

Mumei

v0.1

Generated by Doxygen 1.9.5



<b>1 Mumei</b>	<b>1</b>
1.1 1. Running with Docker:	1
1.1.1 Prerequisites:	1
1.1.2 Steps:	1
1.2 2. Building from source:	1
1.2.1 Prerequisites:	1
1.2.2 Build and install:	2
1.2.3 Uninstall:	2
<b>2 HURON ROS2 Module</b>	<b>3</b>
2.0.1 Usage:	3
2.0.1.1 Building the ROS Workspace:	3
2.0.1.2 Launching Gazebo 11 simulation	3
<b>3 Namespace Index</b>	<b>5</b>
3.1 Namespace List	5
<b>4 Hierarchical Index</b>	<b>7</b>
4.1 Class Hierarchy	7
<b>5 Class Index</b>	<b>9</b>
5.1 Class List	9
<b>6 File Index</b>	<b>11</b>
6.1 File List	11
<b>7 Namespace Documentation</b>	<b>13</b>
7.1 huron Namespace Reference	13
7.1.1 Detailed Description	14
<b>8 Class Documentation</b>	<b>15</b>
8.1 huron::Actuator Class Reference	15
8.2 huron::ActuatorConfiguration Class Reference	15
8.3 huron::driver::can::BusBase Class Reference	16
8.3.1 Member Function Documentation	16
8.3.1.1 recv_message()	17
8.3.1.2 send_message()	17
8.3.1.3 subscribe()	17
8.3.1.4 unsubscribe()	17
8.4 can_Cyclic_t Struct Reference	17
8.5 can_Message_t Struct Reference	18
8.6 can_Signal_t Struct Reference	18
8.7 huron::driver::can::BusBase::CanSubscription Struct Reference	18
8.8 huron::multibody::ComFrame Class Reference	18
8.8.1 Detailed Description	19

8.8.2 Member Function Documentation	19
8.8.2.1 GetTransformFromFrame() [1/2]	19
8.8.2.2 GetTransformFromFrame() [2/2]	19
8.8.2.3 GetTransformInWorld()	19
8.8.2.4 GetTransformToFrame() [1/2]	19
8.8.2.5 GetTransformToFrame() [2/2]	19
8.9 huron::Configuration Class Reference	20
8.9.1 Detailed Description	20
8.9.2 Member Function Documentation	20
8.9.2.1 Get()	20
8.9.2.2 GetFromComponent()	20
8.9.2.3 ValidateKey()	21
8.10 huron::ConstantStateProvider Class Reference	21
8.10.1 Member Function Documentation	21
8.10.1.1 GetNewState()	21
8.10.1.2 RequestStateUpdate()	21
8.11 huron::enable_protected_make_shared< ClassWithProtectedCtor > Class Template Reference	21
8.11.1 Detailed Description	22
8.12 huron::enable_protected_make_unique< ClassWithProtectedCtor > Class Template Reference	22
8.12.1 Detailed Description	22
8.13 huron::Encoder Class Reference	22
8.13.1 Detailed Description	23
8.13.2 Member Function Documentation	23
8.13.2.1 GetNewState()	23
8.13.2.2 GetPosition()	23
8.13.2.3 GetVelocity()	23
8.13.2.4 Reset()	24
8.14 huron::EncoderConfiguration Class Reference	24
8.15 huron::ForceSensingResistor Class Reference	24
8.16 huron::ForceSensingResistorArray Class Reference	25
8.16.1 Member Function Documentation	25
8.16.1.1 GetNewState()	25
8.16.1.2 RequestStateUpdate()	26
8.17 huron::ForceSensingResistorArraySerial Class Reference	26
8.17.1 Detailed Description	26
8.17.2 Member Function Documentation	26
8.17.2.1 GetValue()	27
8.17.2.2 Initialize()	27
8.17.2.3 ReloadAndGetValue()	27
8.17.2.4 RequestStateUpdate()	27
8.17.2.5 SetUp()	27
8.17.2.6 Terminate()	27

8.18 <code>huron::ForceTorqueSensor</code> Class Reference	27
8.18.1 Member Function Documentation	28
8.18.1.1 <code>DoGetWrenchRaw()</code>	28
8.18.1.2 <code>GetNewState()</code>	28
8.18.1.3 <code>GetValue()</code>	28
8.18.1.4 <code>RequestStateUpdate()</code>	28
8.19 <code>huron::ros2::ForceTorqueSensor</code> Class Reference	29
8.19.1 Member Function Documentation	29
8.19.1.1 <code>DoGetWrenchRaw()</code>	29
8.19.1.2 <code>Initialize()</code>	29
8.19.1.3 <code>SetUp()</code>	29
8.19.1.4 <code>Terminate()</code>	30
8.20 <code>huron::multibody::Frame</code> Class Reference	30
8.21 <code>huron::GenericComponent</code> Class Reference	31
8.21.1 Detailed Description	31
8.21.2 Member Function Documentation	31
8.21.2.1 <code>Configure()</code> [1/3]	31
8.21.2.2 <code>Configure()</code> [2/3]	32
8.21.2.3 <code>Configure()</code> [3/3]	32
8.21.2.4 <code>ConfigureKey()</code>	32
8.21.2.5 <code>ConfigureMap()</code>	32
8.21.2.6 <code>Initialize()</code>	32
8.22 <code>huron::ros2::Huron</code> Class Reference	32
8.22.1 Member Function Documentation	33
8.22.1.1 <code>Initialize()</code>	33
8.22.1.2 <code>SetUp()</code>	33
8.22.1.3 <code>Terminate()</code>	33
8.23 <code>huron::ros2::HuronNode</code> Class Reference	33
8.24 <code>huron::multibody::internal::PinocchioModelImpl::Impl</code> Struct Reference	34
8.25 <code>huron::InvalidConfigurationException</code> Class Reference	34
8.26 <code>huron::Joint</code> Class Reference	35
8.26.1 Constructor & Destructor Documentation	35
8.26.1.1 <code>Joint()</code>	35
8.27 <code>huron::multibody::JointDescription</code> Struct Reference	35
8.28 <code>huron::ros2::JointGroupController</code> Class Reference	36
8.28.1 Member Function Documentation	36
8.28.1.1 <code>Move()</code> [1/2]	36
8.28.1.2 <code>Move()</code> [2/2]	36
8.28.1.3 <code>Stop()</code>	37
8.29 <code>huron::ros2::JointStateProvider</code> Class Reference	37
8.29.1 Member Function Documentation	37
8.29.1.1 <code>GetNewState()</code>	37

8.29.1.2 RequestStateUpdate()	38
8.30 huron::LeggedRobot Class Reference	38
8.30.1 Member Function Documentation	38
8.30.1.1 EvalZeroMomentPoint()	38
8.31 huron::Limb Class Reference	38
8.32 huron::multibody::LogicalFrame Class Reference	39
8.32.1 Detailed Description	39
8.32.2 Member Function Documentation	40
8.32.2.1 GetTransformFromFrame() [1/2]	40
8.32.2.2 GetTransformFromFrame() [2/2]	40
8.32.2.3 GetTransformInWorld()	40
8.32.2.4 GetTransformToFrame() [1/2]	40
8.32.2.5 GetTransformToFrame() [2/2]	40
8.33 huron::multibody::Model Class Reference	40
8.33.1 Member Function Documentation	42
8.33.1.1 AddFrame()	42
8.33.1.2 AddJoint()	43
8.33.1.3 AddModelImpl()	43
8.33.1.4 Finalize()	43
8.33.1.5 GetModelImpl()	43
8.33.1.6 UpdateJointStates()	44
8.34 huron::multibody::ModelComposite Class Reference	44
8.35 huron::multibody::internal::ModelImplFactory Class Reference	44
8.36 huron::multibody::internal::ModelImplInterface Class Reference	44
8.36.1 Member Function Documentation	45
8.36.1.1 GetAccelerations()	46
8.36.1.2 GetCentroidalMatrix()	46
8.36.1.3 GetCentroidalMomentum()	46
8.36.1.4 GetCoriolisMatrix()	46
8.36.1.5 GetGravity()	46
8.36.1.6 GetMassMatrix()	46
8.36.1.7 GetNonlinearEffects()	46
8.36.1.8 GetSpatialMomentum()	46
8.36.1.9 GetTorques()	47
8.37 huron::Motor Class Reference	47
8.38 huron::MotorConfiguration Class Reference	47
8.39 huron::MovingGroup Class Reference	48
8.39.1 Member Function Documentation	48
8.39.1.1 Move() [1/2]	49
8.39.1.2 Move() [2/2]	49
8.39.1.3 Stop()	49
8.40 huron::MovingInterface Class Reference	49

8.40.1 Detailed Description	50
8.40.2 Member Function Documentation	50
8.40.2.1 Move()	50
8.40.2.2 Stop()	50
8.41 huron::driver::can::MsgIdFilterSpecs Struct Reference	50
8.42 huron::NotImplementedException Class Reference	51
8.43 huron::odrive::ODrive Class Reference	51
8.43.1 Detailed Description	52
8.43.2 Member Function Documentation	52
8.43.2.1 Calibrate()	52
8.43.2.2 ConfigureKey()	52
8.43.2.3 Initialize()	53
8.44 huron::odrive::ODriveCAN Class Reference	53
8.44.1 Constructor & Destructor Documentation	54
8.44.1.1 ODriveCAN()	54
8.44.2 Member Function Documentation	54
8.44.2.1 ClearErrors()	55
8.44.2.2 Estop()	55
8.44.2.3 GetAdcVoltage()	55
8.44.2.4 GetBusVoltageCurrent()	55
8.44.2.5 GetControllerError()	55
8.44.2.6 GetEncoderCount()	55
8.44.2.7 GetEncoderError()	55
8.44.2.8 GetEncoderEstimates()	55
8.44.2.9 GetIq()	56
8.44.2.10 GetMotorError()	56
8.44.2.11 GetSensorlessError()	56
8.44.2.12 GetSensorlessEstimates()	56
8.44.2.13 Nmt()	56
8.44.2.14 SetAxisNodeid()	56
8.44.2.15 SetAxisRequestedState()	56
8.44.2.16 SetAxisStartupConfig()	56
8.44.2.17 SetControllerModes()	57
8.44.2.18 SetInputPos()	57
8.44.2.19 SetInputTorque()	57
8.44.2.20 SetInputVel()	57
8.44.2.21 SetLimits()	57
8.44.2.22 SetLinearCount()	57
8.44.2.23 SetPosGain()	57
8.44.2.24 SetTrajAccelLimits()	57
8.44.2.25 SetTrajInertia()	58
8.44.2.26 SetTrajVelLimit()	58

8.44.2.27 Setup()	58
8.44.2.28 SetVelGains()	58
8.44.2.29 StartAnticogging()	58
8.44.2.30 Terminate()	58
8.45 huron::odrive::ODrive::ODriveConfiguration Class Reference	58
8.45.1 Constructor & Destructor Documentation	59
8.45.1.1 ODriveConfiguration() [1/2]	59
8.45.1.2 ODriveConfiguration() [2/2]	59
8.46 huron::odrive::ODriveEncoder Class Reference	59
8.46.1 Member Function Documentation	60
8.46.1.1 DoUpdateState()	60
8.46.1.2 Initialize()	60
8.46.1.3 Setup()	60
8.46.1.4 Terminate()	60
8.47 huron::multibody::internal::PinocchioModelImpl Class Reference	60
8.47.1 Member Function Documentation	62
8.47.1.1 BuildFromUrdf()	62
8.47.1.2 ComputeAll()	62
8.47.1.3 EvalCenterOfMassPosition()	62
8.47.1.4 ForwardKinematics() [1/3]	62
8.47.1.5 ForwardKinematics() [2/3]	62
8.47.1.6 ForwardKinematics() [3/3]	62
8.47.1.7 GetAccelerations()	62
8.47.1.8 GetCenterOfMassPosition()	63
8.47.1.9 GetCentroidalMatrix()	63
8.47.1.10 GetCentroidalMomentum()	63
8.47.1.11 GetCoriolisMatrix()	63
8.47.1.12 GetFrameIndex()	63
8.47.1.13 GetFrameName()	63
8.47.1.14 GetFrameTransform()	63
8.47.1.15 GetFrameTransformInWorld()	63
8.47.1.16 GetFrameType()	64
8.47.1.17 GetGravity()	64
8.47.1.18 GetJoint() [1/2]	64
8.47.1.19 GetJoint() [2/2]	64
8.47.1.20 GetJointDescription() [1/2]	64
8.47.1.21 GetJointDescription() [2/2]	64
8.47.1.22 GetJointIndex()	64
8.47.1.23 GetJointNames()	64
8.47.1.24 GetJointTransformInWorld()	65
8.47.1.25 GetJointType()	65
8.47.1.26 GetMassMatrix()	65



8.47.1.27 GetNonlinearEffects()	65
8.47.1.28 GetSpatialMomentum()	65
8.47.1.29 GetTorques()	65
8.47.1.30 is_built()	65
8.47.1.31 NeutralConfiguration()	65
8.47.1.32 num_frames()	66
8.47.1.33 num_joints()	66
8.47.1.34 num_positions()	66
8.47.1.35 num_velocities()	66
8.48 huron::PositionMotor Class Reference	66
8.49 huron::PositionMotorConfiguration Class Reference	67
8.49.1 Constructor & Destructor Documentation	67
8.49.1.1 PositionMotorConfiguration()	67
8.50 PushRecoveryControl Class Reference	67
8.51 huron::Robot Class Reference	67
8.51.1 Member Function Documentation	68
8.51.1.1 UpdateAllStates()	68
8.51.1.2 UpdateJointStates()	68
8.52 huron::RobotConfiguration Class Reference	69
8.53 huron::RotaryEncoder Class Reference	69
8.53.1 Detailed Description	70
8.53.2 Member Function Documentation	70
8.53.2.1 DoUpdateState()	70
8.53.2.2 GetAngleDegree()	70
8.53.2.3 GetCount()	70
8.53.2.4 GetCPR()	70
8.53.2.5 GetPosition()	70
8.53.2.6 GetPrevCount()	70
8.53.2.7 GetVelocity()	71
8.53.2.8 GetVelocityCount()	71
8.53.2.9 GetVelocityDegree()	71
8.53.2.10 RequestStateUpdate()	71
8.53.2.11 Reset()	71
8.54 huron::RotaryEncoderConfiguration Class Reference	71
8.54.1 Constructor & Destructor Documentation	71
8.54.1.1 RotaryEncoderConfiguration()	72
8.55 huron::Sensor Class Reference	72
8.55.1 Member Function Documentation	72
8.55.1.1 GetValue()	72
8.56 huron::SensorWithFrame Class Reference	72
8.57 huron::driver::serial::Serial Class Reference	73
8.57.1 Member Function Documentation	74

8.57.1.1 Available()	74
8.57.1.2 Close()	74
8.57.1.3 Flush()	74
8.57.1.4 FlushInput()	74
8.57.1.5 FlushOutput()	74
8.57.1.6 GetBaudrate()	75
8.57.1.7 GetFlowcontrol()	75
8.57.1.8 GetParity()	75
8.57.1.9 GetPort()	75
8.57.1.10 GetStopbits()	75
8.57.1.11 IsOpen()	75
8.57.1.12 Open()	75
8.57.1.13 Read() [1/4]	75
8.57.1.14 Read() [2/4]	75
8.57.1.15 Read() [3/4]	76
8.57.1.16 Read() [4/4]	76
8.57.1.17 ReadLine() [1/2]	76
8.57.1.18 ReadLine() [2/2]	76
8.57.1.19 ReadLines()	76
8.57.1.20 SendBreak()	76
8.57.1.21 SetBaudrate()	76
8.57.1.22 SetFlowcontrol()	76
8.57.1.23 SetParity()	77
8.57.1.24 SetPort()	77
8.57.1.25 SetStopbits()	77
8.57.1.26 SetTimeout()	77
8.57.1.27 WaitReadable()	77
8.57.1.28 Write() [1/3]	77
8.57.1.29 Write() [2/3]	77
8.57.1.30 Write() [3/3]	77
8.58 huron::driver::serial::SerialBase Class Reference	78
8.59 huron::driver::can::SocketCanBus Class Reference	79
8.59.1 Member Function Documentation	79
8.59.1.1 recv_message()	80
8.59.1.2 send_message()	80
8.59.1.3 subscribe()	80
8.59.1.4 unsubscribe()	80
8.60 huron::StateProvider Class Reference	80
8.61 huron::odrive::TorqueMotor Class Reference	81
8.61.1 Member Function Documentation	82
8.61.1.1 Initialize()	82
8.61.1.2 Move() [1/3]	82

8.61.1.3 Move() [2/3]	82
8.61.1.4 Move() [3/3]	82
8.61.1.5 SetUp()	82
8.61.1.6 Stop()	82
8.61.1.7 Terminate()	83
8.62 huron::TorqueMotor Class Reference	83
8.63 huron::TorqueMotorConfiguration Class Reference	83
8.63.1 Constructor & Destructor Documentation	84
8.63.1.1 TorqueMotorConfiguration()	84
8.64 huron::VelocityMotor Class Reference	84
8.65 huron::VelocityMotorConfiguration Class Reference	84
8.65.1 Constructor & Destructor Documentation	85
8.65.1.1 VelocityMotorConfiguration()	85
8.66 huron::ZeroMomentPoint Class Reference	85
8.66.1 Member Function Documentation	85
8.66.1.1 Eval()	85
8.66.1.2 ZmpToWorld()	86
8.67 huron::ZeroMomentPointFSRArray Class Reference	86
8.67.1 Member Function Documentation	86
8.67.1.1 Eval()	86
8.68 huron::ZeroMomentPointFTSensor Class Reference	87
8.68.1 Member Function Documentation	87
8.68.1.1 Eval()	87
8.69 huron::ZeroMomentPointTotal Class Reference	87
8.69.1 Member Function Documentation	88
8.69.1.1 Eval()	88
<b>9 File Documentation</b>	<b>89</b>
9.1 enable_protected_make_shared.h	89
9.2 enable_protected_make_unique.h	89
9.3 types.h	89
9.4 push_recovery.h	90
9.5 can_helpers.h	91
9.6 canbus.h	92
9.7 socket_can_bus.h	92
9.8 config.h	93
9.9 odrive_config.h	93
9.10 serial.h	93
9.11 wjwwood_serial.h	95
9.12 rotation.h	96
9.13 com_frame.h	96
9.14 frame.h	96

9.15 joint_common.h . . . . .	97
9.16 logical_frame.h . . . . .	99
9.17 model.h . . . . .	100
9.18 model_composite.h . . . . .	102
9.19 model_impl_factory.h . . . . .	102
9.20 model_impl_interface.h . . . . .	103
9.21 model_impl_types.h . . . . .	104
9.22 pinocchio_model_impl.h . . . . .	104
9.23 force_torque_sensor.h . . . . .	105
9.24 force_torque_sensor.h . . . . .	106
9.25 huron.h . . . . .	106
9.26 huron_node.h . . . . .	107
9.27 joint_group_controller.h . . . . .	108
9.28 joint_state_provider.h . . . . .	108
9.29 actuator.h . . . . .	109
9.30 configuration.h . . . . .	109
9.31 constant_state_provider.h . . . . .	110
9.32 encoder.h . . . . .	110
9.33 generic_component.h . . . . .	111
9.34 joint.h . . . . .	112
9.35 legged_robot.h . . . . .	113
9.36 limb.h . . . . .	113
9.37 motor.h . . . . .	113
9.38 moving_group.h . . . . .	114
9.39 moving_interface.h . . . . .	115
9.40 position_motor.h . . . . .	115
9.41 robot.h . . . . .	115
9.42 rotary_encoder.h . . . . .	116
9.43 sensor.h . . . . .	118
9.44 sensor_with_frame.h . . . . .	118
9.45 state_provider.h . . . . .	118
9.46 torque_motor.h . . . . .	119
9.47 velocity_motor.h . . . . .	119
9.48 invalid_configuration_exception.h . . . . .	120
9.49 not_implemented_exception.h . . . . .	120
9.50 zero_moment_point.h . . . . .	121
9.51 zero_moment_point_fsr_array.h . . . . .	121
9.52 zero_moment_point_ft_sensor.h . . . . .	122
9.53 zero_moment_point_total.h . . . . .	122
9.54 odrive.h . . . . .	122
9.55 odrive_can.h . . . . .	124
9.56 odrive_enums.h . . . . .	125

---

9.57 odrive_rotary_encoder.h . . . . .	127
9.58 odrive_torque_motor.h . . . . .	128
9.59 force_sensing_resistor.h . . . . .	128
9.60 force_sensing_resistor_array.h . . . . .	129
9.61 force_sensing_resistor_array_serial.h . . . . .	129
9.62 string.h . . . . .	130
9.63 time.h . . . . .	130
<b>Index</b>	<b>131</b>



# Chapter 1

## Mumei

There are two ways to run `huron`: using the prebuilt Docker image or building from source.

---

### 1.1 1. Running with Docker:

#### 1.1.1 Prerequisites:

- Docker Engine/Desktop
- Computer with amd64 or arm64 chip

#### 1.1.2 Steps:

1. Pull the image:

```
sudo docker pull wpihuron/huron:<tag>
```

Currently, `<tag>` can only be a pull request (e.g. `pr-72`).

2. Run the container in interactive mode:

```
sudo docker run -it --network=host wpihuron/huron:<tag>
```

The option `--network=host` is needed to expose the network interfaces (including CAN) to the container.

3. To build and run an example code:

First, `cd` into a sepecific example folder in `examples`, e.g. `examples/test_robot_api`. Each example code is a normal CMake project. To build the code:

```
mkdir build && cd build
cmake ..
make
```

If everything is correct, the binary will be built in `build` folder, which is ready to be executed.

---

### 1.2 2. Building from source:

Clone the main repo and all submodules:

```
git clone git@github.com:wpi-huron/huron.git --recurse-submodules
```

#### 1.2.1 Prerequisites:

1. ARM toolchains:

```
sudo apt update
sudo apt install gcc-aarch64-linux-gnu g++-aarch64-linux-gnu
```

2. Build and install third-party CAN library

```
cd third_party/libsockcanpp
mkdir build
cd build
cmake .. -DCMAKE_TOOLCHAIN_FILE=../../tools/<x86_64 or arm64>-toolchain.cmake
        -DBUILD_SHARED_LIBS=ON
make
sudo make install
```

### 3. Build and install third-party Serial library

```
cd third_party/serial
mkdir build
cd build
cmake .. -DCMAKE_TOOLCHAIN_FILE=../../tools/<x86_64 or arm64>-toolchain.cmake
        -DBUILD_SHARED_LIBS=ON
make
sudo make install
```

## 1.2.2 Build and install:

### 1. Make sure you are in the root of this repo (huron/)

### 2. Create build folder

```
mkdir build
```

### 3. Build the project

```
cd build
cmake .. [-DBUILD_TYPE=<build-type>] [-DUSE_PINOCCHIO=1]
make
```

### 4. Install huron

```
sudo make install
```

#### Notes:

- Currently, the project can be built on Linux only
- By default, the project builds for Raspberry Pi 64-bit (arm64). To change platform, BUILD\_TYPE needs to be changed. For example, on Linux x86\_64: -DBUILD\_TYPE=x86\_64

---

## 1.2.3 Uninstall:

```
cd build
sudo make uninstall
```



## Chapter 2

# HURON ROS2 Module

This module contains ROS2 packages that support using ROS2 for simulation or hardware control. It includes the HURON description files (URDF/xacro), Gazebo 11 support, and example code. The ROS packages can be found in `src/`.

We are using [ROS2 Humble Hawksbill](#).

List of packages:

- `huron_description`: includes meshes, a macro xacro file to generate a HURON robot, a top-level xacro file to be visualized in RViz.
  - `huron_gazebo`: Gazebo 11-specific configuration, including sensor/control plugins and a Gazebo launch file.
  - `examples`: Example of using HURON API with ROS2 ([TBA](#))
- 

### 2.0.1 Usage:

Note: the steps below assume you have ROS2 Humble installed.

