- common_properties.xacro: Macros for color definition.
- bb.gazebo.xacro: gazebo references dynamics of the links: friction parameters (mu1,mu2),

Gazebo Parameter's List:

name(xacro)	description	value	sdf group
mu1	is the Coulomb friction coefficient for the first friction direction	1.0	ode
mu2	is the friction coefficient for the second friction direction (perpendicular to the first friction direction)	2.0	ode
fdir1	3-tuple specifying direction of mu1 in the collision local reference frame. fdir1 is the vector that defines the direction of mu1, which is the principal contact direction	0 0 0	ode
kp	spring constant equivalents of a contact as a function of SurfaceParams::cfm and SurfaceParams::erp		ode
kd	spring damping constant equivalents of a contact as a function of SurfaceParams::cfm and SurfaceParams::erp.		ode
cfm	Constraint Force Mixing parameter.		ode
erp	Error Reduction Parameter.		ode
min_depth	Minimum depth before ERP takes effect.		ode
max_Vel	Maximum interpenetration error correction velocity. If set to 0, two objects interpenetrating each other will not be pushed apart.		ode
slip1	Artificial contact slip in the primary friction direction		ode
slip2	Artificial contact slip in the secondary friction dirction.		ode

See: *ODESurfaceParams*

Urdf Parameter's List: JOINT TAGS:

name(xacro tag)	description	value
axis	this is the axis around the joint is revolting or linear	xyz="0 1 0"
dynamics	set the friction and the damping	friction="0.7" damping = "0.0"

2.4 Gazebo Parameters

2.5 How to model an omni wheel gazebo:

1. `http://answers.gazebosim.org/question/5562/modeling-omni-wheels/`
2. `http://answers.gazebosim.org/question/5562/modeling-omni-wheels/`
3. `http://answers.gazebosim.org/question/5476/parameters-for-a-skid-steeringsimulated-tracke`
4. `https://bitbucket.org/osrf/gazebo/pull-requests/2652/added-support-for-tracked-vehicles/diff`
5. `https://bitbucket.org/osrf/gazebo/pull-requests/2652/added-support-for-tracked-vehicles/diff`
6. `https://bitbucket.org/osrf/gazebo/issues/2068/directional-friction-still-broken`
7. `https://answers.ros.org/question/212889/gazebo-planar-move-plugin-for-omni-directional-wh`

   For instance, the PR2 and Care-O-Bot are omnidirectional drive robots available for simulation in gazebo. Both use a system of four steered and driven casters (for a total of 8 motors) to achieve omnidirectional mobility. If you're interested in simulation of a meccanum-wheel drive robot, I'm not sure there is one available for gazebo. Last time I looked, the youbot for gazebo used no true meccanum wheels, but a similar system to the two robots I mentioned above.

2.6 omni wheel controllers

1. libgazebo_ros_skid_steer_drive.so https://github.com/fsuarez6/labrob/blob/master/labrob_description/urdf/labrob.urdf.xacro

2.7 Equations for Controller:

$\vartheta_{x,y,z}$ represent the orientation of the body
$\varphi_{x,y,z}$ represent the orientation of the ball
$\psi_{x,y,z}$ are the angles of the virtual actuating wheels.
$T_{x,y,z}$ are the virtual motor torques.
$T_{1,2,3}$ are the real motor torques. Inputs:

1. The Gain Matrix K derived from the Simulink Simulation
2. IMU-Measurements: $\vartheta_{x,y,z}(rad), \quad \dot{\vartheta}_{x,y,z}(rad/sec)$
3. Motor-Measurements(rotation of actuating wheel): $\psi_{x,y,z}(rad), \quad \dot{\psi}_{x,y,z}(rad/sec)$

Wie von virtual wheels auf real wheels umrechnen?!