#### 2.0.1.1 Building the ROS Workspace:

##### 1. Prerequisites:

- You have built and installed the main `huron` project (instructions [here](#)). Note that `huron` needs to be built with x86\_64 toolchain.
- Make sure you are in `huron/ros2/`.

##### 1. Build the workspace:

This command generates 3 new folders in `'huron/ros2/': 'build/', 'install/', and 'log/'`.

Command explanation:

- `'--symlink-install'`: The output files will be symlinks to source files, which means you don't have to re-source the workspace after modifying an `**already built**` file.
- `'--cmake-args -DCMAKE_EXPORT_COMPILE_COMMANDS=ON'`: Generates a `'compile_commands.json'` for auto-completion with `[clangd]` (<https://clangd.llvm.org/>).
- Tip: you can set an alias to call this tedious command faster:  
`"alias cb='colcon build --symlink-install --cmake-args -DCMAKE_EXPORT_COMPILE_COMMANDS=ON'`

Then, you can simply do `cb` instead of the full command.

#### 2.0.1.2 Launching Gazebo 11 simulation

##### 1. Source the workspace setup file

##### 2. Launch Gazebo:

```
"ros2 launch huron_gazebo gazebo.launch.py"
```



## Chapter 3

# Namespace Index

### 3.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

<a href="#">huron</a> . . . . .	13
---------------------------------	----



## Chapter 4

# Hierarchical Index

### 4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

huron::driver::can::BusBase . . . . .	16
huron::driver::can::SocketCanBus . . . . .	79
can_Cyclic_t . . . . .	17
can_Message_t . . . . .	18
can_Signal_t . . . . .	18
huron::driver::can::BusBase::CanSubscription . . . . .	18
class Model	
huron::multibody::ModelComposite . . . . .	44
huron::Configuration . . . . .	20
huron::ActuatorConfiguration . . . . .	15
huron::MotorConfiguration . . . . .	47
huron::PositionMotorConfiguration . . . . .	67
huron::TorqueMotorConfiguration . . . . .	83
huron::VelocityMotorConfiguration . . . . .	84
huron::EncoderConfiguration . . . . .	24
huron::RotaryEncoderConfiguration . . . . .	71
huron::RobotConfiguration . . . . .	69
huron::odrive::ODrive::ODriveConfiguration . . . . .	58
huron::enable_protected_make_shared< ClassWithProtectedCtor > . . . . .	21
huron::enable_protected_make_shared< Frame > . . . . .	21
huron::multibody::Frame . . . . .	30
huron::multibody::ComFrame . . . . .	18
huron::multibody::LogicalFrame . . . . .	39
huron::enable_protected_make_shared< LogicalFrame > . . . . .	21
huron::multibody::LogicalFrame . . . . .	39
huron::enable_protected_make_unique< ClassWithProtectedCtor > . . . . .	22
std::enable_shared_from_this	
huron::multibody::Model . . . . .	40
huron::GenericComponent . . . . .	31
huron::Actuator . . . . .	15
huron::Motor . . . . .	47
huron::PositionMotor . . . . .	66
huron::TorqueMotor . . . . .	83
huron::odrive::TorqueMotor . . . . .	81
huron::VelocityMotor . . . . .	84
huron::Robot . . . . .	67
huron::LeggedRobot . . . . .	38
huron::ros2::Huron . . . . .	32

huron::Sensor . . . . .	72
huron::Encoder . . . . .	22
huron::RotaryEncoder . . . . .	69
huron::odrive::ODriveEncoder . . . . .	59
huron::SensorWithFrame . . . . .	72
huron::ForceSensingResistor . . . . .	24
huron::ForceSensingResistorArray . . . . .	25
huron::ForceSensingResistorArraySerial . . . . .	26
huron::ForceTorqueSensor . . . . .	27
huron::ros2::ForceTorqueSensor . . . . .	29
huron::odrive::ODrive . . . . .	51
huron::odrive::ODriveCAN . . . . .	53
huron::multibody::internal::PinocchioModelImpl::Impl . . . . .	34
huron::Joint . . . . .	35
huron::multibody::JointDescription . . . . .	35
std::logic_error	
huron::InvalidConfigurationException . . . . .	34
huron::NotImplementedException . . . . .	51
huron::multibody::internal::ModelImplFactory . . . . .	44
huron::multibody::internal::ModelImplInterface . . . . .	44
huron::multibody::internal::PinocchioModelImpl . . . . .	60
MovingGroupComponent	
huron::Limb . . . . .	38
huron::MovingInterface . . . . .	49
huron::Actuator . . . . .	15
huron::MovingGroup . . . . .	48
huron::Robot . . . . .	67
huron::ros2::JointGroupController . . . . .	36
huron::driver::can::MsgIdFilterSpecs . . . . .	50
rclcpp::Node	
huron::ros2::HuronNode . . . . .	33
PushRecoveryControl . . . . .	67
huron::driver::serial::SerialBase . . . . .	78
huron::driver::serial::Serial . . . . .	73
huron::StateProvider . . . . .	80
huron::ConstantStateProvider . . . . .	21
huron::Sensor . . . . .	72
huron::ros2::JointStateProvider . . . . .	37
huron::ZeroMomentPoint . . . . .	85
huron::ZeroMomentPointFSRArray . . . . .	86
huron::ZeroMomentPointFTSensor . . . . .	87
huron::ZeroMomentPointTotal . . . . .	87

## Chapter 5

# Class Index

### 5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">huron::Actuator</a>	15
<a href="#">huron::ActuatorConfiguration</a>	15
<a href="#">huron::driver::can::BusBase</a>	16
<a href="#">can_Cyclic_t</a>	17
<a href="#">can_Message_t</a>	18
<a href="#">can_Signal_t</a>	18
<a href="#">huron::driver::can::BusBase::CanSubscription</a>	18
<a href="#">huron::multibody::ComFrame</a>	
Robot center of mass frame	18
<a href="#">huron::Configuration</a>	20
<a href="#">huron::ConstantStateProvider</a>	21
<a href="#">huron::enable_protected_make_shared&lt; ClassWithProtectedCtor &gt;</a>	
This class provides a static method to create a shared_ptr to a class with a protected constructor	21
<a href="#">huron::enable_protected_make_unique&lt; ClassWithProtectedCtor &gt;</a>	
This class provides a static method to create a unique_ptr to a class with a protected constructor	22
<a href="#">huron::Encoder</a>	22
<a href="#">huron::EncoderConfiguration</a>	24
<a href="#">huron::ForceSensingResistor</a>	24
<a href="#">huron::ForceSensingResistorArray</a>	25
<a href="#">huron::ForceSensingResistorArraySerial</a>	26
<a href="#">huron::ForceTorqueSensor</a>	27
<a href="#">huron::ros2::ForceTorqueSensor</a>	29
<a href="#">huron::multibody::Frame</a>	30
<a href="#">huron::GenericComponent</a>	31
<a href="#">huron::ros2::Huron</a>	32
<a href="#">huron::ros2::HuronNode</a>	33
<a href="#">huron::multibody::internal::PinocchioModelImpl::Impl</a>	34
<a href="#">huron::InvalidConfigurationException</a>	34
<a href="#">huron::Joint</a>	35
<a href="#">huron::multibody::JointDescription</a>	35
<a href="#">huron::ros2::JointGroupController</a>	36
<a href="#">huron::ros2::JointStateProvider</a>	37
<a href="#">huron::LeggedRobot</a>	38
<a href="#">huron::Limb</a>	38
<a href="#">huron::multibody::LogicalFrame</a>	
A frame that is defined relative to another frame by an affine transformation. This transformation is user-defined using a function that takes the parent frame's transform in world coordinates as an argument and returns the transform from the parent frame to this frame	39
<a href="#">huron::multibody::Model</a>	40
<a href="#">huron::multibody::ModelComposite</a>	44

<a href="#">huron::multibody::internal::ModelImplFactory</a>	44
<a href="#">huron::multibody::internal::ModelImplInterface</a>	44
<a href="#">huron::Motor</a>	47
<a href="#">huron::MotorConfiguration</a>	47
<a href="#">huron::MovingGroup</a>	48
<a href="#">huron::MovingInterface</a>	49
<a href="#">huron::driver::can::MsgIdFilterSpecs</a>	50
<a href="#">huron::NotImplementedException</a>	51
<a href="#">huron::odrive::ODrive</a>	51
<a href="#">huron::odrive::ODriveCAN</a>	53
<a href="#">huron::odrive::ODrive::ODriveConfiguration</a>	58
<a href="#">huron::odrive::ODriveEncoder</a>	59
<a href="#">huron::multibody::internal::PinocchioModelImpl</a>	60
<a href="#">huron::PositionMotor</a>	66
<a href="#">huron::PositionMotorConfiguration</a>	67
<a href="#">PushRecoveryControl</a>	67
<a href="#">huron::Robot</a>	67
<a href="#">huron::RobotConfiguration</a>	69
<a href="#">huron::RotaryEncoder</a>	69
<a href="#">huron::RotaryEncoderConfiguration</a>	71
<a href="#">huron::Sensor</a>	72
<a href="#">huron::SensorWithFrame</a>	72
<a href="#">huron::driver::serial::Serial</a>	73
<a href="#">huron::driver::serial::SerialBase</a>	78
<a href="#">huron::driver::can::SocketCanBus</a>	79
<a href="#">huron::StateProvider</a>	80
<a href="#">huron::odrive::TorqueMotor</a>	81
<a href="#">huron::TorqueMotor</a>	83
<a href="#">huron::TorqueMotorConfiguration</a>	83
<a href="#">huron::VelocityMotor</a>	84
<a href="#">huron::VelocityMotorConfiguration</a>	84
<a href="#">huron::ZeroMomentPoint</a>	85
<a href="#">huron::ZeroMomentPointFSRArray</a>	86
<a href="#">huron::ZeroMomentPointFTSensor</a>	87
<a href="#">huron::ZeroMomentPointTotal</a>	87



## Chapter 6

# File Index

### 6.1 File List

Here is a list of all documented files with brief descriptions:

/github/workspace/common/include/huron/enable_protected_make_shared.h	89
/github/workspace/common/include/huron/enable_protected_make_unique.h	89
/github/workspace/common/include/huron/types.h	89
/github/workspace/control/include/huron/control/push_recovery.h	90
/github/workspace/driver/can/include/huron/driver/can/can_helpers.h	91
/github/workspace/driver/can/include/huron/driver/can/canbus.h	92
/github/workspace/driver/can/include/huron/driver/can/socket_can_bus.h	92
/github/workspace/driver/config/include/huron/driver/config/config.h	93
/github/workspace/driver/config/include/huron/driver/config/odrive_config.h	93
/github/workspace/driver/serial/include/huron/driver/serial/serial.h	93
/github/workspace/driver/serial/include/huron/driver/serial/wjwwood_serial.h	95
/github/workspace/math/include/huron/math/rotation.h	96
/github/workspace/multibody/include/huron/multibody/com_frame.h	96
/github/workspace/multibody/include/huron/multibody/frame.h	96
/github/workspace/multibody/include/huron/multibody/joint_common.h	97
/github/workspace/multibody/include/huron/multibody/logical_frame.h	99
/github/workspace/multibody/include/huron/multibody/model.h	100
/github/workspace/multibody/include/huron/multibody/model_composite.h	102
/github/workspace/multibody/include/huron/multibody/model_impl_factory.h	102
/github/workspace/multibody/include/huron/multibody/model_impl_interface.h	103
/github/workspace/multibody/include/huron/multibody/model_impl_types.h	104
/github/workspace/multibody/include/huron/multibody/pinocchio_model_impl.h	104
/github/workspace/ros2/src/huron_ros2/include/huron_ros2/force_torque_sensor.h	105
/github/workspace/ros2/src/huron_ros2/include/huron_ros2/huron.h	106
/github/workspace/ros2/src/huron_ros2/include/huron_ros2/huron_node.h	107
/github/workspace/ros2/src/huron_ros2/include/huron_ros2/joint_group_controller.h	108
/github/workspace/ros2/src/huron_ros2/include/huron_ros2/joint_state_provider.h	108
/github/workspace/system/control_interfaces/include/huron/control_interfaces/actuator.h	109
/github/workspace/system/control_interfaces/include/huron/control_interfaces/configuration.h	109
/github/workspace/system/control_interfaces/include/huron/control_interfaces/constant_state_provider.h	110
/github/workspace/system/control_interfaces/include/huron/control_interfaces/encoder.h	110
/github/workspace/system/control_interfaces/include/huron/control_interfaces/generic_component.h	111
/github/workspace/system/control_interfaces/include/huron/control_interfaces/joint.h	112
/github/workspace/system/control_interfaces/include/huron/control_interfaces/legged_robot.h	113
/github/workspace/system/control_interfaces/include/huron/control_interfaces/limb.h	113
/github/workspace/system/control_interfaces/include/huron/control_interfaces/motor.h	113
/github/workspace/system/control_interfaces/include/huron/control_interfaces/moving_group.h	114
/github/workspace/system/control_interfaces/include/huron/control_interfaces/moving_interface.h	115
/github/workspace/system/control_interfaces/include/huron/control_interfaces/position_motor.h	115
/github/workspace/system/control_interfaces/include/huron/control_interfaces/robot.h	115

/github/workspace/system/control_interfaces/include/huron/control_interfaces/rotary_encoder.h . . . . .	116
/github/workspace/system/control_interfaces/include/huron/control_interfaces/sensor.h . . . . .	118
/github/workspace/system/control_interfaces/include/huron/control_interfaces/sensor_with_frame.h . . . . .	118
/github/workspace/system/control_interfaces/include/huron/control_interfaces/state_provider.h . . . . .	118
/github/workspace/system/control_interfaces/include/huron/control_interfaces/torque_motor.h . . . . .	119
/github/workspace/system/control_interfaces/include/huron/control_interfaces/velocity_motor.h . . . . .	119
/github/workspace/system/exceptions/include/huron/exceptions/invalid_configuration_exception.h . . . . .	120
/github/workspace/system/exceptions/include/huron/exceptions/not_implemented_exception.h . . . . .	120
/github/workspace/system/locomotion/include/huron/locomotion/zero_moment_point.h . . . . .	121
/github/workspace/system/locomotion/include/huron/locomotion/zero_moment_point_fsr_array.h . . . . .	121
/github/workspace/system/locomotion/include/huron/locomotion/zero_moment_point_ft_sensor.h . . . . .	122
/github/workspace/system/locomotion/include/huron/locomotion/zero_moment_point_total.h . . . . .	122
/github/workspace/system/odrive/include/huron/odrive/odrive.h . . . . .	122
/github/workspace/system/odrive/include/huron/odrive/odrive_can.h . . . . .	124
/github/workspace/system/odrive/include/huron/odrive/odrive_enums.h . . . . .	125
/github/workspace/system/odrive/include/huron/odrive/odrive_rotary_encoder.h . . . . .	127
/github/workspace/system/odrive/include/huron/odrive/odrive_torque_motor.h . . . . .	128
/github/workspace/system/sensors/include/huron/sensors/force_sensing_resistor.h . . . . .	128
/github/workspace/system/sensors/include/huron/sensors/force_sensing_resistor_array.h . . . . .	129
/github/workspace/system/sensors/include/huron/sensors/force_sensing_resistor_array_serial.h . . . . .	129
/github/workspace/system/sensors/include/huron/sensors/force_torque_sensor.h . . . . .	106
/github/workspace/utils/include/huron/utils/string.h . . . . .	130
/github/workspace/utils/include/huron/utils/time.h . . . . .	130

## Chapter 7

# Namespace Documentation

### 7.1 huron Namespace Reference

#### Classes

- class [Actuator](#)
- class [ActuatorConfiguration](#)
- class [Configuration](#)
- class [ConstantStateProvider](#)
- class [enable\\_protected\\_make\\_shared](#)

*This class provides a static method to create a `shared_ptr` to a class with a protected constructor.*

- class [enable\\_protected\\_make\\_unique](#)

*This class provides a static method to create a `unique_ptr` to a class with a protected constructor.*

- class [Encoder](#)
- class [EncoderConfiguration](#)
- class [ForceSensingResistor](#)
- class [ForceSensingResistorArray](#)
- class [ForceSensingResistorArraySerial](#)
- class [ForceTorqueSensor](#)
- class [GenericComponent](#)
- class [InvalidConfigurationException](#)
- class [Joint](#)
- class [LeggedRobot](#)
- class [Limb](#)
- class [Motor](#)
- class [MotorConfiguration](#)
- class [MovingGroup](#)
- class [MovingInterface](#)
- class [NotImplementedException](#)
- class [PositionMotor](#)
- class [PositionMotorConfiguration](#)
- class [Robot](#)
- class [RobotConfiguration](#)
- class [RotaryEncoder](#)
- class [RotaryEncoderConfiguration](#)
- class [Sensor](#)
- class [SensorWithFrame](#)
- class [StateProvider](#)
- class [TorqueMotor](#)
- class [TorqueMotorConfiguration](#)
- class [VelocityMotor](#)
- class [VelocityMotorConfiguration](#)

- class [ZeroMomentPoint](#)
- class [ZeroMomentPointFSRArray](#)
- class [ZeroMomentPointFTSensor](#)
- class [ZeroMomentPointTotal](#)

## Typedefs

- typedef Eigen::Matrix< double, 6, 1 > **Vector6d**
- typedef Eigen::Matrix< double, 6, 6 > **Matrix6d**
- typedef Eigen::Matrix< double, 6, Eigen::Dynamic > **Matrix6Xd**
- typedef std::unordered\_map< std::string, std::any > **ConfigMap**

### 7.1.1 Detailed Description

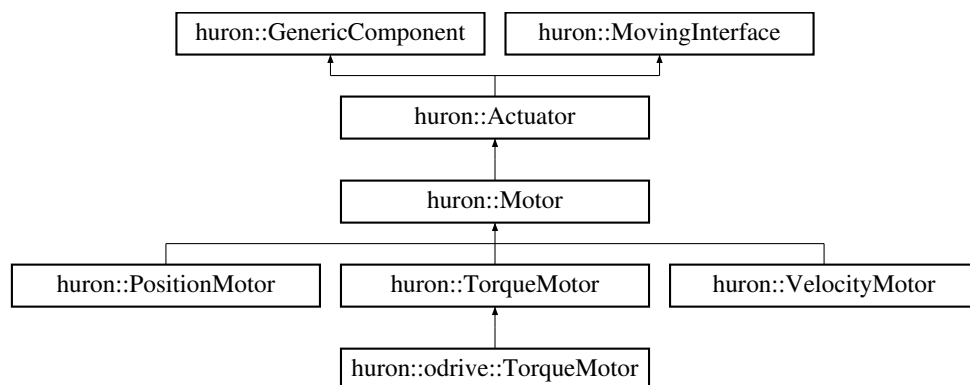
Source: <https://stackoverflow.com/questions/24469927/does-c-have-an-equivalent-to-nets-n>  
See <https://docs.odriverobotics.com/v/0.5.6/can-protocol.html> for more information  
about this CAN API.

## Chapter 8

# Class Documentation

### 8.1 `huron::Actuator` Class Reference

Inheritance diagram for `huron::Actuator`:



#### Public Member Functions

- **Actuator** (size\_t dim, std::unique\_ptr< [ActuatorConfiguration](#) > config)
- **Actuator** (size\_t dim)
- **Actuator** (const [Actuator](#) &)=delete
- [Actuator](#) & **operator=** (const [Actuator](#) &)=delete

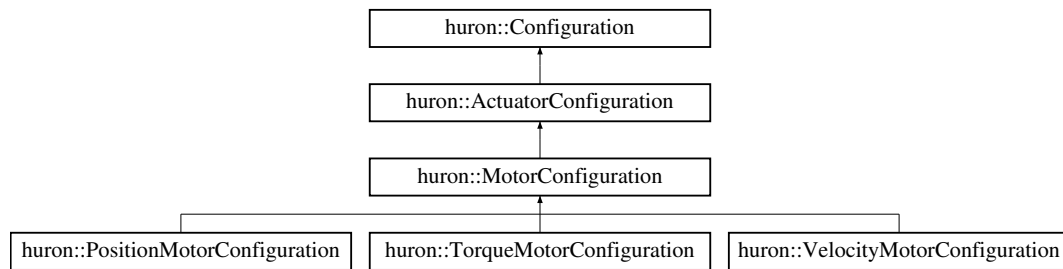
#### Additional Inherited Members

The documentation for this class was generated from the following file:

- `/github/workspace/system/control_interfaces/include/huron/control_interfaces/actuator.h`

### 8.2 `huron::ActuatorConfiguration` Class Reference

Inheritance diagram for `huron::ActuatorConfiguration`:



## Public Member Functions

- **ActuatorConfiguration** (ConfigMap config\_map, std::set< std::string > valid\_keys)
- **ActuatorConfiguration** (ConfigMap config\_map)

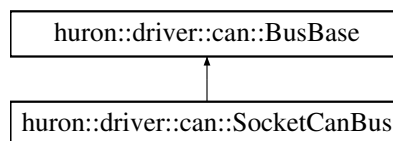
## Additional Inherited Members

The documentation for this class was generated from the following file:

- /github/workspace/system/control\_interfaces/include/huron/control\_interfaces/actuator.h

## 8.3 huron::driver::can::BusBase Class Reference

Inheritance diagram for `huron::driver::can::BusBase`:



## Classes

- struct [CanSubscription](#)

## Public Types

- typedef void(\* **on\_can\_message\_cb\_t**) (void \*ctx, const [can\\_Message\\_t](#) &message)

## Public Member Functions

- **BusBase** (const [BusBase](#) &)=delete
- **BusBase** & **operator=** (const [BusBase](#) &)=delete
- virtual bool **send\_message** (const [can\\_Message\\_t](#) &message)=0  
*Sends the specified CAN message.*
- virtual bool **recv\_message** ([can\\_Message\\_t](#) &message, uint32\_t timeout=UINT32\_MAX)=0  
*Receives a CAN message with the same id as message.*
- virtual bool **subscribe** (const [MsgIdFilterSpecs](#) &filter, on\_can\_message\_cb\_t callback, void \*ctx, [CanSubscription](#) \*\*handle)=0  
*Registers a callback that will be invoked for every incoming CAN message that matches the filter.*
- virtual bool **unsubscribe** ([CanSubscription](#) \*handle)=0  
*Deregisters a callback that was previously registered with [subscribe\(\)](#).*

### 8.3.1 Member Function Documentation

### 8.3.1.1 recv\_message()

```
virtual bool huron::driver::can::BusBase::recv_message (
    can_Message_t & message,
    uint32_t timeout = UINT32_MAX ) [pure virtual]
```

Receives a CAN message with the same id as message.

#### Returns

: true on success or false otherwise (e.g. if the receive queue is empty).

Implemented in [huron::driver::can::SocketCanBus](#).

### 8.3.1.2 send\_message()

```
virtual bool huron::driver::can::BusBase::send_message (
    const can_Message_t & message ) [pure virtual]
```

Sends the specified CAN message.

#### Returns

: true on success or false otherwise (e.g. if the send queue is full).

Implemented in [huron::driver::can::SocketCanBus](#).

### 8.3.1.3 subscribe()

```
virtual bool huron::driver::can::BusBase::subscribe (
    const MsgIdFilterSpecs & filter,
    on_can_message_cb_t callback,
    void * ctx,
    CanSubscription ** handle ) [pure virtual]
```

Registers a callback that will be invoked for every incoming CAN message that matches the filter.

#### Parameters

<i>handle</i>	On success this handle is set to an opaque pointer that can be used to cancel the subscription.
---------------	---

#### Returns

: true on success or false otherwise (e.g. if the maximum number of subscriptions has been reached).

Implemented in [huron::driver::can::SocketCanBus](#).

### 8.3.1.4 unsubscribe()

```
virtual bool huron::driver::can::BusBase::unsubscribe (
    CanSubscription * handle ) [pure virtual]
```

Deregisters a callback that was previously registered with [subscribe\(\)](#).

Implemented in [huron::driver::can::SocketCanBus](#).

The documentation for this class was generated from the following file:

- [/github/workspace/driver/can/include/huron/driver/can/canbus.h](#)

## 8.4 can\_Cyclic\_t Struct Reference

### Public Attributes

- uint32\_t **cycleTime\_ms**

- uint32\_t **lastTime\_ms**

The documentation for this struct was generated from the following file:

- /github/workspace/driver/can/include/huron/driver/can/can\_helpers.h

## 8.5 can\_Message\_t Struct Reference

### Public Attributes

- uint32\_t **id** = 0x000
- bool **isExt** = false
- bool **rtr** = false
- uint8\_t **len** = 8
- uint8\_t **buf** [8] = {0, 0, 0, 0, 0, 0, 0, 0}

The documentation for this struct was generated from the following file:

- /github/workspace/driver/can/include/huron/driver/can/can\_helpers.h

## 8.6 can\_Signal\_t Struct Reference

### Public Attributes

- const uint8\_t **startBit**
- const uint8\_t **length**
- const bool **isIntel**
- const float **factor**
- const float **offset**

The documentation for this struct was generated from the following file:

- /github/workspace/driver/can/include/huron/driver/can/can\_helpers.h

## 8.7 huron::driver::can::BusBase::CanSubscription Struct Reference

The documentation for this struct was generated from the following file:

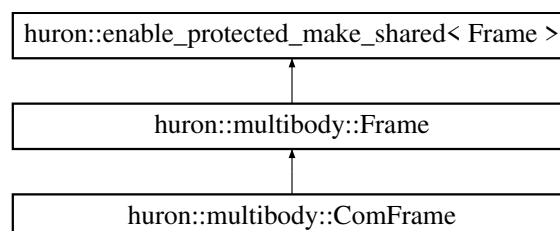
- /github/workspace/driver/can/include/huron/driver/can/canbus.h

## 8.8 huron::multibody::ComFrame Class Reference

[Robot](#) center of mass frame.

```
#include <com_frame.h>
```

Inheritance diagram for huron::multibody::ComFrame:





## Public Member Functions

- **ComFrame** (FrameIndex index, const std::string &name, bool is\_user\_defined, std::weak\_ptr< const [Model](#) > model, FrameIndex parent\_frame\_index)
- **ComFrame** (const [ComFrame](#) &)=delete
- [ComFrame](#) & **operator=** (const [ComFrame](#) &)=delete
- Eigen::Affine3d [GetTransformInWorld](#) () const override
- Eigen::Affine3d [GetTransformFromFrame](#) (const [Frame](#) &other) const override
- Eigen::Affine3d [GetTransformFromFrame](#) (FrameIndex other) const override
- Eigen::Affine3d [GetTransformToFrame](#) (const [Frame](#) &other) const override
- Eigen::Affine3d [GetTransformToFrame](#) (FrameIndex other) const override

## Additional Inherited Members

### 8.8.1 Detailed Description

[Robot](#) center of mass frame.

### 8.8.2 Member Function Documentation

#### 8.8.2.1 GetTransformFromFrame() [1/2]

```
Eigen::Affine3d huron::multibody::ComFrame::GetTransformFromFrame (
    const Frame & other ) const [override], [virtual]
```

Reimplemented from [huron::multibody::Frame](#).

#### 8.8.2.2 GetTransformFromFrame() [2/2]

```
Eigen::Affine3d huron::multibody::ComFrame::GetTransformFromFrame (
    FrameIndex other ) const [override], [virtual]
```

Reimplemented from [huron::multibody::Frame](#).

#### 8.8.2.3 GetTransformInWorld()

```
Eigen::Affine3d huron::multibody::ComFrame::GetTransformInWorld ( ) const [override], [virtual]
```

Reimplemented from [huron::multibody::Frame](#).

#### 8.8.2.4 GetTransformToFrame() [1/2]

```
Eigen::Affine3d huron::multibody::ComFrame::GetTransformToFrame (
    const Frame & other ) const [override], [virtual]
```

Reimplemented from [huron::multibody::Frame](#).

#### 8.8.2.5 GetTransformToFrame() [2/2]

```
Eigen::Affine3d huron::multibody::ComFrame::GetTransformToFrame (
    FrameIndex other ) const [override], [virtual]
```

Reimplemented from [huron::multibody::Frame](#).

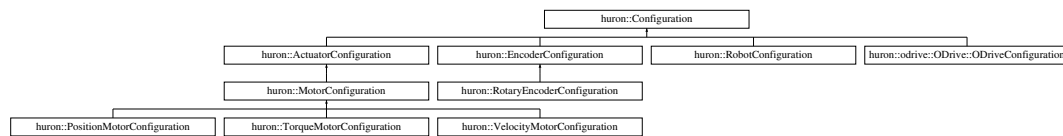
The documentation for this class was generated from the following files:

- /github/workspace/multibody/include/huron/multibody/com\_frame.h
- /github/workspace/multibody/src/com\_frame.cc

## 8.9 huron::Configuration Class Reference

```
#include <configuration.h>
```

Inheritance diagram for huron::Configuration:



### Public Member Functions

- **Configuration** (ConfigMap config\_map, std::set< std::string > valid\_keys)
- **Configuration** (ConfigMap config\_map)
- **Configuration** (const [Configuration](#) &)=delete
- [Configuration](#) & **operator=** (const [Configuration](#) &)=delete
- std::any **Get** (std::string config\_key, bool renew=false)
- bool **Set** (std::string config\_key, std::any config\_value)
- bool **Set** (ConfigMap config\_map)

### Protected Member Functions

- bool [ValidateKey](#) (std::string config\_key)
- ConfigMap **ValidateMap** (ConfigMap config\_map)
- virtual std::any [GetFromComponent](#) (std::string config\_key)

### Protected Attributes

- const std::set< std::string > **valid\_keys\_**
- ConfigMap **config\_map\_**

### 8.9.1 Detailed Description

Abstract data structure for component configuration.

### 8.9.2 Member Function Documentation

#### 8.9.2.1 Get()

```
std::any huron::Configuration::Get (
    std::string config_key,
    bool renew = false )
```

Gets the value of the configuration with key config\_key.

If the configuration is not cached, gets the value from the component (e.g. from the hardware), caches the value, then returns it. To force getting a new value, set renew to true.

#### Exceptions

<a href="#">InvalidConfigurationException</a>	if config_key is invalid.
---	---------------------------

#### 8.9.2.2 GetFromComponent()

```
virtual std::any huron::Configuration::GetFromComponent (
```

```
std::string config_key ) [inline], [protected], [virtual]
```

Gets the configuration value from the hardware component. This method needs to be overridden by concrete configuration classes.

### 8.9.2.3 ValidateKey()

```
bool huron::Configuration::ValidateKey (
    std::string config_key ) [inline], [protected]
```

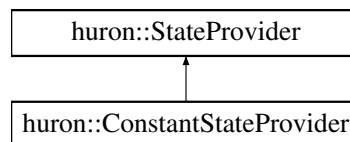
Checks if the key is valid (i.e. in a list of valid keys).

The documentation for this class was generated from the following files:

- /github/workspace/system/control\_interfaces/include/huron/control\_interfaces/configuration.h
- /github/workspace/system/control\_interfaces/src/configuration.cc

## 8.10 huron::ConstantStateProvider Class Reference

Inheritance diagram for huron::ConstantStateProvider:



### Public Member Functions

- **ConstantStateProvider** (const Eigen::MatrixXd &state)
- **ConstantStateProvider** (const [ConstantStateProvider](#) &)=delete
- **ConstantStateProvider & operator=** (const [ConstantStateProvider](#) &)=delete
- void [RequestStateUpdate](#) () override
- void [GetNewState](#) (Eigen::Ref< Eigen::MatrixXd > new\_state) const override
- void **SetState** (const Eigen::MatrixXd &state)

### 8.10.1 Member Function Documentation

#### 8.10.1.1 GetNewState()

```
void huron::ConstantStateProvider::GetNewState (
    Eigen::Ref< Eigen::MatrixXd > new_state ) const [inline], [override], [virtual]
```

Implements [huron::StateProvider](#).

#### 8.10.1.2 RequestStateUpdate()

```
void huron::ConstantStateProvider::RequestStateUpdate ( ) [inline], [override], [virtual]
```

Implements [huron::StateProvider](#).

The documentation for this class was generated from the following file:

- /github/workspace/system/control\_interfaces/include/huron/control\_interfaces/constant\_state\_provider.h

## 8.11 huron::enable\_protected\_make\_shared< ClassWithProtectedCtor > Class Template Reference

This class provides a static method to create a shared\_ptr to a class with a protected constructor.

```
#include <enable_protected_make_shared.h>
```

## Static Protected Member Functions

- `template<typename... Args>`  
`static std::shared_ptr< ClassWithProtectedCtor > make_shared (Args &&... args)`

### 8.11.1 Detailed Description

```
template<typename ClassWithProtectedCtor>
class huron::enable_protected_make_shared< ClassWithProtectedCtor >
```

This class provides a static method to create a `shared_ptr` to a class with a protected constructor.

This is useful for classes that should not be instantiated directly, but should instead be created by a factory method.

Example: `class Foo : public enable_protected_make_shared<Foo> { public: friend class Bar; protected: Foo(int a, int b) : a_(a), b_(b) {} int a_; int b_; }; class Bar { std::shared_ptr<Foo> CreateFoo(int a, double b) { return Foo::make_shared(a, b); } }`

<https://stackoverflow.com/a/73236821>

The documentation for this class was generated from the following file:

- `/github/workspace/common/include/huron/enable_protected_make_shared.h`

## 8.12 huron::enable\_protected\_make\_unique< ClassWithProtectedCtor > Class Template Reference

This class provides a static method to create a `unique_ptr` to a class with a protected constructor.

```
#include <enable_protected_make_unique.h>
```

## Static Protected Member Functions

- `template<typename... Args>`  
`static std::unique_ptr< ClassWithProtectedCtor > make_unique (Args &&... args)`

### 8.12.1 Detailed Description

```
template<typename ClassWithProtectedCtor>
class huron::enable_protected_make_unique< ClassWithProtectedCtor >
```

This class provides a static method to create a `unique_ptr` to a class with a protected constructor.

This is useful for classes that should not be instantiated directly, but should instead be created by a factory method.

Example: `class Foo : public enable_protected_make_unique<Foo> { public: friend class Bar; protected: Foo(int a, int b) : a_(a), b_(b) {} int a_; int b_; }; class Bar { std::unique_ptr<Foo> CreateFoo(int a, double b) { return Foo::make_unique(a, b); } }`

<https://stackoverflow.com/a/73236821>

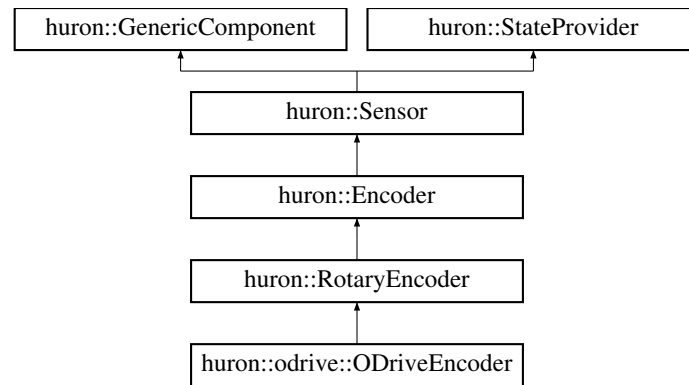
The documentation for this class was generated from the following file:

- `/github/workspace/common/include/huron/enable_protected_make_unique.h`

## 8.13 huron::Encoder Class Reference

```
#include <encoder.h>
```

Inheritance diagram for `huron::Encoder`:



## Public Member Functions

- **Encoder** (double gear\_ratio, std::unique\_ptr< [EncoderConfiguration](#) > config)
- **Encoder** (double gear\_ratio)
- **Encoder** (std::unique\_ptr< [EncoderConfiguration](#) > config)
- **Encoder** (const [Encoder](#) &)=delete
- [Encoder](#) & **operator=** (const [Encoder](#) &)=delete
- void [GetNewState](#) (Eigen::Ref< Eigen::MatrixXd > new\_state) const override
- virtual double [GetPosition](#) () const =0
- virtual double [GetVelocity](#) () const =0
- virtual void [Reset](#) ()=0

## Protected Attributes

- double **gear\_ratio\_**

## Additional Inherited Members

### 8.13.1 Detailed Description

Abstract class for encoder A generic encoder has count and velocity.

### 8.13.2 Member Function Documentation

#### 8.13.2.1 GetNewState()

```
void huron::Encoder::GetNewState (
    Eigen::Ref< Eigen::MatrixXd > new_state ) const [inline], [override], [virtual]
```

Implements [huron::stateProvider](#).

#### 8.13.2.2 GetPosition()

```
virtual double huron::Encoder::GetPosition ( ) const [pure virtual]
```

Implemented in [huron::RotaryEncoder](#).

#### 8.13.2.3 GetVelocity()

```
virtual double huron::Encoder::GetVelocity ( ) const [pure virtual]
```

Implemented in [huron::RotaryEncoder](#).

### 8.13.2.4 Reset()

virtual void huron::Encoder::Reset ( ) [pure virtual]

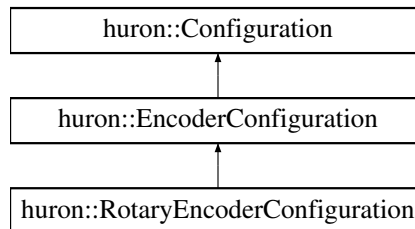
Implemented in [huron::RotaryEncoder](#).

The documentation for this class was generated from the following file:

- [/github/workspace/system/control\\_interfaces/include/huron/control\\_interfaces/encoder.h](#)

## 8.14 huron::EncoderConfiguration Class Reference

Inheritance diagram for huron::EncoderConfiguration:



### Public Member Functions

- **EncoderConfiguration** (ConfigMap config\_map, std::set< std::string > valid\_keys)
- **EncoderConfiguration** (ConfigMap config\_map)

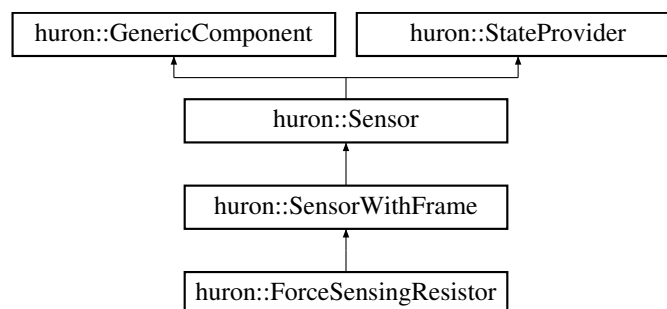
### Additional Inherited Members

The documentation for this class was generated from the following file:

- [/github/workspace/system/control\\_interfaces/include/huron/control\\_interfaces/encoder.h](#)

## 8.15 huron::ForceSensingResistor Class Reference

Inheritance diagram for huron::ForceSensingResistor:



### Public Member Functions

- **ForceSensingResistor** (std::weak\_ptr< const [multibody::Frame](#) > frame)
- **ForceSensingResistor** (std::weak\_ptr< const [multibody::Frame](#) > frame, std::unique\_ptr< [Configuration](#) > config)
- **ForceSensingResistor** (const [ForceSensingResistor](#) &)=delete
- [ForceSensingResistor](#) & **operator=** (const [ForceSensingResistor](#) &)=delete

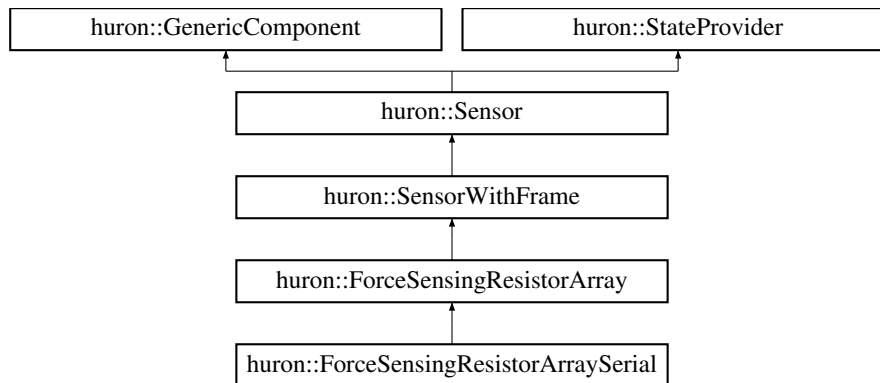
## Additional Inherited Members

The documentation for this class was generated from the following files:

- /github/workspace/system/sensors/include/huron/sensors/force\_sensing\_resistor.h
- /github/workspace/system/sensors/src/force\_sensing\_resistor.cc

## 8.16 huron::ForceSensingResistorArray Class Reference

Inheritance diagram for huron::ForceSensingResistorArray:



## Public Member Functions

- **ForceSensingResistorArray** (const std::string &name, std::weak\_ptr< const [multibody::Frame](#) > frame, const std::vector< std::shared\_ptr< [ForceSensingResistor](#) > > &fsr\_array)
- **ForceSensingResistorArray** (const std::string &name, std::weak\_ptr< const [multibody::Frame](#) > frame, const std::vector< std::shared\_ptr< [ForceSensingResistor](#) > > &fsr\_array, std::unique\_ptr< [Configuration](#) > config)
- **ForceSensingResistorArray** (const [ForceSensingResistorArray](#) &)=delete
- [ForceSensingResistorArray](#) & **operator=** (const [ForceSensingResistorArray](#) &)=delete
- void [RequestStateUpdate](#) () override
- void [GetNewState](#) (Eigen::Ref< Eigen::MatrixXd > new\_state) const override
- Eigen::Affine3d [GetSensorPose](#) (size\_t index) const
- size\_t [num\\_sensors](#) () const

## Protected Attributes

- std::string [name\\_](#)
- Eigen::VectorXd [values\\_](#)
- std::vector< std::shared\_ptr< [ForceSensingResistor](#) > > [fsr\\_array\\_](#)

## Additional Inherited Members

### 8.16.1 Member Function Documentation

#### 8.16.1.1 GetNewState()

```
void huron::ForceSensingResistorArray::GetNewState (
    Eigen::Ref< Eigen::MatrixXd > new_state ) const [override], [virtual]
Implements huron::StateProvider.
```

### 8.16.1.2 RequestStateUpdate()

```
void huron::ForceSensingResistorArray::RequestStateUpdate ( ) [override], [virtual]
```

Implements [huron::StateProvider](#).

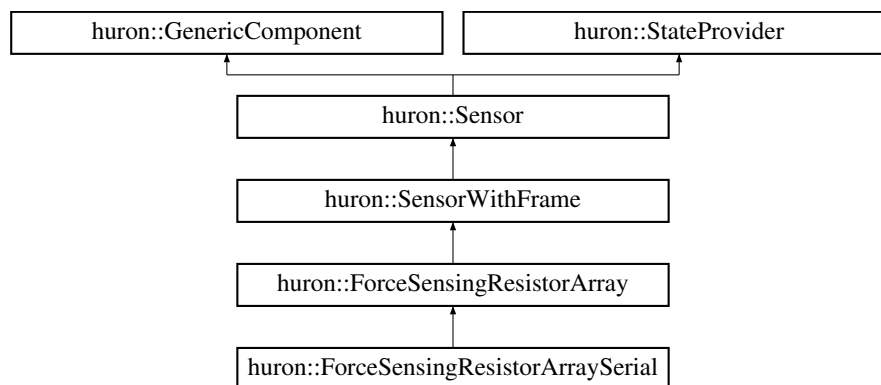
The documentation for this class was generated from the following files:

- /github/workspace/system/sensors/include/huron/sensors/force\_sensing\_resistor\_array.h
- /github/workspace/system/sensors/src/force\_sensing\_resistor\_array.cc

## 8.17 huron::ForceSensingResistorArraySerial Class Reference

```
#include <force_sensing_resistor_array_serial.h>
```

Inheritance diagram for huron::ForceSensingResistorArraySerial:



### Public Member Functions

- **ForceSensingResistorArraySerial** (const std::string &name, std::weak\_ptr< const [multibody::Frame](#) > frame, const std::vector< std::shared\_ptr< [ForceSensingResistor](#) > > &fsr\_array, std::shared\_ptr< [driver::serial::SerialBase](#) > serial)
- **ForceSensingResistorArraySerial** (const std::string &name, std::weak\_ptr< const [multibody::Frame](#) > frame, const std::vector< std::shared\_ptr< [ForceSensingResistor](#) > > &fsr\_array, std::shared\_ptr< [driver::serial::SerialBase](#) > serial, std::unique\_ptr< [Configuration](#) > config)
- **ForceSensingResistorArraySerial** (const [ForceSensingResistorArraySerial](#) &)=delete
- **ForceSensingResistorArraySerial & operator=** (const [ForceSensingResistorArraySerial](#) &)=delete
- void [RequestStateUpdate](#) () override
- Eigen::VectorXd [GetValue](#) () const override
- *Get the sensor value.*
- Eigen::VectorXd [ReloadAndGetValue](#) () override
- void [Initialize](#) () override
- void [SetUp](#) () override
- void [Terminate](#) () override

### Additional Inherited Members

#### 8.17.1 Detailed Description

An array of FSR with values transmitted over Serial communication.

The sensor values are in double but sent in string in the following syntax: <sensor\_name>,<val\_1>,<val\_2>,...,<val\_n>

The sensor values should be sent periodically.

#### 8.17.2 Member Function Documentation



### 8.17.2.1 GetValue()

```
Eigen::VectorXd huron::ForceSensingResistorArraySerial::GetValue ( ) const [override], [virtual]
```

Get the sensor value.

Reimplemented from [huron::Sensor](#).

### 8.17.2.2 Initialize()

```
void huron::ForceSensingResistorArraySerial::Initialize ( ) [override], [virtual]
```

Implements [huron::GenericComponent](#).

### 8.17.2.3 ReloadAndGetValue()

```
Eigen::VectorXd huron::ForceSensingResistorArraySerial::ReloadAndGetValue ( ) [override],  
[virtual]
```

Reimplemented from [huron::Sensor](#).

### 8.17.2.4 RequestStateUpdate()

```
void huron::ForceSensingResistorArraySerial::RequestStateUpdate ( ) [override], [virtual]
```

Reimplemented from [huron::ForceSensingResistorArray](#).

### 8.17.2.5 SetUp()

```
void huron::ForceSensingResistorArraySerial::SetUp ( ) [override], [virtual]
```

Implements [huron::GenericComponent](#).

### 8.17.2.6 Terminate()

```
void huron::ForceSensingResistorArraySerial::Terminate ( ) [override], [virtual]
```

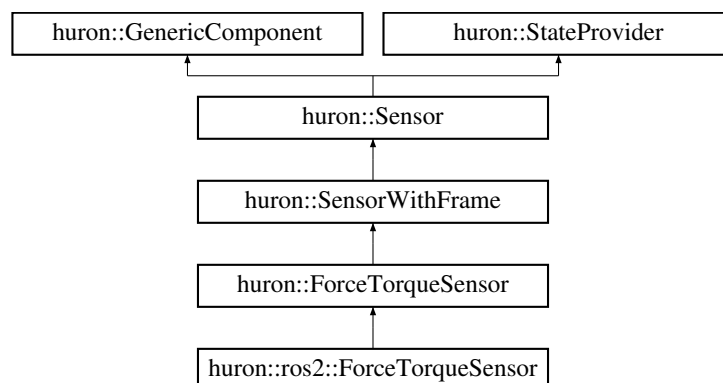
Implements [huron::GenericComponent](#).