2.8 Control

sobald diff drive plugin angeschaltet drehen sich die raeder viel zu schnell ....
Diff Drive in ballbot.launch an oder ausschalten.
in bb.gazebo.xacro transmission und controller festlegen.
zudem yaml file(currently I use: effort_controllers/JointVelocityController)
Effort Joint Interface as Hardware Interface is used.
Do this example first: http://gazebosim.org/tutorials/?tut=ros_control
Also try this bb8 gazebo tutorial: https://www.youtube.com/watch?v=j5qC9l448p8

2.8.1 Plugins

- gazebo-ros-control
- diff drive

2.8.2 Launch

```
roslaunch rrbot_control rrbot_control.launch
```

These files are executed one after another:

1. load config
2. controller_spawner

2.9 Sensors

2.9.1 IMU

We want to simulate the IMU of the opencr board. STRG+T to see imu topic values! *Imu of opencr board simulated*

Simulate like this: rviz rviz dann als fixed frame nimm: imu_link. Und add topic imu und waehle als topic ballbot/sensor/imu

The simulated IMU outputs values like: orientation (x,y,z,w), angluar velocity(x,y,z), linear velocity(x,y,z), linear acceleration(x,y,z).

The opencr real IMU gives values like: orientation(x,y,z,w), angular velocity(x,y,z), linear acceleration(x,y,z) see *http://turtlebot3.readthedocs.io/en/latest/appendix_opencr.html*

3 Model

3.1 Composition

The Ballbot consists of three parts, which are depicted in Figure 3.1.

- Body with motors
- 3 omni-directional wheels
- Ball

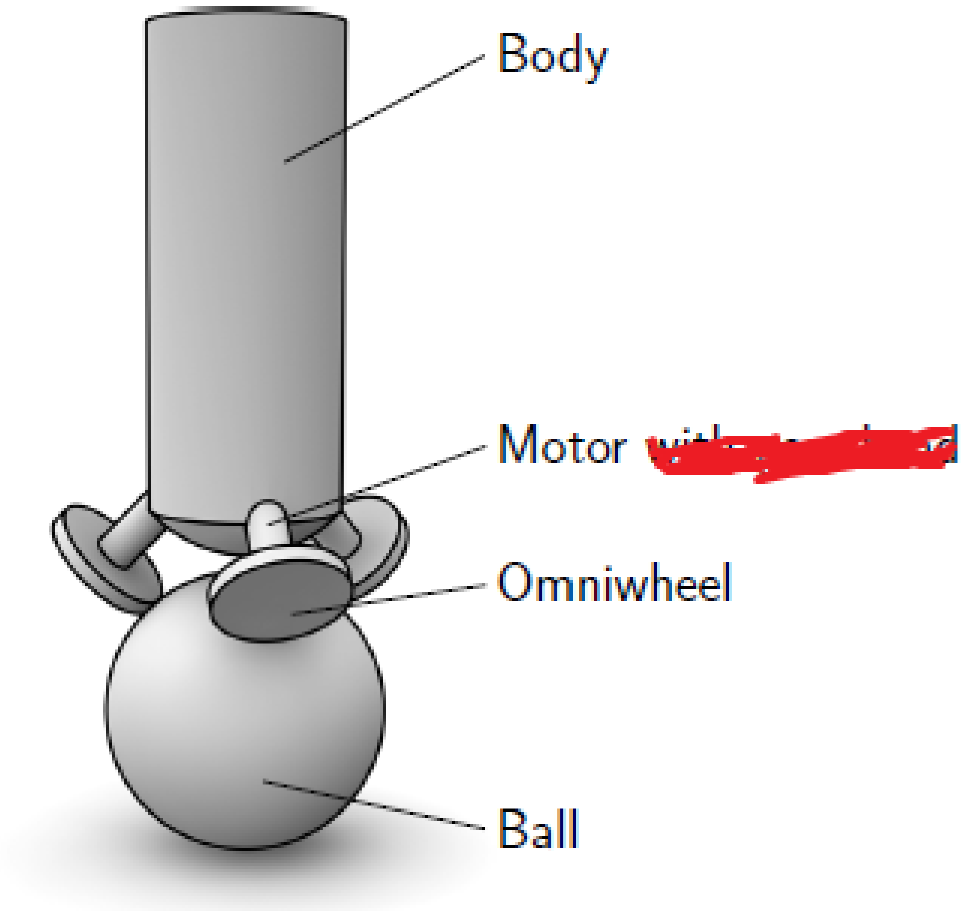


Abbildung 3.1: Parts for the 3D-Model

3.2 Assumptions

To reduce the complexity of the system, the following assumptions are made:

- **No slip** between the contact points between the ball/ground and wheels/ball
- **No friction**; except the friction, which occurs at the rotation of the ball around the z-axis
- **No deformation**
- **Fast motor dynamics**; The controlling of the motor is much faster than the controller of the Ballbot
- **Ball moves only horizontal**

3.3 TODO

1. Horn komplett rein auf beilagscheibe und schauen dass das mit der 0 position stimmt!

3.4 Model Parameters

Traegheitsmoment kugel Hohlzylinder: $J = m\frac{r_i^2+r_a^2}{2} = 0.4 * \frac{65^2+70^2}{2}kg * 10^{-6}m^2 = 1.825 * 10^{-3}kgm^2$
Traegheitsmoment wheel Vollzylinder: $J = m\frac{1}{2}r^2 = 0.05146 * 0.020^2 = 1.0292 * 10^{-4}kgm^2$

Tabelle 3.1: My caption

Parameter	Variable	Value	Source
Mass of the ball	m_K	0,4 kg	Datasheet
Mass of the ball	m_K	0,397 kg	Measured
Mass of the ball	m_K	0,631 kg	SolidEdge
Mass of the body, complete (with motors/wheels)	m_B	?	Measured
Mass of the body, complete (with motors/wheels)	m_B	1,785 kg	SolidEdge
Mass of the body (without motors/wheels)	m_B	?	Measured
Mass of the body (without motors/wheels)	m_B	1,394 kg	SolidEdge
Mass of Omniwheel	m_{OW}	0,050 kg	Measured
Mass of Omniwheel	m_{OW}	0,046 kg	SolidEdge
Mass of the virtual wheel	m_{VW}	0,384 kg	Measured
Mass of the substructure, complete (with motors/wheels)	m_S	0,506 kg	Measured
Mass of the substructure, complete (with motors/wheels)	m_S	0,457 kg	SolidEdge
Mass of plate	m_P	0,078 kg	Measured
Mass of plate	m_P	?	SolidEdge
Radius of the ball	r_K	0,07 m	Datasheet
Radius of the body	r_B	0,0703 m	Measured
Radius of the Wheels	r_W	0,03 m	Datasheet
Height of the center of gravity	l	0,24045 m	SolidEdge
Height of the body	h	0,34294 m	SolidEdge
Inertia of the Ball	Θ_K	0,00131 kgm^2	Computed
Inertia of the Body (x-axis)	Θ_{Bx}	0,08751 kgm^2	SolidEdge
Inertia of the Body (y-axis)	Θ_{By}	0,08788 kgm^2	SolidEdge
Inertia of the body (z-axis)	Θ_{Bz}	0,00329 kgm^2	SolidEdge
Inertia of the body (xy plane)	Θ_{Bxy}	-0,00001 kgm^2	SolidEdge
Inertia of the body (xz plane)	Θ_{Bxz}	0,00203 kgm^2	SolidEdge
Inertia of the body(zy plane)	Θ_{Bzy}	0,00018 kgm^2	SolidEdge
Inertia of the rotor (motor)	Θ_M	0,444e-6 kgm^2	Adoption
Inertia of Omniwheel	Θ_{OW}	0.000023157 kgm^2	Computed
Inertia of the actuating wheel in yz/xz	Θ_W	0,058873 kgm^2	Computed
Inertia of the actuating wheel in xy	Θ_{Wxy}	0,16656 kgm^2	Computed
Gear ratio	i	353,5	Datasheet
Gravitational acceleration	g	9,81 m/s^2	BachelorThesis