The documentation for this class was generated from the following files:

- [/github/workspace/system/sensors/include/huron/sensors/force\\_sensing\\_resistor\\_array\\_serial.h](#)
- [/github/workspace/system/sensors/src/force\\_sensing\\_resistor\\_array\\_serial.cc](#)

## 8.18 huron::ForceTorqueSensor Class Reference

Inheritance diagram for `huron::ForceTorqueSensor`:



## Public Member Functions

- **ForceTorqueSensor** (bool reverse\_wrench\_direction, std::weak\_ptr< const [multibody::Frame](#) > frame)
- **ForceTorqueSensor** (bool reverse\_wrench\_direction, std::weak\_ptr< const [multibody::Frame](#) > frame, std::unique\_ptr< [Configuration](#) > config)
- **ForceTorqueSensor** (const [ForceTorqueSensor](#) &)=delete
- **ForceTorqueSensor & operator=** (const [ForceTorqueSensor](#) &)=delete
- void [RequestStateUpdate](#) () override
- void [GetNewState](#) (Eigen::Ref< Eigen::MatrixXd > new\_state) const override
- Eigen::VectorXd [GetValue](#) () const override

## Protected Member Functions

- virtual Vector6d [DoGetWrenchRaw](#) ()=0

## Protected Attributes

- bool [reverse\\_wrench\\_direction\\_](#)

## 8.18.1 Member Function Documentation

### 8.18.1.1 DoGetWrenchRaw()

```
virtual Vector6d huron::ForceTorqueSensor::DoGetWrenchRaw ( ) [protected], [pure virtual]
```

To be overridden.

Implemented in [huron::ros2::ForceTorqueSensor](#).

### 8.18.1.2 GetNewState()

```
void huron::ForceTorqueSensor::GetNewState (
    Eigen::Ref< Eigen::MatrixXd > new_state ) const [override], [virtual]
```

Implements [huron::StateProvider](#).

### 8.18.1.3 GetValue()

```
Eigen::VectorXd huron::ForceTorqueSensor::GetValue ( ) const [override], [virtual]
```

Measures the external forces and moments.

**Returns**

Wrench 6x1 vector  $[F_x, F_y, F_z, T_x, T_y, T_z]^T$ .

Reimplemented from [huron::Sensor](#).

### 8.18.1.4 RequestStateUpdate()

```
void huron::ForceTorqueSensor::RequestStateUpdate ( ) [override], [virtual]
```

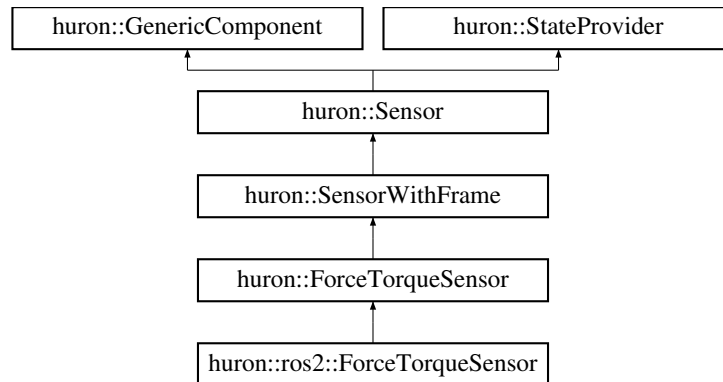
Implements [huron::StateProvider](#).

The documentation for this class was generated from the following files:

- /github/workspace/system/sensors/include/huron/sensors/force\_torque\_sensor.h
- /github/workspace/system/sensors/src/force\_torque\_sensor.cc

## 8.19 huron::ros2::ForceTorqueSensor Class Reference

Inheritance diagram for huron::ros2::ForceTorqueSensor:



### Public Member Functions

- **ForceTorqueSensor** (bool reverse\_wrench\_direction, std::weak\_ptr< const [multibody::Frame](#) > frame)
- **ForceTorqueSensor** (const [ForceTorqueSensor](#) &)=delete
- [ForceTorqueSensor](#) & **operator=** (const [ForceTorqueSensor](#) &)=delete
- void [Initialize](#) () override
- void [SetUp](#) () override
- void [Terminate](#) () override

### Protected Member Functions

- Vector6d [DoGetWrenchRaw](#) () override

### Friends

- class **HuronNode**

### Additional Inherited Members

#### 8.19.1 Member Function Documentation

##### 8.19.1.1 DoGetWrenchRaw()

Vector6d huron::ros2::ForceTorqueSensor::DoGetWrenchRaw ( ) [override], [protected], [virtual]

To be overridden.

Implements [huron::ForceTorqueSensor](#).

##### 8.19.1.2 Initialize()

void huron::ros2::ForceTorqueSensor::Initialize ( ) [override], [virtual]

Implements [huron::GenericComponent](#).

##### 8.19.1.3 SetUp()

void huron::ros2::ForceTorqueSensor::SetUp ( ) [override], [virtual]

Implements [huron::GenericComponent](#).

#### 8.19.1.4 Terminate()

void `huron::ros2::ForceTorqueSensor::Terminate` ( ) [override], [virtual]

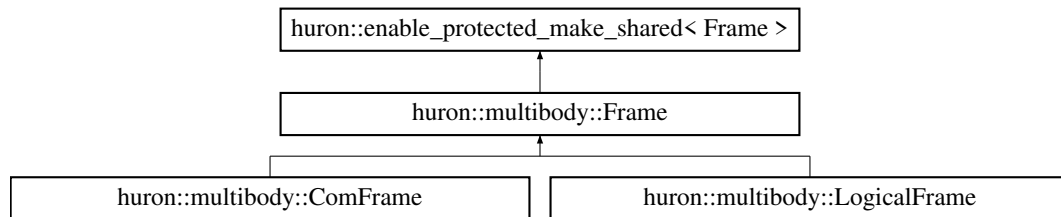
Implements [huron::GenericComponent](#).

The documentation for this class was generated from the following files:

- `/github/workspace/ros2/src/huron_ros2/include/huron_ros2/force_torque_sensor.h`
- `/github/workspace/ros2/src/huron_ros2/src/force_torque_sensor.cc`

## 8.20 `huron::multibody::Frame` Class Reference

Inheritance diagram for `huron::multibody::Frame`:



### Public Member Functions

- **Frame** (const [Frame](#) &)=delete
- **Frame** & **operator=** (const [Frame](#) &)=delete
- virtual Eigen::Affine3d **GetTransformInWorld** () const
- virtual Eigen::Affine3d **GetTransformFromFrame** (const [Frame](#) &other) const
- virtual Eigen::Affine3d **GetTransformFromFrame** (FrameIndex other) const
- virtual Eigen::Affine3d **GetTransformToFrame** (const [Frame](#) &other) const
- virtual Eigen::Affine3d **GetTransformToFrame** (FrameIndex other) const
- const std::string & **name** () const
- FrameIndex **index** () const
- FrameType **type** () const
- bool **is\_user\_defined** () const

### Protected Member Functions

- **Frame** (FrameIndex index, const std::string &name, FrameType type, bool is\_user\_defined, std::weak\_ptr< const [Model](#) > model)

### Protected Attributes

- const FrameIndex **index\_**  
[Frame](#) name.
- const std::string **name\_**
- const FrameType **type\_**
- bool **is\_user\_defined\_**
- const std::weak\_ptr< const [Model](#) > **model\_**

### Friends

- class **Model**

## Additional Inherited Members

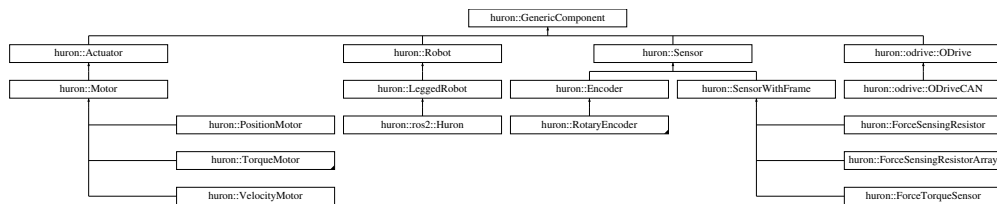
The documentation for this class was generated from the following files:

- /github/workspace/multibody/include/huron/multibody/frame.h
- /github/workspace/multibody/src/frame.cc

## 8.21 huron::GenericComponent Class Reference

```
#include <generic_component.h>
```

Inheritance diagram for huron::GenericComponent:



## Public Member Functions

- **GenericComponent** (std::unique\_ptr< [Configuration](#) > config)
- **GenericComponent** (const [GenericComponent](#) &)=delete
- [GenericComponent](#) & **operator=** (const [GenericComponent](#) &)=delete
- void [Configure](#) (std::string config\_key, std::any config\_value)
- void [Configure](#) (ConfigMap config)
- void [Configure](#) (std::unique\_ptr< [Configuration](#) > config\_ptr)
- virtual void [Initialize](#) ()=0
- virtual void [SetUp](#) ()=0
- virtual void [Terminate](#) ()=0

## Protected Member Functions

- virtual void [ConfigureKey](#) (std::string config\_key, std::any config\_value)
- virtual void [ConfigureMap](#) (const ConfigMap &config\_map)

## Protected Attributes

- std::unique\_ptr< [Configuration](#) > [config\\_](#)

### 8.21.1 Detailed Description

Interface for all components.

### 8.21.2 Member Function Documentation

#### 8.21.2.1 Configure() [1/3]

```
void huron::GenericComponent::Configure (
    ConfigMap config ) [inline]
```

Configure using a ConfigMap. A necessary condition for this operation is that all keys in the ConfigMap are valid.

**8.21.2.2 Configure()** [2/3]

```
void huron::GenericComponent::Configure (
    std::string config_key,
    std::any config_value ) [inline]
```

Configure using a key-value pair.

**8.21.2.3 Configure()** [3/3]

```
void huron::GenericComponent::Configure (
    std::unique_ptr< Configuration > config_ptr ) [inline]
```

Replace the underlying [Configuration](#) object by a new one.

**8.21.2.4 ConfigureKey()**

```
virtual void huron::GenericComponent::ConfigureKey (
    std::string config_key,
    std::any config_value ) [inline], [protected], [virtual]
```

Configure the hardware component with the specified key-value pair. This method needs to be defined by the user.

**Precondition**

The configuration pair is valid and stored into config\_.

Reimplemented in [huron::odrive::ODrive](#).

**8.21.2.5 ConfigureMap()**

```
virtual void huron::GenericComponent::ConfigureMap (
    const ConfigMap & config_map ) [inline], [protected], [virtual]
```

Configure the hardware component with the specified configuration map.

**Precondition**

The configuration map is valid.

**8.21.2.6 Initialize()**

```
virtual void huron::GenericComponent::Initialize ( ) [pure virtual]
```

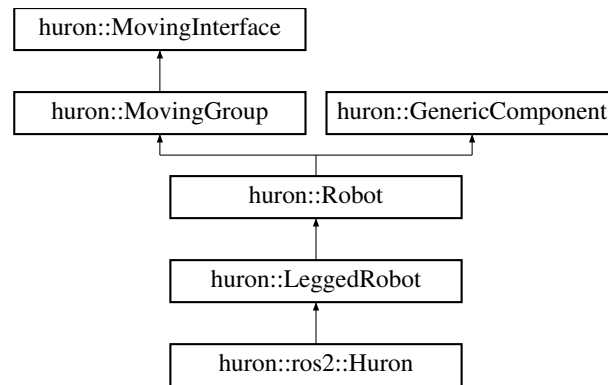
Implemented in [huron::odrive::ODrive](#).

The documentation for this class was generated from the following file:

- [/github/workspace/system/control\\_interfaces/include/huron/control\\_interfaces/generic\\_component.h](#)

**8.22 huron::ros2::Huron Class Reference**

Inheritance diagram for huron::ros2::Huron:



## Public Member Functions

- **Huron** (std::shared\_ptr< [HuronNode](#) > node, std::unique\_ptr< [huron::RobotConfiguration](#) > config)
- **Huron** (std::shared\_ptr< [HuronNode](#) > node)
- **Huron** (const [Huron](#) &)=delete
- **Huron** & **operator=** (const [Huron](#) &)=delete
- void **Initialize** () override
- void **SetUp** () override
- void **Terminate** () override
- void **Loop** ()

## Additional Inherited Members

### 8.22.1 Member Function Documentation

#### 8.22.1.1 Initialize()

void huron::ros2::Huron::Initialize ( ) [override], [virtual]  
 Implements [huron::GenericComponent](#).

#### 8.22.1.2 SetUp()

void huron::ros2::Huron::SetUp ( ) [override], [virtual]  
 Implements [huron::GenericComponent](#).

#### 8.22.1.3 Terminate()

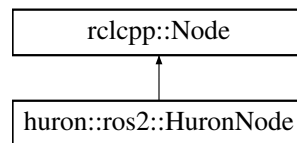
void huron::ros2::Huron::Terminate ( ) [override], [virtual]  
 Implements [huron::GenericComponent](#).

The documentation for this class was generated from the following files:

- /github/workspace/ros2/src/huron\_ros2/include/huron\_ros2/huron.h
- /github/workspace/ros2/src/huron\_ros2/src/huron.cc

## 8.23 huron::ros2::HuronNode Class Reference

Inheritance diagram for huron::ros2::HuronNode:



## Public Member Functions

- void **JointStateCallback** (std::shared\_ptr< const sensor\_msgs::msg::JointState > msg)
- void **OdomCallback** (std::shared\_ptr< const nav\_msgs::msg::Odometry > msg)
- void **WrenchStampedCallback** (size\_t idx, std::shared\_ptr< const geometry\_msgs::msg::WrenchStamped > msg)
- void **PublishFloat64MultiArray** (size\_t idx, const std::vector< double > &values)
- const huron::Vector6d & **GetWrench** (size\_t idx) const
- void **AddJointStateProvider** (std::shared\_ptr< [JointStateProvider](#) > jsp, const std::string &topic, size\_t nq, size\_t nv, bool is\_odom=false)
- *Add a subscriber to the joint state topic. There can be at most one subscriber to the joint state topic.*
- void **AddForceTorqueSensor** (std::shared\_ptr< [ForceTorqueSensor](#) > ft\_sensor, const std::string &topic)
- void **AddJointGroupController** (std::shared\_ptr< [JointGroupController](#) > jgc, const std::string &topic)
- void **Finalize** ()
- *Finalize the configuration. This method must be called after adding all the ROS2 components to the node, else exceptions will be thrown.*
- Eigen::VectorXd **GetJointState** (size\_t id\_q, size\_t dim\_q, size\_t id\_v, size\_t dim\_v) const

The documentation for this class was generated from the following files:

- /github/workspace/ros2/src/huron\_ros2/include/huron\_ros2/huron\_node.h
- /github/workspace/ros2/src/huron\_ros2/src/huron\_node.cc

## 8.24 huron::multibody::internal::PinocchioModelImpl::Impl Struct Reference

### Public Attributes

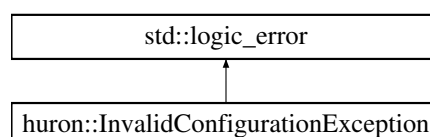
- int **dummy**
- pinocchio::Model **model\_**
- pinocchio::Data **data\_**

The documentation for this struct was generated from the following files:

- /github/workspace/multibody/src/no\_pinocchio\_model\_impl.cc
- /github/workspace/multibody/src/pinocchio\_model\_impl.cc

## 8.25 huron::InvalidConfigurationException Class Reference

Inheritance diagram for huron::InvalidConfigurationException:





## Public Member Functions

- **InvalidConfigurationException** (const char \*message)
- virtual const char \* **what** () const throw ()

The documentation for this class was generated from the following file:

- /github/workspace/system/exceptions/include/huron/exceptions/invalid\_configuration\_exception.h

## 8.26 huron::Joint Class Reference

### Public Member Functions

- **Joint** (std::unique\_ptr< [JointDescription](#) > joint\_desc, std::shared\_ptr< [StateProvider](#) > state\_provider=nullptr)
- **Joint** (const [Joint](#) &)=delete
- **Joint & operator=** (const [Joint](#) &)=delete
- void **SetIndices** (size\_t id\_q, size\_t id\_v)

### 8.26.1 Constructor & Destructor Documentation

#### 8.26.1.1 Joint()

```
huron::Joint::Joint (
    std::unique_ptr< JointDescription > joint_desc,
    std::shared_ptr< StateProvider > state_provider = nullptr ) [explicit]
```

Creates a [Joint](#) that connects the specified parent and child frames.

The documentation for this class was generated from the following files:

- /github/workspace/system/control\_interfaces/include/huron/control\_interfaces/joint.h
- /github/workspace/system/control\_interfaces/src/joint.cc

## 8.27 huron::multibody::JointDescription Struct Reference

### Public Member Functions

- **JointDescription** (size\_t id, const std::string &name, size\_t parent\_frame\_id, size\_t child\_frame\_id, size\_t num\_positions, size\_t num\_velocities, JointType type, const Eigen::VectorXd &min\_position, const Eigen::VectorXd &max\_position, const Eigen::VectorXd &min\_velocity, const Eigen::VectorXd &max\_velocity, const Eigen::VectorXd &min\_acceleration, const Eigen::VectorXd &max\_acceleration, const Eigen::VectorXd &min\_torque, const Eigen::VectorXd &max\_torque, const Eigen::VectorXd &friction, const Eigen::VectorXd &damping)
- **JointDescription** (size\_t id, const std::string &name, size\_t parent\_frame\_id, size\_t child\_frame\_id, size\_t num\_positions, size\_t num\_velocities, JointType type, const Eigen::VectorXd &min\_position, const Eigen::VectorXd &max\_position, const Eigen::VectorXd &min\_velocity, const Eigen::VectorXd &max\_velocity, const Eigen::VectorXd &min\_acceleration, const Eigen::VectorXd &max\_acceleration, const Eigen::VectorXd &min\_torque, const Eigen::VectorXd &max\_torque)
- **JointDescription** (size\_t id, const std::string &name, size\_t parent\_frame\_id, size\_t child\_frame\_id, size\_t num\_positions, size\_t num\_velocities, JointType type)
- JointIndex **id** () const
- const std::string & **name** () const
- FrameIndex **parent\_frame\_id** () const
- FrameIndex **child\_frame\_id** () const
- size\_t **num\_positions** () const
- size\_t **num\_velocities** () const
- JointType **type** () const

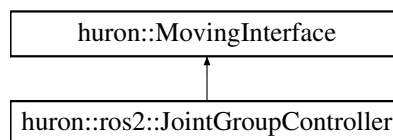
- `const Eigen::VectorXd & min_position () const`
- `const Eigen::VectorXd & max_position () const`
- `const Eigen::VectorXd & min_velocity () const`
- `const Eigen::VectorXd & max_velocity () const`
- `const Eigen::VectorXd & min_acceleration () const`
- `const Eigen::VectorXd & max_acceleration () const`
- `const Eigen::VectorXd & min_torque () const`
- `const Eigen::VectorXd & max_torque () const`
- `const Eigen::VectorXd & friction () const`
- `const Eigen::VectorXd & damping () const`

The documentation for this struct was generated from the following file:

- `/github/workspace/multibody/include/huron/multibody/joint_common.h`

## 8.28 huron::ros2::JointGroupController Class Reference

Inheritance diagram for `huron::ros2::JointGroupController`:



### Public Member Functions

- **JointGroupController** (size\_t dim)
- **JointGroupController** (const [JointGroupController](#) &)=delete
- [JointGroupController](#) & **operator=** (const [JointGroupController](#) &)=delete
- bool [Move](#) (const std::vector< double > &values) override
- bool [Move](#) (const Eigen::VectorXd &values) override
- bool [Stop](#) () override

### Friends

- class **HuronNode**

### Additional Inherited Members

#### 8.28.1 Member Function Documentation

##### 8.28.1.1 Move() [1/2]

```
bool huron::ros2::JointGroupController::Move (
    const Eigen::VectorXd & values ) [override], [virtual]
```

Implements [huron::MovingInterface](#).

##### 8.28.1.2 Move() [2/2]

```
bool huron::ros2::JointGroupController::Move (
    const std::vector< double > & values ) [override], [virtual]
```

Moves the component by the specified input vector.

This method can be used if the component needs more than one input. For example, a position controlled motor needs position input, velocity feedforward, and current feedforward.

## Parameters

<i>value</i>	Input value vector.
--------------	---------------------

## Returns

true if the operation is successful, false otherwise.

Implements [huron::MovingInterface](#).

**8.28.1.3 Stop()**

```
bool huron::ros2::JointGroupController::Stop ( ) [override], [virtual]
```

Stops the component from moving.

## Returns

true if the operation is successful, false otherwise.

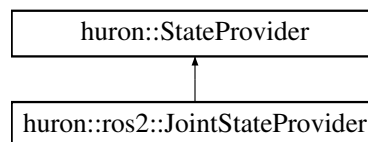
Implements [huron::MovingInterface](#).

The documentation for this class was generated from the following files:

- /github/workspace/ros2/src/huron\_ros2/include/huron\_ros2/joint\_group\_controller.h
- /github/workspace/ros2/src/huron\_ros2/src/joint\_group\_controller.cc

**8.29 huron::ros2::JointStateProvider Class Reference**

Inheritance diagram for huron::ros2::JointStateProvider:

**Public Member Functions**

- **JointStateProvider** (size\_t id\_q, size\_t nq, size\_t id\_v, size\_t nv)
- void [RequestStateUpdate](#) () override
- void [GetNewState](#) (Eigen::Ref< Eigen::MatrixXd > new\_state) const override

**Friends**

- class **HuronNode**

**8.29.1 Member Function Documentation****8.29.1.1 GetNewState()**

```
void huron::ros2::JointStateProvider::GetNewState (
    Eigen::Ref< Eigen::MatrixXd > new_state ) const [override], [virtual]
```

Implements [huron::StateProvider](#).

### 8.29.1.2 RequestStateUpdate()

```
void huron::ros2::JointStateProvider::RequestStateUpdate ( ) [override], [virtual]
```

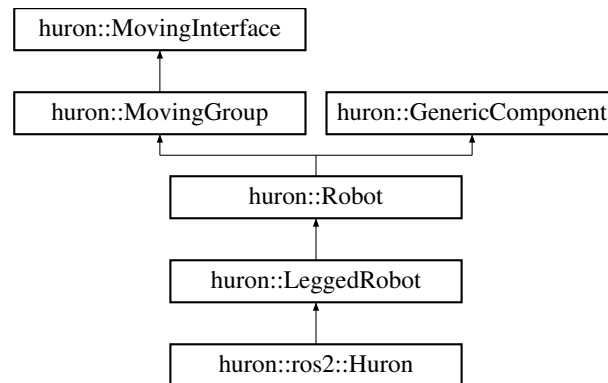
Implements [huron::stateProvider](#).

The documentation for this class was generated from the following files:

- /github/workspace/ros2/src/huron\_ros2/include/huron\_ros2/joint\_state\_provider.h
- /github/workspace/ros2/src/huron\_ros2/src/joint\_state\_provider.cc

## 8.30 huron::LeggedRobot Class Reference

Inheritance diagram for huron::LeggedRobot:



### Public Member Functions

- **LeggedRobot** (std::unique\_ptr< [RobotConfiguration](#) > config)
- **LeggedRobot** (const [LeggedRobot](#) &)=delete
- **LeggedRobot & operator=** (const [LeggedRobot](#) &)=delete
- void **InitializeZmp** (std::shared\_ptr< [ZeroMomentPoint](#) > zmp)
- Eigen::Vector2d **EvalZeroMomentPoint** ()

### Additional Inherited Members

#### 8.30.1 Member Function Documentation

##### 8.30.1.1 EvalZeroMomentPoint()

```
Eigen::Vector2d huron::LeggedRobot::EvalZeroMomentPoint ( )
```

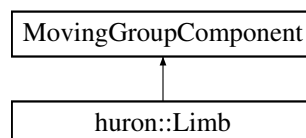
Computes the Center of Mass in Base frame.

The documentation for this class was generated from the following files:

- /github/workspace/system/control\_interfaces/include/huron/control\_interfaces/legged\_robot.h
- /github/workspace/system/control\_interfaces/src/legged\_robot.cc

## 8.31 huron::Limb Class Reference

Inheritance diagram for huron::Limb:



## Public Member Functions

- void **Init** (std::vector< [Joint](#) > joints)
- void **AddJoint** ([Joint](#) &joint)

The documentation for this class was generated from the following files:

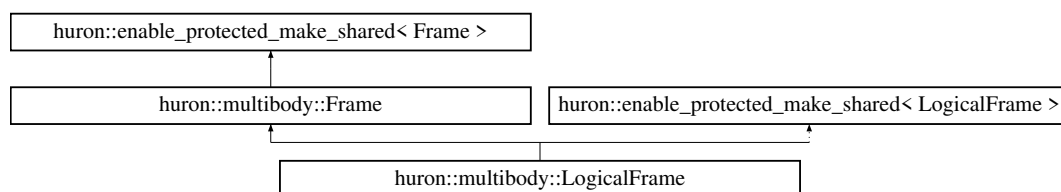
- /github/workspace/system/control\_interfaces/include/huron/control\_interfaces/limb.h
- /github/workspace/system/control\_interfaces/src/limb.cc

## 8.32 huron::multibody::LogicalFrame Class Reference

A frame that is defined relative to another frame by an affine transformation. This transformation is user-defined using a function that takes the parent frame's transform in world coordinates as an argument and returns the transform from the parent frame to this frame.

```
#include <logical_frame.h>
```

Inheritance diagram for huron::multibody::LogicalFrame:



## Public Member Functions

- **LogicalFrame** (const [LogicalFrame](#) &)=delete
- **LogicalFrame** & **operator=** (const [LogicalFrame](#) &)=delete
- Eigen::Affine3d **GetTransformInWorld** () const override
- Eigen::Affine3d **GetTransformFromFrame** (const [Frame](#) &other) const override
- Eigen::Affine3d **GetTransformFromFrame** (FrameIndex other) const override
- Eigen::Affine3d **GetTransformToFrame** (const [Frame](#) &other) const override
- Eigen::Affine3d **GetTransformToFrame** (FrameIndex other) const override

## Protected Member Functions

- **LogicalFrame** (FrameIndex index, const std::string &name, bool is\_user\_defined, std::weak\_ptr< const [Model](#) > model, FrameIndex parent\_frame\_index, std::function< Eigen::Affine3d(const Eigen::Affine3d &)> transform\_function)

## Friends

- class **Model**

## Additional Inherited Members

### 8.32.1 Detailed Description

A frame that is defined relative to another frame by an affine transformation. This transformation is user-defined using a function that takes the parent frame's transform in world coordinates as an argument and returns the transform from the parent frame to this frame.

#### Note

This class can only be instantiated by the [Model](#) class using AddLogicalFrame().

## Parameters

<i>index</i>	The index of this frame.
<i>name</i>	The name of this frame.
<i>model</i>	The model that this frame is a part of.
<i>parent_frame_index</i>	The index of the frame that this frame is defined relative to.
<i>transform_function</i>	The function that defines the transformation from the parent frame to this frame.

## 8.32.2 Member Function Documentation

### 8.32.2.1 GetTransformFromFrame() [1/2]

```
Eigen::Affine3d huron::multibody::LogicalFrame::GetTransformFromFrame (
    const Frame & other ) const [override], [virtual]
```

Reimplemented from [huron::multibody::Frame](#).

### 8.32.2.2 GetTransformFromFrame() [2/2]

```
Eigen::Affine3d huron::multibody::LogicalFrame::GetTransformFromFrame (
    FrameIndex other ) const [override], [virtual]
```

Reimplemented from [huron::multibody::Frame](#).

### 8.32.2.3 GetTransformInWorld()

```
Eigen::Affine3d huron::multibody::LogicalFrame::GetTransformInWorld ( ) const [override],
[virtual]
```

Reimplemented from [huron::multibody::Frame](#).

### 8.32.2.4 GetTransformToFrame() [1/2]

```
Eigen::Affine3d huron::multibody::LogicalFrame::GetTransformToFrame (
    const Frame & other ) const [override], [virtual]
```

Reimplemented from [huron::multibody::Frame](#).

### 8.32.2.5 GetTransformToFrame() [2/2]

```
Eigen::Affine3d huron::multibody::LogicalFrame::GetTransformToFrame (
    FrameIndex other ) const [override], [virtual]
```

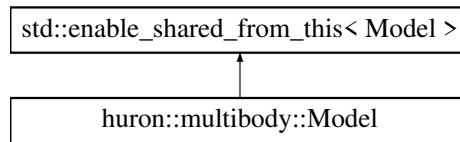
Reimplemented from [huron::multibody::Frame](#).

The documentation for this class was generated from the following files:

- [/github/workspace/multibody/include/huron/multibody/logical\\_frame.h](#)
- [/github/workspace/multibody/src/logical\\_frame.cc](#)

## 8.33 huron::multibody::Model Class Reference

Inheritance diagram for [huron::multibody::Model](#):



## Public Member Functions

- **Model** (const [Model](#) &)=delete
- **Model & operator=** (const [Model](#) &)=delete
- void **AddModelImpl** (ModelImplType type, bool set\_as\_default=false)
- template<typename ... Args>  
void **AddJoint** (JointIndex index, Args &&... args)
- [Joint](#) \*const **GetJoint** (JointIndex index)
- [Joint](#) \*const **GetJoint** (const std::string &name)
- void **SetJointStateProvider** (JointIndex index, std::shared\_ptr< [StateProvider](#) > state\_provider)
- JointIndex **GetJointIndex** (const std::string &joint\_name) const
- template<typename FrameImpl , typename ... Args>  
std::weak\_ptr< const [Frame](#) > **AddFrame** (const std::string &name, Args &&... args)
- std::weak\_ptr< const [Frame](#) > **GetFrame** (FrameIndex index) const
- std::weak\_ptr< const [Frame](#) > **GetFrame** (const std::string &name) const
- void **BuildFromUrdf** (const std::string &urdf\_path, JointType root\_joint\_type=JointType::kFixed)
- void **Finalize** (const Eigen::VectorXd &initial\_state)
- void **Finalize** ()
- void **UpdateJointStates** ()
- void **SetDefaultModelImpl** (size\_t index)
- size\_t **GetDefaultModelImpl** () const
- Eigen::Affine3d **GetJointTransformInWorld** (size\_t joint\_index) const
- FrameIndex **GetFrameIndex** (const std::string &frame\_name) const
- const std::string & **GetFrameName** (FrameIndex frame\_index) const
- Eigen::Affine3d **GetFrameTransform** (FrameIndex from\_frame, FrameIndex to\_frame) const
- Eigen::Affine3d **GetFrameTransformInWorld** (FrameIndex frame) const
- Eigen::VectorXd **NeutralConfiguration** () const
- Eigen::Vector3d **EvalCenterOfMassPosition** ()
- Eigen::Vector3d **GetCenterOfMassPosition** () const
- const Eigen::VectorBlock< const Eigen::VectorXd > **GetPositions** () const
- const Eigen::VectorBlock< const Eigen::VectorXd > **GetVelocities** () const
- const Eigen::VectorXd & **GetAccelerations** () const  
*Get the generalized accelerations of the model.*
- const Eigen::VectorXd & **GetTorques** () const  
*Get the joint torques.*
- const Eigen::MatrixXd & **GetMassMatrix** () const  
*Get the mass matrix with the cached value.*
- const Eigen::MatrixXd & **GetCoriolisMatrix** () const  
*Get the Coriolis matrix with the cached value.*
- const Eigen::VectorXd & **GetNonlinearEffects** () const  
*Get the nonlinear effects vector.*
- const Eigen::VectorXd & **GetGravity** () const  
*Get the gravity vector.*
- const huron::Vector6d & **GetSpatialMomentum** () const  
*Get the spatial momentum with respect to the specified frame.*
- huron::Vector6d **GetCentroidalMomentum** () const  
*Get the centroidal momentum.*

- `const huron::Matrix6Xd & GetCentroidalMatrix () const`  
*Get the centroidal momentum matrix with the cached value.*
- `void ComputeAll ()`
- `void ForwardKinematics ()`
- `bool is_finalized () const`
- `size_t num_positions () const`
- `size_t num_velocities () const`
- `size_t num_joints () const`
- `size_t num_frames () const`

## Protected Member Functions

- `ModellImplInterface const * GetModellImpl (size_t index) const`
- `template<typename FrameImpl, typename ... Args>`  
`void DoAddFrame (const std::string &name, bool is_user_defined, Args &&... args)`
- `void DoAddFrameFromModelDescription (FrameIndex idx, const std::string &name, FrameType type)`

## Protected Attributes

- `size_t default_impl_index = 0`
- `std::vector< std::unique_ptr< ModellImplInterface > > impls_`
- `std::vector< std::shared_ptr< Joint > > joints_`
- `Eigen::VectorXd states_`  
*The joint states [q, v].*
- `size_t num_positions = 0`
- `size_t num_velocities = 0`
- `std::vector< std::shared_ptr< Frame > > frames_`  
*All frames, including those defined by the model description file and user-defined ones.*
- `std::unordered_map< std::string, FrameIndex > frame_name_to_index_`
- `bool is_constructed = false`
- `bool is_finalized = false`

## 8.33.1 Member Function Documentation

### 8.33.1.1 AddFrame()

```
template<typename FrameImpl, typename ... Args>
std::weak_ptr< const Frame > huron::multibody::Model::AddFrame (
    const std::string & name,
    Args &&... args ) [inline]
```

Adds a frame to the model. Currently supported formats for external users:

- `AddFrame<LogicalFrame>(name, parent_frame, transform_function)`

#### Note

As of now, frames can only be added after the model is built from URDF. This will be changed in the future.

#### Parameters

<i>index</i>	The index of the frame.
<i>args</i>	Arguments to be passed to the constructor of the frame.



### 8.33.1.2 AddJoint()

```
template<typename ... Args>
void huron::multibody::Model::AddJoint (
    JointIndex index,
    Args &&... args ) [inline]
```

Adds a joint to the model.

#### Parameters

<i>index</i>	The index of the joint.
<i>args</i>	Arguments to be passed to the constructor of the joint.

### 8.33.1.3 AddModelImpl()

```
void huron::multibody::Model::AddModelImpl (
    ModelImplType type,
    bool set_as_default = false )
```

Adds a model implementation to the model.

#### Parameters

<i>type</i>	The type of the model implementation.
<i>set_as_default</i>	If true, the model implementation will be set as the default model implementation.

#### Exceptions

<i>std::runtime_error</i>	if the model implementation is not available.
---------------------------	---

### 8.33.1.4 Finalize()

```
void huron::multibody::Model::Finalize (
    const Eigen::VectorXd & initial_state )
```

Performs final configuration and checks the validity of the model:

- Checks if all joints are added to the model.
- Adjusts the joint state indices in the states vector. This method also sets the initial state [q, v] of the model.

#### Exceptions

<i>std::runtime_error</i>	if the model is not valid.
---------------------------	----------------------------

### 8.33.1.5 GetModelImpl()

```
internal::ModelImplInterface const * huron::multibody::Model::GetModelImpl (
    size_t index ) const [protected]
```

Returns the model implementation at the given index. This function is provided for testing subclasses only.

### 8.33.1.6 UpdateJointStates()

```
void huron::multibody::Model::UpdateJointStates ( )
```

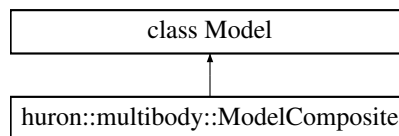
Updates the joint states [q, v] of the model.

The documentation for this class was generated from the following files:

- /github/workspace/multibody/include/huron/multibody/model.h
- /github/workspace/multibody/src/model.cc

## 8.34 huron::multibody::ModelComposite Class Reference

Inheritance diagram for huron::multibody::ModelComposite:



### Public Member Functions

- void **RegisterModel** (std::unique\_ptr< [Model](#) > model)

The documentation for this class was generated from the following file:

- /github/workspace/multibody/include/huron/multibody/model\_composite.h

## 8.35 huron::multibody::internal::ModelImplFactory Class Reference

### Public Member Functions

- **ModelImplFactory** (const [ModelImplFactory](#) &)=delete
- [ModelImplFactory](#) & **operator=** (const [ModelImplFactory](#) &)=delete

### Friends

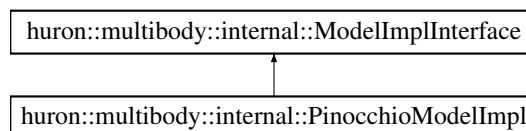
- class **multibody::Model**

The documentation for this class was generated from the following file:

- /github/workspace/multibody/include/huron/multibody/model\_impl\_factory.h

## 8.36 huron::multibody::internal::ModelImplInterface Class Reference

Inheritance diagram for huron::multibody::internal::ModelImplInterface:



## Public Member Functions

- **ModelImplInterface** (const [ModelImplInterface](#) &)=delete
- **ModelImplInterface** & **operator=** (const [ModelImplInterface](#) &)=delete
- virtual void **BuildFromUrdf** (const std::string &urdf\_path, JointType root\_joint\_type)
- virtual const std::vector< std::string > & **GetJointNames** () const
- virtual std::weak\_ptr< [Joint](#) > **GetJoint** (const std::string &name) const
- virtual std::weak\_ptr< [Joint](#) > **GetJoint** (size\_t joint\_index) const
- virtual JointType **GetJointType** (size\_t joint\_index) const
- virtual JointIndex **GetJointIndex** (const std::string &joint\_name) const =0
- virtual std::unique\_ptr< [JointDescription](#) > **GetJointDescription** (JointIndex joint\_index) const
- virtual std::unique\_ptr< [JointDescription](#) > **GetJointDescription** (const std::string &joint\_name) const
- virtual Eigen::Affine3d **GetJointTransformInWorld** (size\_t joint\_index) const
- virtual FrameIndex **GetFrameIndex** (const std::string &frame\_name) const
- virtual const std::string & **GetFrameName** (FrameIndex frame\_index) const
- virtual FrameType **GetFrameType** (FrameIndex frame\_index) const
- virtual Eigen::Affine3d **GetFrameTransform** (FrameIndex from\_frame, FrameIndex to\_frame) const
- virtual Eigen::Affine3d **GetFrameTransformInWorld** (FrameIndex frame) const
- virtual Eigen::Vector3d **EvalCenterOfMassPosition** ()
- virtual Eigen::Vector3d **GetCenterOfMassPosition** () const
- virtual Eigen::VectorXd **NeutralConfiguration** () const
- virtual const Eigen::VectorXd & **GetAccelerations** () const  
*Get the generalized accelerations of the model.*
- virtual const Eigen::VectorXd & **GetTorques** () const  
*Get the joint torques.*
- virtual const Eigen::MatrixXd & **GetMassMatrix** () const  
*Get the mass matrix with the cached value.*
- virtual const Eigen::MatrixXd & **GetCoriolisMatrix** () const  
*Get the Coriolis matrix with the cached value.*
- virtual const Eigen::VectorXd & **GetNonlinearEffects** () const  
*Get the nonlinear effects vector.*
- virtual const Eigen::VectorXd & **GetGravity** () const  
*Get the gravity vector.*
- virtual const huron::Vector6d & **GetSpatialMomentum** () const  
*Get the spatial momentum with respect to the specified frame.*
- virtual huron::Vector6d **GetCentroidalMomentum** () const  
*Get the centroidal momentum.*
- virtual const huron::Matrix6Xd & **GetCentroidalMatrix** () const  
*Get the centroidal momentum matrix with the cached value.*
- virtual void **ComputeAll** (const Eigen::Ref< const Eigen::VectorXd > &q, const Eigen::Ref< const Eigen::VectorXd > &v)
- virtual void **ForwardKinematics** (const Eigen::Ref< const Eigen::VectorXd > &q)
- virtual void **ForwardKinematics** (const Eigen::Ref< const Eigen::VectorXd > &q, const Eigen::Ref< const Eigen::VectorXd > &v)
- virtual void **ForwardKinematics** (const Eigen::Ref< const Eigen::VectorXd > &q, const Eigen::Ref< const Eigen::VectorXd > &v, const Eigen::Ref< const Eigen::VectorXd > &a)
- virtual bool **is\_built** () const
- virtual size\_t **num\_positions** () const
- virtual size\_t **num\_velocities** () const
- virtual size\_t **num\_joints** () const
- virtual size\_t **num\_frames** () const

### 8.36.1 Member Function Documentation

**8.36.1.1 GetAccelerations()**

```
const Eigen::VectorXd & huron::multibody::internal::ModelImplInterface::GetAccelerations ( )
const [virtual]
```

Get the generalized accelerations of the model.

Reimplemented in [huron::multibody::internal::PinocchioModelImpl](#).

**8.36.1.2 GetCentroidalMatrix()**

```
const huron::Matrix6Xd & huron::multibody::internal::ModelImplInterface::GetCentroidalMatrix (
) const [virtual]
```

Get the centroidal momentum matrix with the cached value.

Reimplemented in [huron::multibody::internal::PinocchioModelImpl](#).

**8.36.1.3 GetCentroidalMomentum()**

```
huron::Vector6d huron::multibody::internal::ModelImplInterface::GetCentroidalMomentum ( )
const [virtual]
```

Get the centroidal momentum.

Reimplemented in [huron::multibody::internal::PinocchioModelImpl](#).

**8.36.1.4 GetCoriolisMatrix()**

```
const Eigen::MatrixXd & huron::multibody::internal::ModelImplInterface::GetCoriolisMatrix ( )
const [virtual]
```

Get the Coriolis matrix with the cached value.

Reimplemented in [huron::multibody::internal::PinocchioModelImpl](#).

**8.36.1.5 GetGravity()**

```
const Eigen::VectorXd & huron::multibody::internal::ModelImplInterface::GetGravity ( ) const
[virtual]
```

Get the gravity vector.

Reimplemented in [huron::multibody::internal::PinocchioModelImpl](#).

**8.36.1.6 GetMassMatrix()**

```
const Eigen::MatrixXd & huron::multibody::internal::ModelImplInterface::GetMassMatrix ( )
const [virtual]
```

Get the mass matrix with the cached value.

Reimplemented in [huron::multibody::internal::PinocchioModelImpl](#).

**8.36.1.7 GetNonlinearEffects()**

```
const Eigen::VectorXd & huron::multibody::internal::ModelImplInterface::GetNonlinearEffects (
) const [virtual]
```

Get the nonlinear effects vector.

Reimplemented in [huron::multibody::internal::PinocchioModelImpl](#).

**8.36.1.8 GetSpatialMomentum()**

```
const huron::Vector6d & huron::multibody::internal::ModelImplInterface::GetSpatialMomentum ( )
const [virtual]
```

Get the spatial momentum with respect to the specified frame.  
Reimplemented in [huron::multibody::internal::PinocchioModelImpl](#).

### 8.36.1.9 GetTorques()

```
const Eigen::VectorXd & huron::multibody::internal::ModelImplInterface::GetTorques ( ) const
[virtual]
```

Get the joint torques.

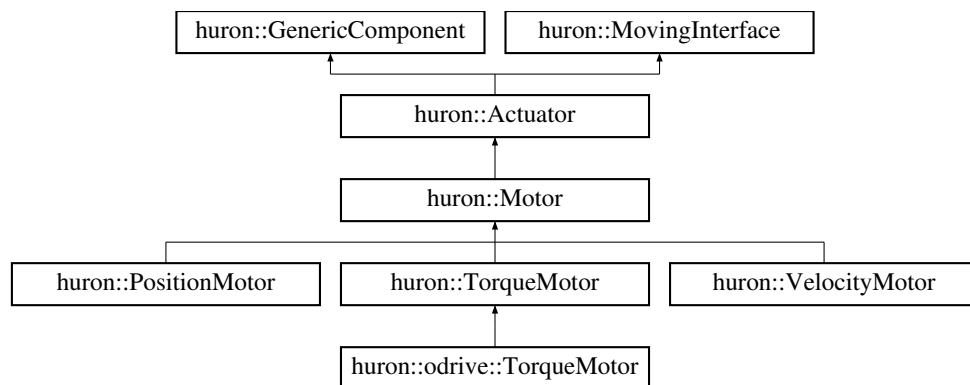
Reimplemented in [huron::multibody::internal::PinocchioModelImpl](#).

The documentation for this class was generated from the following files:

- /github/workspace/multibody/include/huron/multibody/model\_impl\_interface.h
- /github/workspace/multibody/src/model\_impl\_default.cc

## 8.37 huron::Motor Class Reference

Inheritance diagram for huron::Motor:



### Public Member Functions

- **Motor** (std::unique\_ptr< [MotorConfiguration](#) > config, double gear\_ratio=1.0)
- **Motor** (double gear\_ratio)
- **Motor** (const [Motor](#) &)=delete
- [Motor](#) & **operator=** (const [Motor](#) &)=delete
- virtual bool **Move** (double value)=0

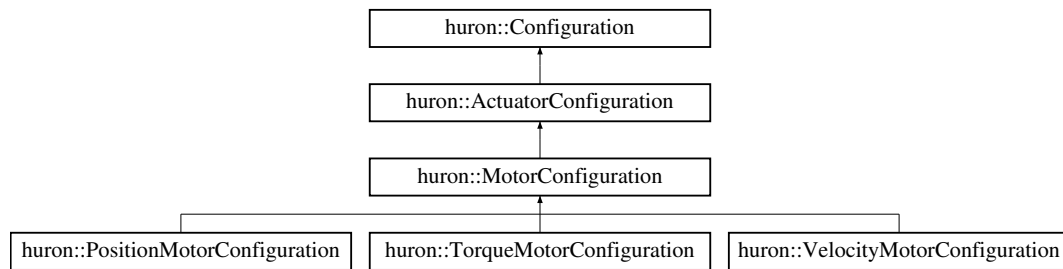
### Additional Inherited Members

The documentation for this class was generated from the following file:

- /github/workspace/system/control\_interfaces/include/huron/control\_interfaces/motor.h

## 8.38 huron::MotorConfiguration Class Reference

Inheritance diagram for huron::MotorConfiguration:



## Public Member Functions

- **MotorConfiguration** (ConfigMap config\_map, std::set< std::string > valid\_keys)
- **MotorConfiguration** (ConfigMap config\_map)

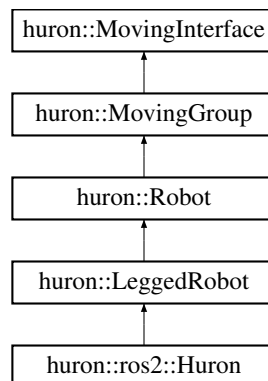
## Additional Inherited Members

The documentation for this class was generated from the following file:

- /github/workspace/system/control\_interfaces/include/huron/control\_interfaces/motor.h

## 8.39 huron::MovingGroup Class Reference

Inheritance diagram for huron::MovingGroup:



## Public Member Functions

- **MovingGroup** (const [MovingGroup](#) &)=delete
- [MovingGroup](#) & **operator=** (const [MovingGroup](#) &)=delete
- virtual void **AddToGroup** (std::shared\_ptr< [MovingInterface](#) > component)
- bool [Move](#) (const std::vector< double > &values) override
- bool [Move](#) (const Eigen::VectorXd &values) override
- bool [Stop](#) () override

## Protected Attributes

- std::vector< std::shared\_ptr< [MovingInterface](#) > > **moving\_components\_**
- std::vector< size\_t > **moving\_interface\_dims\_**

### 8.39.1 Member Function Documentation

**8.39.1.1 Move()** [1/2]

```
bool huron::MovingGroup::Move (
    const Eigen::VectorXd & values ) [override], [virtual]
```

Implements [huron::MovingInterface](#).

**8.39.1.2 Move()** [2/2]

```
bool huron::MovingGroup::Move (
    const std::vector< double > & values ) [override], [virtual]
```

Moves the component by the specified input vector.

This method can be used if the component needs more than one input. For example, a position controlled motor needs position input, velocity feedforward, and current feedforward.

**Parameters**

<i>value</i>	Input value vector.
--------------	---------------------

**Returns**

true if the operation is successful, false otherwise.

Implements [huron::MovingInterface](#).

**8.39.1.3 Stop()**

```
bool huron::MovingGroup::Stop ( ) [override], [virtual]
```

Stops the component from moving.

**Returns**

true if the operation is successful, false otherwise.

Implements [huron::MovingInterface](#).

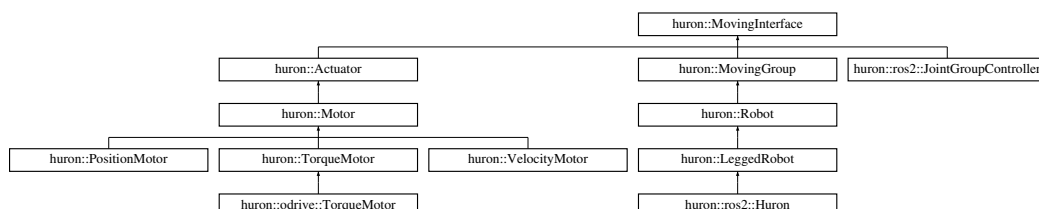
The documentation for this class was generated from the following files:

- /github/workspace/system/control\_interfaces/include/huron/control\_interfaces/moving\_group.h
- /github/workspace/system/control\_interfaces/src/moving\_group.cc

**8.40 huron::MovingInterface Class Reference**

```
#include <moving_interface.h>
```

Inheritance diagram for huron::MovingInterface:

**Public Member Functions**

- **MovingInterface** (size\_t dim)
- **MovingInterface** (const [MovingInterface](#) &)=delete
- **MovingInterface & operator=** (const [MovingInterface](#) &)=delete

- virtual bool [Move](#) (const std::vector< double > &values)=0
- virtual bool **Move** (const Eigen::VectorXd &values)=0
- virtual bool [Stop](#) ()=0
- size\_t **dim** () const

## Protected Attributes

- size\_t **dim\_**

### 8.40.1 Detailed Description

Interface for components that can move.

### 8.40.2 Member Function Documentation

#### 8.40.2.1 Move()

```
virtual bool huron::MovingInterface::Move (
    const std::vector< double > & values ) [pure virtual]
```

Moves the component by the specified input vector.

This method can be used if the component needs more than one input. For example, a position controlled motor needs position input, velocity feedforward, and current feedforward.

#### Parameters

<i>value</i>	Input value vector.
--------------	---------------------

#### Returns

true if the operation is successful, false otherwise.

Implemented in [huron::ros2::JointGroupController](#), [huron::MovingGroup](#), and [huron::odrive::TorqueMotor](#).

#### 8.40.2.2 Stop()

```
virtual bool huron::MovingInterface::Stop ( ) [pure virtual]
```

Stops the component from moving.

#### Returns

true if the operation is successful, false otherwise.

Implemented in [huron::ros2::JointGroupController](#), [huron::MovingGroup](#), and [huron::odrive::TorqueMotor](#).

The documentation for this class was generated from the following file:

- /github/workspace/system/control\_interfaces/include/huron/control\_interfaces/moving\_interface.h

## 8.41 huron::driver::can::MsgIdFilterSpecs Struct Reference

### Public Attributes

- std::variant< uint16\_t, uint32\_t > **id**
- uint32\_t **mask**

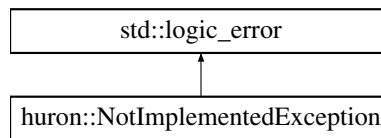
The documentation for this struct was generated from the following file:

- /github/workspace/driver/can/include/huron/driver/can/canbus.h



## 8.42 huron::NotImplementedException Class Reference

Inheritance diagram for huron::NotImplementedException:



### Public Member Functions

- **NotImplementedException** (const char \*function)
- virtual const char \* **what** () const throw ()

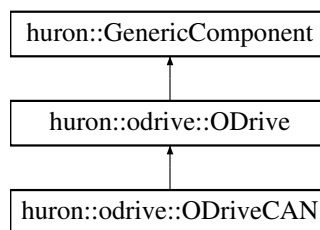
The documentation for this class was generated from the following file:

- /github/workspace/system/exceptions/include/huron/exceptions/not\_implemented\_exception.h

## 8.43 huron::odrive::ODrive Class Reference

```
#include <odrive.h>
```

Inheritance diagram for huron::odrive::ODrive:



### Classes

- class [ODriveConfiguration](#)

### Public Member Functions

- **ODrive** (std::unique\_ptr< [ODriveConfiguration](#) > config, uint32\_t get\_timeout)
- **ODrive** (uint32\_t get\_timeout=kGetTimeout)
- **ODrive** (const [ODrive](#) &)=delete
- [ODrive](#) & **operator=** (const [ODrive](#) &)=delete
- void [Initialize](#) () override
- bool [Calibrate](#) ()
- void [ConfigureKey](#) (std::string config\_key, std::any config\_value) override
- virtual bool **GetMotorError** (uint64\_t &motor\_error)=0
- virtual bool **GetEncoderError** (uint32\_t &encoder\_error)=0
- virtual bool **GetControllerError** (uint32\_t &controller\_error)=0
- virtual bool **GetSensorlessError** (uint32\_t &sensorless\_error)=0
- virtual bool **GetEncoderEstimates** (float &pos, float &vel)=0
- virtual bool **GetEncoderCount** (int32\_t &shadow\_cnt, int32\_t &cnt\_cpr)=0
- virtual bool **GetIq** (float &iq\_setpoint, float &iq\_measured)=0
- virtual bool **GetSensorlessEstimates** (float &pos, float &vel)=0
- virtual bool **GetBusVoltageCurrent** (float &bus\_voltage, float &bus\_current)=0
- virtual bool **GetAdcVoltage** (float &adc\_voltage)=0

- virtual bool **SetAxisNodeid** (uint32\_t axis\_id)=0
- virtual bool **SetAxisRequestedState** (uint32\_t state)=0
- virtual bool **SetAxisStartupConfig** ()=0
- virtual bool **SetInputPos** (float input\_pos, int16\_t vel\_ff, int16\_t torque\_ff)=0
- virtual bool **SetInputVel** (float input\_vel, float torque\_ff)=0
- virtual bool **SetInputTorque** (float input\_torque)=0
- virtual bool **SetControllerModes** (int32\_t control\_mode, int32\_t input\_mode)=0
- virtual bool **SetLimits** (float velocity\_limit, float current\_limit)=0
- virtual bool **SetTrajVelLimit** (float traj\_vel\_limit)=0
- virtual bool **SetTrajAccelLimits** (float traj\_accel\_limit, float traj\_decel\_limit)=0
- virtual bool **SetTrajInertia** (float traj\_inertia)=0
- virtual bool **SetLinearCount** (int32\_t position)=0
- virtual bool **SetPosGain** (float pos\_gain)=0
- virtual bool **SetVelGains** (float vel\_gain, float vel\_interator\_gain)=0
- virtual bool **Nmt** ()=0
- virtual bool **Estop** ()=0
- virtual bool **ClearErrors** ()=0
- virtual bool **StartAnticogging** ()=0

## Protected Attributes

- uint32\_t **get\_timeout\_**
- bool **is\_calibrated\_** = false

## Static Protected Attributes

- static const uint32\_t **kGetTimeout** = 100

## Additional Inherited Members

### 8.43.1 Detailed Description

Interface for using [ODrive](#) motor controllers.

### 8.43.2 Member Function Documentation

#### 8.43.2.1 Calibrate()

```
bool huron::odrive::ODrive::Calibrate ( )
```

Performs full calibration of the [ODrive](#).

#### 8.43.2.2 ConfigureKey()

```
void huron::odrive::ODrive::ConfigureKey (
    std::string config_key,
    std::any config_value ) [override], [virtual]
```

Configure the hardware component with the specified key-value pair. This method needs to be defined by the user.

#### Precondition

The configuration pair is valid and stored into config\_.

Reimplemented from [huron::GenericComponent](#).

### 8.43.2.3 Initialize()

```
void huron::odrive::ODrive::Initialize ( ) [override], [virtual]
```

Puts the [ODrive](#) in IDLE state and, if not completed before, perform full calibration.

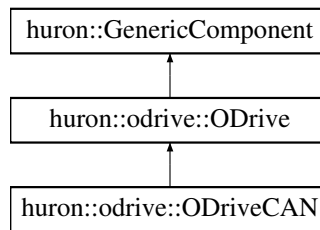
Implements [huron::GenericComponent](#).

The documentation for this class was generated from the following files:

- /github/workspace/system/odrive/include/huron/odrive/odrive.h
- /github/workspace/system/odrive/src/odrive.cc

## 8.44 huron::odrive::ODriveCAN Class Reference

Inheritance diagram for huron::odrive::ODriveCAN:



### Public Types

- enum {  
**MSG\_CO\_NMT\_CTRL** = 0x000 , **MSG\_ODRIVE\_HEARTBEAT** , **MSG\_ODRIVE\_ESTOP** , **MSG\_GET\_MOTOR\_ERROR** ,  
**MSG\_GET\_ENCODER\_ERROR** , **MSG\_GET\_SENSORLESS\_ERROR** , **MSG\_SET\_AXIS\_NODE\_ID** ,  
**MSG\_SET\_AXIS\_REQUESTED\_STATE** ,  
**MSG\_SET\_AXIS\_STARTUP\_CONFIG** , **MSG\_GET\_ENCODER\_ESTIMATES** , **MSG\_GET\_ENCODER\_COUNT** , **MSG\_SET\_CONTROLLER\_MODES** ,  
**MSG\_SET\_INPUT\_POS** , **MSG\_SET\_INPUT\_VEL** , **MSG\_SET\_INPUT\_TORQUE** , **MSG\_SET\_LIMITS** ,  
**MSG\_START\_ANTICOGGING** , **MSG\_SET\_TRAJ\_VEL\_LIMIT** , **MSG\_SET\_TRAJ\_ACCEL\_LIMITS** ,  
**MSG\_SET\_TRAJ\_INERTIA** ,  
**MSG\_GET\_IQ** , **MSG\_GET\_SENSORLESS\_ESTIMATES** , **MSG\_RESET\_ODRIVE** , **MSG\_GET\_BUS\_VOLTAGE\_CURRENT** ,  
**MSG\_CLEAR\_ERRORS** , **MSG\_SET\_LINEAR\_COUNT** , **MSG\_SET\_POS\_GAIN** , **MSG\_SET\_VEL\_GAINS** ,  
**MSG\_GET\_ADC\_VOLTAGE** , **MSG\_GET\_CONTROLLER\_ERROR** , **MSG\_CO\_HEARTBEAT\_CMD** = 0x700 }

### Public Member Functions

- [ODriveCAN](#) ([huron::driver::can::BusBase](#) \*canbus, uint32\_t axis\_id, std::unique\_ptr< [ODriveConfiguration](#) > config, uint32\_t get\_timeout=kGetTimeout)
- **ODriveCAN** (const [ODriveCAN](#) &)=delete
- [ODriveCAN](#) & **operator=** (const [ODriveCAN](#) &)=delete
- void [SetUp](#) () override
- void [Terminate](#) () override
- bool [GetMotorError](#) (uint64\_t &motor\_error) override
- bool [GetEncoderError](#) (uint32\_t &encoder\_error) override
- bool [GetControllerError](#) (uint32\_t &controller\_error) override
- bool [GetSensorlessError](#) (uint32\_t &sensorless\_error) override
- bool [GetEncoderEstimates](#) (float &pos, float &vel) override
- bool [GetEncoderCount](#) (int32\_t &shadow\_cnt, int32\_t &cnt\_cpr) override
- bool [GetIq](#) (float &iq\_setpoint, float &iq\_measured) override

- bool [GetSensorlessEstimates](#) (float &pos, float &vel) override
- bool [GetBusVoltageCurrent](#) (float &bus\_voltage, float &bus\_current) override
- bool [GetAdcVoltage](#) (float &adc\_voltage) override
- bool [SetAxisNodeId](#) (uint32\_t axis\_id) override
- bool [SetAxisRequestedState](#) (uint32\_t state) override
- bool [SetAxisStartupConfig](#) () override
- bool [SetInputPos](#) (float input\_pos, int16\_t vel\_ff, int16\_t torque\_ff) override
- bool [SetInputVel](#) (float input\_vel, float torque\_ff) override
- bool [SetInputTorque](#) (float input\_torque) override
- bool [SetControllerModes](#) (int32\_t control\_mode, int32\_t input\_mode) override
- bool [SetLimits](#) (float velocity\_limit, float current\_limit) override
- bool [SetTrajVelLimit](#) (float traj\_vel\_limit) override
- bool [SetTrajAccelLimits](#) (float traj\_accel\_limit, float traj\_decel\_limit) override
- bool [SetTrajInertia](#) (float traj\_inertia) override
- bool [SetLinearCount](#) (int32\_t position) override
- bool [SetPosGain](#) (float pos\_gain) override
- bool [SetVelGains](#) (float vel\_gain, float vel\_interator\_gain) override
- bool [Nmt](#) () override
- bool [Estop](#) () override
- bool [ClearErrors](#) () override
- bool [StartAnticogging](#) () override

## Static Public Member Functions

- static constexpr uint32\_t [GetNodeId](#) (uint32\_t msgID)
- static constexpr uint8\_t [GetCmdId](#) (uint32\_t msgID)

## Static Public Attributes

- static constexpr uint8\_t [NUM\\_NODE\\_ID\\_BITS](#) = 6
- static constexpr uint8\_t [NUM\\_CMD\\_ID\\_BITS](#) = 11 - NUM\_NODE\_ID\_BITS

## Additional Inherited Members

### 8.44.1 Constructor & Destructor Documentation

#### 8.44.1.1 ODriveCAN()

```
huron::odrive::ODriveCAN::ODriveCAN (
    huron::driver::can::BusBase * canbus,
    uint32_t axis_id,
    std::unique_ptr< ODriveConfiguration > config,
    uint32_t get_timeout = kGetTimeout )
```

Constructor of [ODriveCAN](#). As the CAN interface of [ODrive](#) v3.6 does not allow reading configuration from hardware, a default configuration matrix must be passed to the constructor.

#### Precondition

The configuration is the same as on hardware component.

### 8.44.2 Member Function Documentation

#### 8.44.2.1 ClearErrors()

bool huron::odrive::ODriveCAN::ClearErrors ( ) [override], [virtual]  
Implements [huron::odrive::ODrive](#).

#### 8.44.2.2 Estop()

bool huron::odrive::ODriveCAN::Estop ( ) [override], [virtual]  
Implements [huron::odrive::ODrive](#).

#### 8.44.2.3 GetAdcVoltage()

bool huron::odrive::ODriveCAN::GetAdcVoltage (   
float & adc\_voltage ) [override], [virtual]  
Implements [huron::odrive::ODrive](#).

#### 8.44.2.4 GetBusVoltageCurrent()

bool huron::odrive::ODriveCAN::GetBusVoltageCurrent (   
float & bus\_voltage,   
float & bus\_current ) [override], [virtual]  
Implements [huron::odrive::ODrive](#).

#### 8.44.2.5 GetControllerError()

bool huron::odrive::ODriveCAN::GetControllerError (   
uint32\_t & controller\_error ) [override], [virtual]  
Implements [huron::odrive::ODrive](#).

#### 8.44.2.6 GetEncoderCount()

bool huron::odrive::ODriveCAN::GetEncoderCount (   
int32\_t & shadow\_cnt,   
int32\_t & cnt\_cpr ) [override], [virtual]  
Implements [huron::odrive::ODrive](#).

#### 8.44.2.7 GetEncoderError()

bool huron::odrive::ODriveCAN::GetEncoderError (   
uint32\_t & encoder\_error ) [override], [virtual]  
Implements [huron::odrive::ODrive](#).

#### 8.44.2.8 GetEncoderEstimates()

bool huron::odrive::ODriveCAN::GetEncoderEstimates (   
float & pos,   
float & vel ) [override], [virtual]  
Implements [huron::odrive::ODrive](#).

#### 8.44.2.9 GetIq()

```
bool huron::odrive::ODriveCAN::GetIq (
    float & iq_setpoint,
    float & iq_measured ) [override], [virtual]
```

Implements [huron::odrive::ODrive](#).

#### 8.44.2.10 GetMotorError()

```
bool huron::odrive::ODriveCAN::GetMotorError (
    uint64_t & motor_error ) [override], [virtual]
```

Implements [huron::odrive::ODrive](#).

#### 8.44.2.11 GetSensorlessError()

```
bool huron::odrive::ODriveCAN::GetSensorlessError (
    uint32_t & sensorless_error ) [override], [virtual]
```

Implements [huron::odrive::ODrive](#).

#### 8.44.2.12 GetSensorlessEstimates()

```
bool huron::odrive::ODriveCAN::GetSensorlessEstimates (
    float & pos,
    float & vel ) [override], [virtual]
```

Implements [huron::odrive::ODrive](#).

#### 8.44.2.13 Nmt()

```
bool huron::odrive::ODriveCAN::Nmt ( ) [override], [virtual]
```

Implements [huron::odrive::ODrive](#).

#### 8.44.2.14 SetAxisNodeid()

```
bool huron::odrive::ODriveCAN::SetAxisNodeid (
    uint32_t axis_id ) [override], [virtual]
```

Implements [huron::odrive::ODrive](#).

#### 8.44.2.15 SetAxisRequestedState()

```
bool huron::odrive::ODriveCAN::SetAxisRequestedState (
    uint32_t state ) [override], [virtual]
```

Implements [huron::odrive::ODrive](#).

#### 8.44.2.16 SetAxisStartupConfig()

```
bool huron::odrive::ODriveCAN::SetAxisStartupConfig ( ) [override], [virtual]
```

Implements [huron::odrive::ODrive](#).

#### 8.44.2.17 SetControllerModes()

```
bool huron::odrive::ODriveCAN::SetControllerModes (
    int32_t control_mode,
    int32_t input_mode ) [override], [virtual]
```

Implements [huron::odrive::ODrive](#).

#### 8.44.2.18 SetInputPos()

```
bool huron::odrive::ODriveCAN::SetInputPos (
    float input_pos,
    int16_t vel_ff,
    int16_t torque_ff ) [override], [virtual]
```

Implements [huron::odrive::ODrive](#).

#### 8.44.2.19 SetInputTorque()

```
bool huron::odrive::ODriveCAN::SetInputTorque (
    float input_torque ) [override], [virtual]
```

Implements [huron::odrive::ODrive](#).

#### 8.44.2.20 SetInputVel()

```
bool huron::odrive::ODriveCAN::SetInputVel (
    float input_vel,
    float torque_ff ) [override], [virtual]
```

Implements [huron::odrive::ODrive](#).

#### 8.44.2.21 SetLimits()

```
bool huron::odrive::ODriveCAN::SetLimits (
    float velocity_limit,
    float current_limit ) [override], [virtual]
```

Implements [huron::odrive::ODrive](#).

#### 8.44.2.22 SetLinearCount()

```
bool huron::odrive::ODriveCAN::SetLinearCount (
    int32_t position ) [override], [virtual]
```

Implements [huron::odrive::ODrive](#).

#### 8.44.2.23 SetPosGain()

```
bool huron::odrive::ODriveCAN::SetPosGain (
    float pos_gain ) [override], [virtual]
```

Implements [huron::odrive::ODrive](#).

#### 8.44.2.24 SetTrajAccelLimits()

```
bool huron::odrive::ODriveCAN::SetTrajAccelLimits (
    float traj_accel_limit,
    float traj_decel_limit ) [override], [virtual]
```

Implements [huron::odrive::ODrive](#).

**8.44.2.25 SetTrajInertia()**

```
bool huron::odrive::ODriveCAN::SetTrajInertia (
    float traj_inertia ) [override], [virtual]
```

Implements [huron::odrive::ODrive](#).

**8.44.2.26 SetTrajVelLimit()**

```
bool huron::odrive::ODriveCAN::SetTrajVelLimit (
    float traj_vel_limit ) [override], [virtual]
```

Implements [huron::odrive::ODrive](#).

**8.44.2.27 Setup()**

```
void huron::odrive::ODriveCAN::Setup ( ) [override], [virtual]
```

Implements [huron::GenericComponent](#).

**8.44.2.28 SetVelGains()**

```
bool huron::odrive::ODriveCAN::SetVelGains (
    float vel_gain,
    float vel_interator_gain ) [override], [virtual]
```

Implements [huron::odrive::ODrive](#).

**8.44.2.29 StartAnticogging()**

```
bool huron::odrive::ODriveCAN::StartAnticogging ( ) [override], [virtual]
```

Implements [huron::odrive::ODrive](#).

**8.44.2.30 Terminate()**

```
void huron::odrive::ODriveCAN::Terminate ( ) [override], [virtual]
```

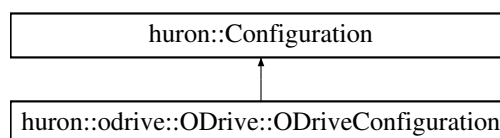
Implements [huron::GenericComponent](#).

The documentation for this class was generated from the following files:

- /github/workspace/system/odrive/include/huron/odrive/odrive\_can.h
- /github/workspace/system/odrive/src/odrive\_can.cc

**8.45 huron::odrive::ODrive::ODriveConfiguration Class Reference**

Inheritance diagram for huron::odrive::ODrive::ODriveConfiguration:





## Public Member Functions

- [ODriveConfiguration](#) (ConfigMap config\_map, std::set< std::string > valid\_keys)
- [ODriveConfiguration](#) (ConfigMap config\_map)
- [ODriveConfiguration](#) ()

## Additional Inherited Members

### 8.45.1 Constructor & Destructor Documentation

#### 8.45.1.1 ODriveConfiguration() [1/2]

```
huron::odrive::ODrive::ODriveConfiguration::ODriveConfiguration (
    ConfigMap config_map,
    std::set< std::string > valid_keys ) [inline]
```

Supports further inheritance.

#### 8.45.1.2 ODriveConfiguration() [2/2]

```
huron::odrive::ODrive::ODriveConfiguration::ODriveConfiguration ( ) [inline]
```

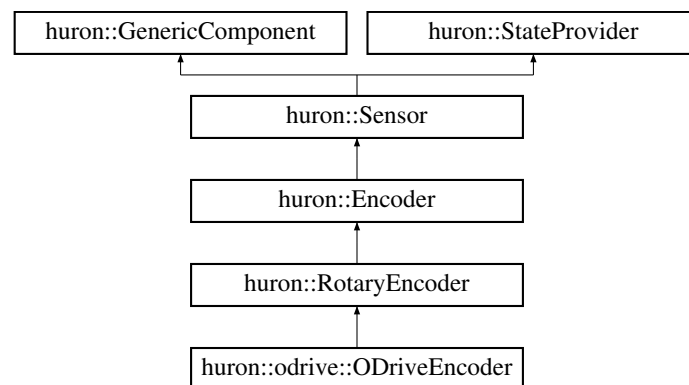
Default constructor of [ODriveConfiguration](#). This constructor is not recommended as for some protocols like CAN, [ODrive](#) cannot read the config values from hardware. The recommended way is to initialize the local config map with initial values.

The documentation for this class was generated from the following file:

- /github/workspace/system/odrive/include/huron/odrive/odrive.h

## 8.46 huron::odrive::ODriveEncoder Class Reference

Inheritance diagram for huron::odrive::ODriveEncoder:



## Public Member Functions

- **ODriveEncoder** (double gear\_ratio, std::unique\_ptr< [RotaryEncoderConfiguration](#) > config, std::shared\_ptr< [ODrive](#) > odrive)
- **ODriveEncoder** (double gear\_ratio, double cpr, std::shared\_ptr< [ODrive](#) > odrive)
- **ODriveEncoder** (double cpr, std::shared\_ptr< [ODrive](#) > odrive)
- **ODriveEncoder** (const [ODriveEncoder](#) &)=delete
- [ODriveEncoder](#) & **operator=** (const [ODriveEncoder](#) &)=delete
- void [Initialize](#) () override
- void [SetUp](#) () override
- void [Terminate](#) () override

## Protected Member Functions

- void [DoUpdateState](#) () override

## Additional Inherited Members

### 8.46.1 Member Function Documentation

#### 8.46.1.1 DoUpdateState()

```
void huron::odrive::ODriveEncoder::DoUpdateState ( ) [override], [protected], [virtual]
```

Classes derived from [RotaryEncoder](#) should override this function instead of directly overriding RequestUpdateState(). [RotaryEncoder](#) already handled the internal count update for convenience.

This function should update the current count (count\_) and, if possible, velocity (velocity\_).

Implements [huron::RotaryEncoder](#).

#### 8.46.1.2 Initialize()

```
void huron::odrive::ODriveEncoder::Initialize ( ) [override], [virtual]
```

Implements [huron::GenericComponent](#).

#### 8.46.1.3 Setup()

```
void huron::odrive::ODriveEncoder::Setup ( ) [override], [virtual]
```

Implements [huron::GenericComponent](#).

#### 8.46.1.4 Terminate()

```
void huron::odrive::ODriveEncoder::Terminate ( ) [override], [virtual]
```

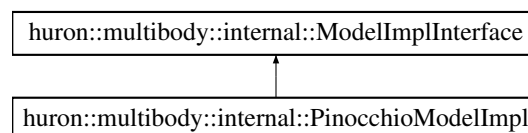
Implements [huron::GenericComponent](#).

The documentation for this class was generated from the following files:

- /github/workspace/system/odrive/include/huron/odrive/odrive\_rotary\_encoder.h
- /github/workspace/system/odrive/src/odrive\_rotary\_encoder.cc

## 8.47 huron::multibody::internal::PinocchioModelImpl Class Reference

Inheritance diagram for huron::multibody::internal::PinocchioModelImpl:



## Classes

- struct [Impl](#)

## Public Member Functions

- **PinocchioModelImpl** (const [PinocchioModelImpl](#) &)=delete
- [PinocchioModelImpl](#) & **operator=** (const [PinocchioModelImpl](#) &)=delete

- void [BuildFromUrdf](#) (const std::string &urdf\_path, JointType root\_joint\_type) override
- const std::vector< std::string > & [GetJointNames](#) () const override
- std::weak\_ptr< [Joint](#) > [GetJoint](#) (const std::string &name) const override
- std::weak\_ptr< [Joint](#) > [GetJoint](#) (size\_t joint\_index) const override
- JointType [GetJointType](#) (size\_t joint\_index) const override
- JointIndex [GetJointIndex](#) (const std::string &joint\_name) const override
- std::unique\_ptr< [JointDescription](#) > [GetJointDescription](#) (JointIndex joint\_index) const override
- std::unique\_ptr< [JointDescription](#) > [GetJointDescription](#) (const std::string &joint\_name) const override
- Eigen::Affine3d [GetJointTransformInWorld](#) (size\_t joint\_index) const override
- FrameIndex [GetFrameIndex](#) (const std::string &frame\_name) const override
- const std::string & [GetFrameName](#) (FrameIndex frame\_index) const override
- FrameType [GetFrameType](#) (FrameIndex frame\_index) const override
- Eigen::Affine3d [GetFrameTransform](#) (FrameIndex from\_frame, FrameIndex to\_frame) const override
- Eigen::Affine3d [GetFrameTransformInWorld](#) (FrameIndex frame) const override
- Eigen::Vector3d [EvalCenterOfMassPosition](#) () override
- Eigen::Vector3d [GetCenterOfMassPosition](#) () const override
- Eigen::VectorXd [NeutralConfiguration](#) () const override
- const Eigen::VectorXd & [GetAccelerations](#) () const override  
*Get the generalized accelerations of the model.*
- const Eigen::VectorXd & [GetTorques](#) () const override  
*Get the joint torques.*
- const Eigen::MatrixXd & [GetMassMatrix](#) () const override  
*Get the mass matrix with the cached value.*
- const Eigen::MatrixXd & [GetCoriolisMatrix](#) () const override  
*Get the Coriolis matrix with the cached value.*
- const Eigen::VectorXd & [GetNonlinearEffects](#) () const override  
*Get the nonlinear effects vector.*
- const Eigen::VectorXd & [GetGravity](#) () const override  
*Get the gravity vector.*
- const huron::Vector6d & [GetSpatialMomentum](#) () const override  
*Get the spatial momentum with respect to the specified frame.*
- huron::Vector6d [GetCentroidalMomentum](#) () const override  
*Get the centroidal momentum.*
- const huron::Matrix6Xd & [GetCentroidalMatrix](#) () const override  
*Get the centroidal momentum matrix with the cached value.*
- void [ComputeAll](#) (const Eigen::Ref< const Eigen::VectorXd > &q, const Eigen::Ref< const Eigen::VectorXd > &v) override
- void [ForwardKinematics](#) (const Eigen::Ref< const Eigen::VectorXd > &q) override
- void [ForwardKinematics](#) (const Eigen::Ref< const Eigen::VectorXd > &q, const Eigen::Ref< const Eigen::VectorXd > &v) override
- void [ForwardKinematics](#) (const Eigen::Ref< const Eigen::VectorXd > &q, const Eigen::Ref< const Eigen::VectorXd > &v, const Eigen::Ref< const Eigen::VectorXd > &a) override
- bool [is\\_built](#) () const override
- size\_t [num\\_positions](#) () const override
- size\_t [num\\_velocities](#) () const override
- size\_t [num\\_joints](#) () const override
- size\_t [num\\_frames](#) () const override

## Static Public Member Functions

- static bool [IsAvailable](#) ()

## 8.47.1 Member Function Documentation

### 8.47.1.1 BuildFromUrdf()

```
void huron::multibody::internal::PinocchioModelImpl::BuildFromUrdf (
    const std::string & urdf_path,
    JointType root_joint_type ) [override], [virtual]
```

Reimplemented from [huron::multibody::internal::ModellImplInterface](#).

### 8.47.1.2 ComputeAll()

```
void huron::multibody::internal::PinocchioModelImpl::ComputeAll (
    const Eigen::Ref< const Eigen::VectorXd > & q,
    const Eigen::Ref< const Eigen::VectorXd > & v ) [override], [virtual]
```

Reimplemented from [huron::multibody::internal::ModellImplInterface](#).

### 8.47.1.3 EvalCenterOfMassPosition()

```
Eigen::Vector3d huron::multibody::internal::PinocchioModelImpl::EvalCenterOfMassPosition ( )
[override], [virtual]
```

Reimplemented from [huron::multibody::internal::ModellImplInterface](#).

### 8.47.1.4 ForwardKinematics() [1/3]

```
void huron::multibody::internal::PinocchioModelImpl::ForwardKinematics (
    const Eigen::Ref< const Eigen::VectorXd > & q ) [override], [virtual]
```

Reimplemented from [huron::multibody::internal::ModellImplInterface](#).

### 8.47.1.5 ForwardKinematics() [2/3]

```
void huron::multibody::internal::PinocchioModelImpl::ForwardKinematics (
    const Eigen::Ref< const Eigen::VectorXd > & q,
    const Eigen::Ref< const Eigen::VectorXd > & v ) [override], [virtual]
```

Reimplemented from [huron::multibody::internal::ModellImplInterface](#).

### 8.47.1.6 ForwardKinematics() [3/3]

```
void huron::multibody::internal::PinocchioModelImpl::ForwardKinematics (
    const Eigen::Ref< const Eigen::VectorXd > & q,
    const Eigen::Ref< const Eigen::VectorXd > & v,
    const Eigen::Ref< const Eigen::VectorXd > & a ) [override], [virtual]
```

Reimplemented from [huron::multibody::internal::ModellImplInterface](#).

### 8.47.1.7 GetAccelerations()

```
const Eigen::VectorXd & huron::multibody::internal::PinocchioModelImpl::GetAccelerations ( )
const [override], [virtual]
```

Get the generalized accelerations of the model.

Reimplemented from [huron::multibody::internal::ModellImplInterface](#).

**8.47.1.8 GetCenterOfMassPosition()**

```
Eigen::Vector3d huron::multibody::internal::PinocchioModelImpl::GetCenterOfMassPosition ( )
const [override], [virtual]
```

Reimplemented from [huron::multibody::internal::ModelImplInterface](#).

**8.47.1.9 GetCentroidalMatrix()**

```
const huron::Matrix6Xd & huron::multibody::internal::PinocchioModelImpl::GetCentroidalMatrix (
) const [override], [virtual]
```

Get the centroidal momentum matrix with the cached value.

Reimplemented from [huron::multibody::internal::ModelImplInterface](#).

**8.47.1.10 GetCentroidalMomentum()**

```
huron::Vector6d huron::multibody::internal::PinocchioModelImpl::GetCentroidalMomentum ( )
const [override], [virtual]
```

Get the centroidal momentum.

Reimplemented from [huron::multibody::internal::ModelImplInterface](#).

**8.47.1.11 GetCoriolisMatrix()**

```
const Eigen::MatrixX<T> & huron::multibody::internal::PinocchioModelImpl::GetCoriolisMatrix ( )
const [override], [virtual]
```

Get the Coriolis matrix with the cached value.

Reimplemented from [huron::multibody::internal::ModelImplInterface](#).

**8.47.1.12 GetFrameIndex()**

```
FrameIndex huron::multibody::internal::PinocchioModelImpl::GetFrameIndex (
    const std::string & frame_name ) const [override], [virtual]
```

Reimplemented from [huron::multibody::internal::ModelImplInterface](#).

**8.47.1.13 GetFrameName()**

```
const std::string & huron::multibody::internal::PinocchioModelImpl::GetFrameName (
    FrameIndex frame_index ) const [override], [virtual]
```

Reimplemented from [huron::multibody::internal::ModelImplInterface](#).

**8.47.1.14 GetFrameTransform()**

```
Eigen::Affine3d huron::multibody::internal::PinocchioModelImpl::GetFrameTransform (
    FrameIndex from_frame,
    FrameIndex to_frame ) const [override], [virtual]
```

Reimplemented from [huron::multibody::internal::ModelImplInterface](#).

**8.47.1.15 GetFrameTransformInWorld()**

```
Eigen::Affine3d huron::multibody::internal::PinocchioModelImpl::GetFrameTransformInWorld (
    FrameIndex frame ) const [override], [virtual]
```

Reimplemented from [huron::multibody::internal::ModelImplInterface](#).

**8.47.1.16 GetFrameType()**

```
FrameType huron::multibody::internal::PinocchioModelImpl::GetFrameType (
    FrameIndex frame_index ) const [override], [virtual]
```

Reimplemented from [huron::multibody::internal::ModellImplInterface](#).

**8.47.1.17 GetGravity()**

```
const Eigen::VectorXd & huron::multibody::internal::PinocchioModelImpl::GetGravity ( ) const
[override], [virtual]
```

Get the gravity vector.

Reimplemented from [huron::multibody::internal::ModellImplInterface](#).

**8.47.1.18 GetJoint() [1/2]**

```
std::weak_ptr< Joint > huron::multibody::internal::PinocchioModelImpl::GetJoint (
    const std::string & name ) const [override], [virtual]
```

Reimplemented from [huron::multibody::internal::ModellImplInterface](#).

**8.47.1.19 GetJoint() [2/2]**

```
std::weak_ptr< Joint > huron::multibody::internal::PinocchioModelImpl::GetJoint (
    size_t joint_index ) const [override], [virtual]
```

Reimplemented from [huron::multibody::internal::ModellImplInterface](#).

**8.47.1.20 GetJointDescription() [1/2]**

```
std::unique_ptr< JointDescription > huron::multibody::internal::PinocchioModelImpl::GetJoint↵
Description (
    const std::string & joint_name ) const [override], [virtual]
```

Reimplemented from [huron::multibody::internal::ModellImplInterface](#).

**8.47.1.21 GetJointDescription() [2/2]**

```
std::unique_ptr< JointDescription > huron::multibody::internal::PinocchioModelImpl::GetJoint↵
Description (
    JointIndex joint_index ) const [override], [virtual]
```

Reimplemented from [huron::multibody::internal::ModellImplInterface](#).

**8.47.1.22 GetJointIndex()**

```
JointIndex huron::multibody::internal::PinocchioModelImpl::GetJointIndex (
    const std::string & joint_name ) const [override], [virtual]
```

Implements [huron::multibody::internal::ModellImplInterface](#).

**8.47.1.23 GetJointNames()**

```
const std::vector< std::string > & huron::multibody::internal::PinocchioModelImpl::GetJoint↵
Names ( ) const [override], [virtual]
```

Reimplemented from [huron::multibody::internal::ModellImplInterface](#).

**8.47.1.24 GetJointTransformInWorld()**

```
Eigen::Affine3d huron::multibody::internal::PinocchioModelImpl::GetJointTransformInWorld (
    size_t joint_index ) const [override], [virtual]
```

Reimplemented from [huron::multibody::internal::ModelImplInterface](#).

**8.47.1.25 GetJointType()**

```
JointType huron::multibody::internal::PinocchioModelImpl::GetJointType (
    size_t joint_index ) const [override], [virtual]
```

Reimplemented from [huron::multibody::internal::ModelImplInterface](#).

**8.47.1.26 GetMassMatrix()**

```
const Eigen::MatrixXd & huron::multibody::internal::PinocchioModelImpl::GetMassMatrix ( )
const [override], [virtual]
```

Get the mass matrix with the cached value.

Reimplemented from [huron::multibody::internal::ModelImplInterface](#).

**8.47.1.27 GetNonlinearEffects()**

```
const Eigen::VectorXd & huron::multibody::internal::PinocchioModelImpl::GetNonlinearEffects (
) const [override], [virtual]
```

Get the nonlinear effects vector.

Reimplemented from [huron::multibody::internal::ModelImplInterface](#).

**8.47.1.28 GetSpatialMomentum()**

```
const huron::Vector6d & huron::multibody::internal::PinocchioModelImpl::GetSpatialMomentum ( )
const [override], [virtual]
```

Get the spatial momentum with respect to the specified frame.

Reimplemented from [huron::multibody::internal::ModelImplInterface](#).

**8.47.1.29 GetTorques()**

```
const Eigen::VectorXd & huron::multibody::internal::PinocchioModelImpl::GetTorques ( ) const
[override], [virtual]
```

Get the joint torques.

Reimplemented from [huron::multibody::internal::ModelImplInterface](#).

**8.47.1.30 is\_built()**

```
bool huron::multibody::internal::PinocchioModelImpl::is_built ( ) const [inline], [override],
[virtual]
```

Reimplemented from [huron::multibody::internal::ModelImplInterface](#).

**8.47.1.31 NeutralConfiguration()**

```
Eigen::VectorXd huron::multibody::internal::PinocchioModelImpl::NeutralConfiguration ( ) const
[override], [virtual]
```

Reimplemented from [huron::multibody::internal::ModelImplInterface](#).

### 8.47.1.32 num\_frames()

```
size_t huron::multibody::internal::PinocchioModelImpl::num_frames ( ) const [inline], [override], [virtual]
```

Reimplemented from [huron::multibody::internal::ModelImplInterface](#).

### 8.47.1.33 num\_joints()

```
size_t huron::multibody::internal::PinocchioModelImpl::num_joints ( ) const [inline], [override], [virtual]
```

Reimplemented from [huron::multibody::internal::ModelImplInterface](#).

### 8.47.1.34 num\_positions()

```
size_t huron::multibody::internal::PinocchioModelImpl::num_positions ( ) const [inline], [override], [virtual]
```

Reimplemented from [huron::multibody::internal::ModelImplInterface](#).

### 8.47.1.35 num\_velocities()

```
size_t huron::multibody::internal::PinocchioModelImpl::num_velocities ( ) const [inline], [override], [virtual]
```

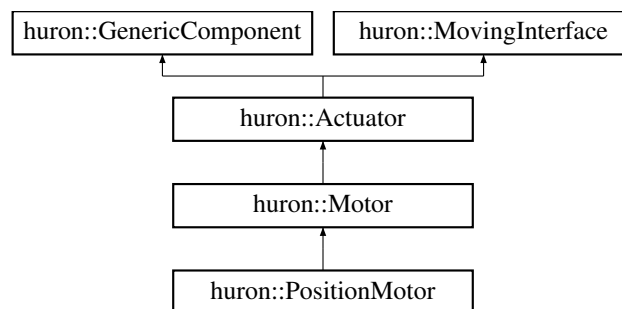
Reimplemented from [huron::multibody::internal::ModelImplInterface](#).

The documentation for this class was generated from the following files:

- /github/workspace/multibody/include/huron/multibody/pinocchio\_model\_impl.h
- /github/workspace/multibody/src/no\_pinocchio\_model\_impl.cc
- /github/workspace/multibody/src/pinocchio\_model\_impl.cc

## 8.48 huron::PositionMotor Class Reference

Inheritance diagram for huron::PositionMotor:



### Public Member Functions

- **PositionMotor** (std::unique\_ptr< [PositionMotorConfiguration](#) > config, double gear\_ratio)
- **PositionMotor** (double gear\_ratio)
- **PositionMotor** (const [PositionMotor](#) &)=delete
- [PositionMotor](#) & **operator=** (const [PositionMotor](#) &)=delete

### Additional Inherited Members

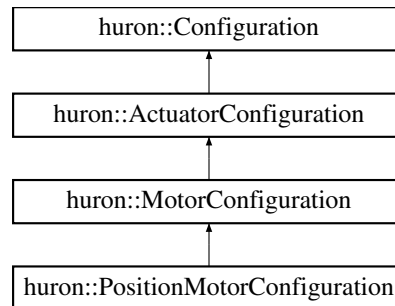
The documentation for this class was generated from the following file:

- /github/workspace/system/control\_interfaces/include/huron/control\_interfaces/position\_motor.h



## 8.49 huron::PositionMotorConfiguration Class Reference

Inheritance diagram for huron::PositionMotorConfiguration:



### Public Member Functions

- [PositionMotorConfiguration](#) (ConfigMap config\_map, std::set< std::string > valid\_keys)

### Additional Inherited Members

#### 8.49.1 Constructor & Destructor Documentation

##### 8.49.1.1 PositionMotorConfiguration()

```

huron::PositionMotorConfiguration::PositionMotorConfiguration (
    ConfigMap config_map,
    std::set< std::string > valid_keys ) [inline]

```

Supports further inheritance.

The documentation for this class was generated from the following file:

- /github/workspace/system/control\_interfaces/include/huron/control\_interfaces/position\_motor.h

## 8.50 PushRecoveryControl Class Reference

### Public Member Functions

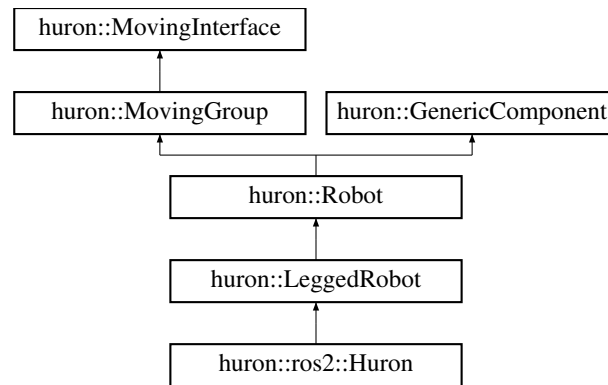
- Eigen::MatrixXd **GetTorque** (const Eigen::Vector2d &cop, const Eigen::VectorXd &position, const Eigen::↔ VectorXd &velocity)

The documentation for this class was generated from the following files:

- /github/workspace/control/include/huron/control/push\_recovery.h
- /github/workspace/control/src/push\_recovery.cc

## 8.51 huron::Robot Class Reference

Inheritance diagram for huron::Robot:



## Public Member Functions

- **Robot** (std::unique\_ptr< [RobotConfiguration](#) > config)
- **Robot** (const [Robot](#) &)=delete
- [Robot](#) & **operator=** (const [Robot](#) &)=delete
- [Model](#) \*const **GetModel** ()
- void **RegisterStateProvider** (std::shared\_ptr< [StateProvider](#) > state\_provider, bool is\_joint\_state\_provider=false)
- void **UpdateAllStates** ()
- void **UpdateJointStates** ()
- const Eigen::VectorBlock< const Eigen::VectorXd > **GetJointPositions** () const
- const Eigen::VectorBlock< const Eigen::VectorXd > **GetJointVelocities** () const

## Protected Member Functions

- **Robot** (std::unique\_ptr< [RobotConfiguration](#) > config, std::shared\_ptr< [Model](#) > model)
- **Robot** (std::shared\_ptr< [Model](#) > model)

## Protected Attributes

- std::shared\_ptr< [Model](#) > **model\_**
- std::vector< std::shared\_ptr< [StateProvider](#) > > **non\_joint\_state\_providers\_**
- std::vector< std::shared\_ptr< [StateProvider](#) > > **joint\_state\_providers\_**

## 8.51.1 Member Function Documentation

### 8.51.1.1 UpdateAllStates()

```
void huron::Robot::UpdateAllStates ( )
```

Calls RequestStateUpdate() on all the registered state providers.

### 8.51.1.2 UpdateJointStates()

```
void huron::Robot::UpdateJointStates ( )
```

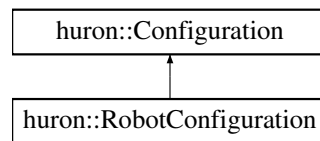
Calls RequestStateUpdate() on all the registered state providers for joints.

The documentation for this class was generated from the following files:

- /github/workspace/system/control\_interfaces/include/huron/control\_interfaces/robot.h
- /github/workspace/system/control\_interfaces/src/robot.cc

## 8.52 huron::RobotConfiguration Class Reference

Inheritance diagram for huron::RobotConfiguration:



### Public Member Functions

- **RobotConfiguration** (ConfigMap config\_map, std::set< std::string > valid\_keys)
- **RobotConfiguration** (ConfigMap config\_map)

### Additional Inherited Members

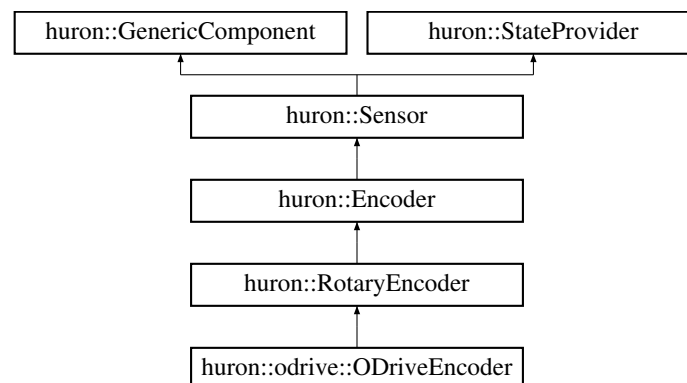
The documentation for this class was generated from the following file:

- /github/workspace/system/control\_interfaces/include/huron/control\_interfaces/robot.h

## 8.53 huron::RotaryEncoder Class Reference

```
#include <rotary_encoder.h>
```

Inheritance diagram for huron::RotaryEncoder:



### Public Member Functions

- **RotaryEncoder** (double gear\_ratio, std::unique\_ptr< [RotaryEncoderConfiguration](#) > config)
- **RotaryEncoder** (double gear\_ratio, double cpr)
- **RotaryEncoder** (double cpr)
- **RotaryEncoder** (const [RotaryEncoder](#) &)=delete
- [RotaryEncoder](#) & **operator=** (const [RotaryEncoder](#) &)=delete
- void [RequestStateUpdate](#) () final
- double [GetCount](#) () const
- double [GetVelocityCount](#) () const
- double [GetPrevCount](#) () const
- double [GetCPR](#) () const
- double [GetPosition](#) () const override
- double [GetAngleDegree](#) () const
- double [GetVelocity](#) () const override
- double [GetVelocityDegree](#) () const
- void [Reset](#) () override

## Protected Member Functions

- virtual void [DoUpdateState](#) ()=0

## Protected Attributes

- double **velocity\_** = 0.0  
*[Encoder](#) velocity in counts per second.*
- double **prev\_velocity\_** = 0.0  
*[Encoder](#) previous velocity in counts per second.*
- double **count\_** = 0.0
- double **prev\_count\_** = 0.0
- double **cpr\_**

### 8.53.1 Detailed Description

Abstract class for using an encoder.

### 8.53.2 Member Function Documentation

#### 8.53.2.1 DoUpdateState()

```
virtual void huron::RotaryEncoder::DoUpdateState ( ) [protected], [pure virtual]
```

Classes derived from [RotaryEncoder](#) should override this function instead of directly overriding [RequestUpdateState\(\)](#). [RotaryEncoder](#) already handled the internal count update for convenience.

This function should update the current count (**count\_**) and, if possible, velocity (**velocity\_**).

Implemented in [huron::odrive::ODriveEncoder](#).

#### 8.53.2.2 GetAngleDegree()

```
double huron::RotaryEncoder::GetAngleDegree ( ) const [inline]
```

Gets the current angle in degrees. This takes into account the gear ratio and CPR.

#### 8.53.2.3 GetCount()

```
double huron::RotaryEncoder::GetCount ( ) const [inline]
```

Gets the current encoder count.

#### 8.53.2.4 GetCPR()

```
double huron::RotaryEncoder::GetCPR ( ) const [inline]
```

Gets the counts per revolution (CPR).

#### 8.53.2.5 GetPosition()

```
double huron::RotaryEncoder::GetPosition ( ) const [inline], [override], [virtual]
```

Gets the current angle in radians. This takes into account the gear ratio and CPR.

Implements [huron::Encoder](#).

#### 8.53.2.6 GetPrevCount()

```
double huron::RotaryEncoder::GetPrevCount ( ) const [inline]
```

Gets the previous encoder count.

**8.53.2.7 GetVelocity()**

```
double huron::RotaryEncoder::GetVelocity ( ) const [inline], [override], [virtual]
```

Gets the current velocity in radians/second. This takes into account the gear ratio and CPR.

Implements [huron::Encoder](#).

**8.53.2.8 GetVelocityCount()**

```
double huron::RotaryEncoder::GetVelocityCount ( ) const [inline]
```

Gets the current encoder velocity in count.

**8.53.2.9 GetVelocityDegree()**

```
double huron::RotaryEncoder::GetVelocityDegree ( ) const [inline]
```

Gets the current velocity in degrees/second. This takes into account the gear ratio and CPR.

**8.53.2.10 RequestStateUpdate()**

```
void huron::RotaryEncoder::RequestStateUpdate ( ) [inline], [final], [virtual]
```

Implements [huron::StateProvider](#).

**8.53.2.11 Reset()**

```
void huron::RotaryEncoder::Reset ( ) [inline], [override], [virtual]
```

Resets the encoder count.

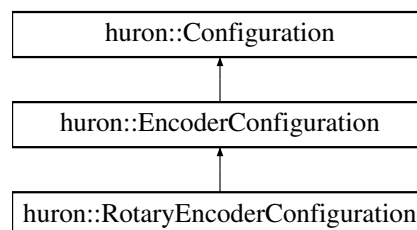
Implements [huron::Encoder](#).

The documentation for this class was generated from the following file:

- [/github/workspace/system/control\\_interfaces/include/huron/control\\_interfaces/rotary\\_encoder.h](#)

**8.54 huron::RotaryEncoderConfiguration Class Reference**

Inheritance diagram for huron::RotaryEncoderConfiguration:

**Public Member Functions**

- [RotaryEncoderConfiguration](#) (ConfigMap config\_map, std::set< std::string > valid\_keys)
- [RotaryEncoderConfiguration](#) (double cpr)

**Additional Inherited Members****8.54.1 Constructor & Destructor Documentation**

### 8.54.1.1 RotaryEncoderConfiguration()

```
huron::RotaryEncoderConfiguration::RotaryEncoderConfiguration (
    ConfigMap config_map,
    std::set< std::string > valid_keys ) [inline]
```

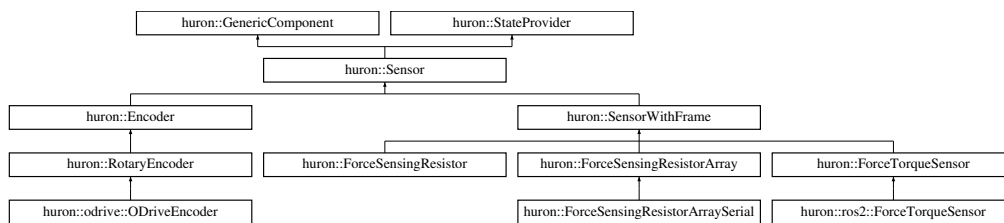
Supports further inheritance.

The documentation for this class was generated from the following file:

- /github/workspace/system/control\_interfaces/include/huron/control\_interfaces/rotary\_encoder.h

## 8.55 huron::Sensor Class Reference

Inheritance diagram for huron::Sensor:



### Public Member Functions

- **Sensor** (const Eigen::Vector2i &dim, std::unique\_ptr< [Configuration](#) > config)
- **Sensor** (const Eigen::Vector2i &dim)
- **Sensor** (int rows, int cols, std::unique\_ptr< [Configuration](#) > config)
- **Sensor** (int rows, int cols)
- **Sensor** (const [Sensor](#) &)=delete
- [Sensor](#) & **operator=** (const [Sensor](#) &)=delete
- virtual Eigen::VectorXd [GetValue](#) () const  
*Get the sensor value.*
- virtual Eigen::VectorXd **ReloadAndGetValue** ()

### Additional Inherited Members

#### 8.55.1 Member Function Documentation

##### 8.55.1.1 GetValue()

```
Eigen::VectorXd huron::Sensor::GetValue ( ) const [virtual]
```

Get the sensor value.

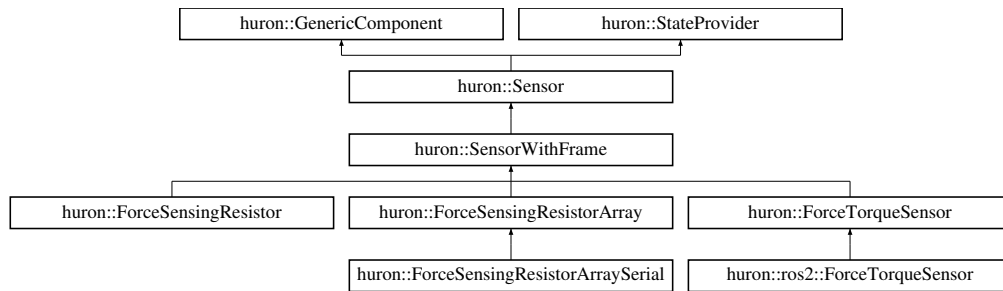
Reimplemented in [huron::ForceSensingResistorArraySerial](#), and [huron::ForceTorqueSensor](#).

The documentation for this class was generated from the following files:

- /github/workspace/system/control\_interfaces/include/huron/control\_interfaces/sensor.h
- /github/workspace/system/control\_interfaces/src/sensor.cc

## 8.56 huron::SensorWithFrame Class Reference

Inheritance diagram for huron::SensorWithFrame:



## Public Member Functions

- **SensorWithFrame** (const Eigen::Vector2i &dim, std::weak\_ptr< const [Frame](#) > frame)
- **SensorWithFrame** (const Eigen::Vector2i &dim, std::weak\_ptr< const [Frame](#) > frame, std::unique\_ptr< [Configuration](#) > config)
- **SensorWithFrame** (int rows, int cols, std::weak\_ptr< const [Frame](#) > frame)
- **SensorWithFrame** (int rows, int cols, std::weak\_ptr< const [Frame](#) > frame, std::unique\_ptr< [Configuration](#) > config)
- **SensorWithFrame** (const [SensorWithFrame](#) &)=delete
- **SensorWithFrame** & **operator=** (const [SensorWithFrame](#) &)=delete
- std::weak\_ptr< const [Frame](#) > **GetSensorFrame** () const  
*Get the sensor frame.*

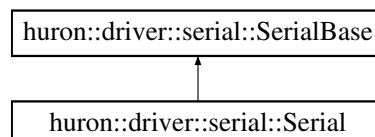
## Additional Inherited Members

The documentation for this class was generated from the following file:

- /github/workspace/system/control\_interfaces/include/huron/control\_interfaces/sensor\_with\_frame.h

## 8.57 huron::driver::serial::Serial Class Reference

Inheritance diagram for huron::driver::serial::Serial:



## Public Member Functions

- **Serial** (std::string port, uint32\_t baudrate, Parity parity, StopBits stopbits, FlowControl flowcontrol)
- **Serial** (const [Serial](#) &)=delete
- **Serial** & **operator=** (const [Serial](#) &)=delete
- void **Open** () override
- bool **IsOpen** () override
- void **Close** () override
- size\_t **Available** () override
- bool **WaitReadable** () override
- size\_t **Read** (uint8\_t \*buffer, size\_t nbytes) override
- size\_t **Read** (std::vector< uint8\_t > &buffer, size\_t nbytes=1) override
- size\_t **Read** (std::string &buffer, size\_t nbytes=1) override
- std::string **Read** (size\_t nbytes=1) override
- size\_t **ReadLine** (std::string &buffer, size\_t nbytes=65536, std::string eol="\n") override
- std::string **ReadLine** (size\_t nbytes=65536, std::string eol="\n") override

- `std::vector< std::string > ReadLines` (size\_t nbytes=65536, std::string eol="\n") override
- `size_t Write` (const uint8\_t \*data, size\_t nbytes) override
- `size_t Write` (const std::vector< uint8\_t > &data) override
- `size_t Write` (const std::string &data) override
- `void SetPort` (const std::string &port) override
- `std::string GetPort` () const override
- `void SetTimeout` (uint32\_t inter\_byte\_timeout, uint32\_t read\_timeout\_constant, uint32\_t read\_timeout\_multiplier, uint32\_t write\_timeout\_constant, uint32\_t write\_timeout\_multiplier) override
- `void SetBaudrate` (uint32\_t baudrate) override
- `uint32_t GetBaudrate` () const override
- `void SetParity` (Parity parity) override
- `Parity GetParity` () const override
- `void SetStopbits` (StopBits stopbits) override
- `StopBits GetStopbits` () const override
- `void SetFlowcontrol` (FlowControl flowcontrol) override
- `FlowControl GetFlowcontrol` () const override
- `void Flush` () override
- `void FlushInput` () override
- `void FlushOutput` () override
- `void SendBreak` (int duration) override

## Additional Inherited Members

### 8.57.1 Member Function Documentation

#### 8.57.1.1 Available()

`size_t huron::driver::serial::Serial::Available ( ) [override], [virtual]`  
 Implements [huron::driver::serial::SerialBase](#).

#### 8.57.1.2 Close()

`void huron::driver::serial::Serial::Close ( ) [override], [virtual]`  
 Implements [huron::driver::serial::SerialBase](#).

#### 8.57.1.3 Flush()

`void huron::driver::serial::Serial::Flush ( ) [override], [virtual]`  
 Implements [huron::driver::serial::SerialBase](#).

#### 8.57.1.4 FlushInput()

`void huron::driver::serial::Serial::FlushInput ( ) [override], [virtual]`  
 Implements [huron::driver::serial::SerialBase](#).

#### 8.57.1.5 FlushOutput()

`void huron::driver::serial::Serial::FlushOutput ( ) [override], [virtual]`  
 Implements [huron::driver::serial::SerialBase](#).



#### 8.57.1.6 GetBaudrate()

uint32\_t huron::driver::serial::Serial::GetBaudrate ( ) const [override], [virtual]  
Implements [huron::driver::serial::SerialBase](#).

#### 8.57.1.7 GetFlowcontrol()

FlowControl huron::driver::serial::Serial::GetFlowcontrol ( ) const [override], [virtual]  
Implements [huron::driver::serial::SerialBase](#).

#### 8.57.1.8 GetParity()

Parity huron::driver::serial::Serial::GetParity ( ) const [override], [virtual]  
Implements [huron::driver::serial::SerialBase](#).

#### 8.57.1.9 GetPort()

std::string huron::driver::serial::Serial::GetPort ( ) const [override], [virtual]  
Implements [huron::driver::serial::SerialBase](#).

#### 8.57.1.10 GetStopbits()

StopBits huron::driver::serial::Serial::GetStopbits ( ) const [override], [virtual]  
Implements [huron::driver::serial::SerialBase](#).

#### 8.57.1.11 IsOpen()

bool huron::driver::serial::Serial::IsOpen ( ) [override], [virtual]  
Implements [huron::driver::serial::SerialBase](#).

#### 8.57.1.12 Open()

void huron::driver::serial::Serial::Open ( ) [override], [virtual]  
Implements [huron::driver::serial::SerialBase](#).

#### 8.57.1.13 Read() [1/4]

std::string huron::driver::serial::Serial::Read (  
    size\_t nbytes = 1 ) [override], [virtual]  
Implements [huron::driver::serial::SerialBase](#).

#### 8.57.1.14 Read() [2/4]

size\_t huron::driver::serial::Serial::Read (  
    std::string & buffer,  
    size\_t nbytes = 1 ) [override], [virtual]  
Implements [huron::driver::serial::SerialBase](#).

**8.57.1.15 Read() [3/4]**

```
size_t huron::driver::serial::Serial::Read (
    std::vector< uint8_t > & buffer,
    size_t nbytes = 1 ) [override], [virtual]
```

Implements [huron::driver::serial::SerialBase](#).

**8.57.1.16 Read() [4/4]**

```
size_t huron::driver::serial::Serial::Read (
    uint8_t * buffer,
    size_t nbytes ) [override], [virtual]
```

Implements [huron::driver::serial::SerialBase](#).

**8.57.1.17 ReadLine() [1/2]**

```
std::string huron::driver::serial::Serial::ReadLine (
    size_t nbytes = 65536,
    std::string eol = "\n" ) [override], [virtual]
```

Implements [huron::driver::serial::SerialBase](#).

**8.57.1.18 ReadLine() [2/2]**

```
size_t huron::driver::serial::Serial::ReadLine (
    std::string & buffer,
    size_t nbytes = 65536,
    std::string eol = "\n" ) [override], [virtual]
```

Implements [huron::driver::serial::SerialBase](#).

**8.57.1.19 ReadLines()**

```
std::vector< std::string > huron::driver::serial::Serial::ReadLines (
    size_t nbytes = 65536,
    std::string eol = "\n" ) [override], [virtual]
```

Implements [huron::driver::serial::SerialBase](#).

**8.57.1.20 SendBreak()**

```
void huron::driver::serial::Serial::SendBreak (
    int duration ) [override], [virtual]
```

Implements [huron::driver::serial::SerialBase](#).

**8.57.1.21 SetBaudrate()**

```
void huron::driver::serial::Serial::SetBaudrate (
    uint32_t baudrate ) [override], [virtual]
```

Implements [huron::driver::serial::SerialBase](#).

**8.57.1.22 SetFlowcontrol()**

```
void huron::driver::serial::Serial::SetFlowcontrol (
    FlowControl flowcontrol ) [override], [virtual]
```

Implements [huron::driver::serial::SerialBase](#).

### 8.57.1.23 SetParity()

```
void huron::driver::serial::Serial::SetParity (
    Parity parity ) [override], [virtual]
```

Implements [huron::driver::serial::SerialBase](#).

### 8.57.1.24 SetPort()

```
void huron::driver::serial::Serial::SetPort (
    const std::string & port ) [override], [virtual]
```

Implements [huron::driver::serial::SerialBase](#).

### 8.57.1.25 SetStopbits()

```
void huron::driver::serial::Serial::SetStopbits (
    StopBits stopbits ) [override], [virtual]
```

Implements [huron::driver::serial::SerialBase](#).

### 8.57.1.26 SetTimeout()

```
void huron::driver::serial::Serial::SetTimeout (
    uint32_t inter_byte_timeout,
    uint32_t read_timeout_constant,
    uint32_t read_timeout_multiplier,
    uint32_t write_timeout_constant,
    uint32_t write_timeout_multiplier ) [override], [virtual]
```

Implements [huron::driver::serial::SerialBase](#).

### 8.57.1.27 WaitReadable()

```
bool huron::driver::serial::Serial::WaitReadable ( ) [override], [virtual]
```

Implements [huron::driver::serial::SerialBase](#).

### 8.57.1.28 Write() [1/3]

```
size_t huron::driver::serial::Serial::Write (
    const std::string & data ) [override], [virtual]
```

Implements [huron::driver::serial::SerialBase](#).

### 8.57.1.29 Write() [2/3]

```
size_t huron::driver::serial::Serial::Write (
    const std::vector< uint8_t > & data ) [override], [virtual]
```

Implements [huron::driver::serial::SerialBase](#).

### 8.57.1.30 Write() [3/3]

```
size_t huron::driver::serial::Serial::Write (
    const uint8_t * data,
    size_t nbytes ) [override], [virtual]
```

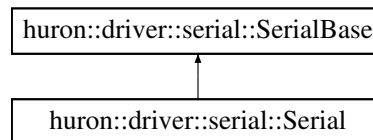
Implements [huron::driver::serial::SerialBase](#).

The documentation for this class was generated from the following files:

- /github/workspace/driver/serial/include/huron/driver/serial/wjwwood\_serial.h
- /github/workspace/driver/serial/src/wjwwood\_serial.cc

## 8.58 huron::driver::serial::SerialBase Class Reference

Inheritance diagram for huron::driver::serial::SerialBase:



### Public Member Functions

- **SerialBase** (std::string port, uint32\_t baudrate, Parity parity, StopBits stopbits, FlowControl flowcontrol)
- **SerialBase** (const [SerialBase](#) &)=delete
- [SerialBase](#) & **operator=** (const [SerialBase](#) &)=delete
- virtual void **Open** ()=0
- virtual bool **IsOpen** ()=0
- virtual void **Close** ()=0
- virtual size\_t **Available** ()=0
- virtual bool **WaitReadable** ()=0
- virtual size\_t **Read** (uint8\_t \*buffer, size\_t nbytes)=0
- virtual size\_t **Read** (std::vector< uint8\_t > &buffer, size\_t nbytes=1)=0
- virtual size\_t **Read** (std::string &buffer, size\_t nbytes=1)=0
- virtual std::string **Read** (size\_t nbytes=1)=0
- virtual size\_t **ReadLine** (std::string &buffer, size\_t nbytes=65536, std::string eol="\n")=0
- virtual std::string **ReadLine** (size\_t nbytes=65536, std::string eol="\n")=0
- virtual std::vector< std::string > **ReadLines** (size\_t nbytes=65536, std::string eol="\n")=0
- virtual size\_t **Write** (const uint8\_t \*data, size\_t nbytes)=0
- virtual size\_t **Write** (const std::vector< uint8\_t > &data)=0
- virtual size\_t **Write** (const std::string &data)=0
- virtual void **SetPort** (const std::string &port)=0
- virtual std::string **GetPort** () const =0
- virtual void **SetTimeout** (uint32\_t inter\_byte\_timeout, uint32\_t read\_timeout\_constant, uint32\_t read\_timeout\_multiplier, uint32\_t write\_timeout\_constant, uint32\_t write\_timeout\_multiplier)=0
- virtual void **SetBaudrate** (uint32\_t baudrate)=0
- virtual uint32\_t **GetBaudrate** () const =0
- virtual void **SetParity** (Parity parity)=0
- virtual Parity **GetParity** () const =0
- virtual void **SetStopbits** (StopBits stopbits)=0
- virtual StopBits **GetStopbits** () const =0
- virtual void **SetFlowcontrol** (FlowControl flowcontrol)=0
- virtual FlowControl **GetFlowcontrol** () const =0
- virtual void **Flush** ()=0
- virtual void **FlushInput** ()=0
- virtual void **FlushOutput** ()=0
- virtual void **SendBreak** (int duration)=0

## Protected Attributes

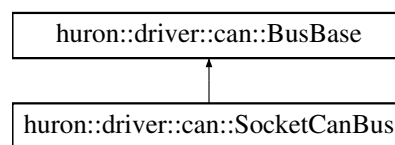
- std::string **port\_**
- uint32\_t **baudrate\_**
- Parity **parity\_**
- StopBits **stopbits\_**
- FlowControl **flowcontrol\_**

The documentation for this class was generated from the following file:

- /github/workspace/driver/serial/include/huron/driver/serial/serial.h

## 8.59 huron::driver::can::SocketCanBus Class Reference

Inheritance diagram for huron::driver::can::SocketCanBus:



## Public Member Functions

- **SocketCanBus** (std::string can\_if, uint32\_t axis\_id)
- **SocketCanBus** (const [SocketCanBus](#) &)=delete
- [SocketCanBus](#) & **operator=** (const [SocketCanBus](#) &)=delete
- bool [send\\_message](#) (const [can\\_Message\\_t](#) &message) final  
*Sends the specified CAN message.*
- bool [recv\\_message](#) ([can\\_Message\\_t](#) &message, uint32\_t timeout=UINT32\_MAX) final  
*Receives a CAN message with the same id as message.*
- bool [subscribe](#) (const [MsgIdFilterSpecs](#) &filter, on\_can\_message\_cb\_t callback, void \*ctx, [CanSubscription](#) \*\*handle) final  
*Registers a callback that will be invoked for every incoming CAN message that matches the filter.*
- bool [unsubscribe](#) ([CanSubscription](#) \*handle) final  
*Deregisters a callback that was previously registered with [subscribe\(\)](#).*

## Public Attributes

- std::string **can\_if\_**
- uint32\_t **axis\_id\_**
- sockcanpp::milliseconds **recv\_timeout\_**
- sockcanpp::CanDriver **can\_driver\_** {can\_if\_, CAN\_RAW}

## Static Public Attributes

- static constexpr sockcanpp::milliseconds **kRecvTimeout** {1000}
- static const uint8\_t **kCanFifoNone** = 0xff

## Additional Inherited Members

### 8.59.1 Member Function Documentation

### 8.59.1.1 `recv_message()`

```
bool huron::driver::can::SocketCanBus::recv_message (
    can_Message_t & message,
    uint32_t timeout = UINT32_MAX ) [final], [virtual]
```

Receives a CAN message with the same id as message.

#### Returns

: true on success or false otherwise (e.g. if the receive queue is empty).

Implements [huron::driver::can::BusBase](#).

### 8.59.1.2 `send_message()`

```
bool huron::driver::can::SocketCanBus::send_message (
    const can_Message_t & message ) [final], [virtual]
```

Sends the specified CAN message.

#### Returns

: true on success or false otherwise (e.g. if the send queue is full).

Implements [huron::driver::can::BusBase](#).

### 8.59.1.3 `subscribe()`

```
bool huron::driver::can::SocketCanBus::subscribe (
    const MsgIdFilterSpecs & filter,
    on_can_message_cb_t callback,
    void * ctx,
    CanSubscription ** handle ) [final], [virtual]
```

Registers a callback that will be invoked for every incoming CAN message that matches the filter.

#### Parameters

<i>handle</i>	On success this handle is set to an opaque pointer that can be used to cancel the subscription.
---------------	---

#### Returns

: true on success or false otherwise (e.g. if the maximum number of subscriptions has been reached).

Implements [huron::driver::can::BusBase](#).

### 8.59.1.4 `unsubscribe()`

```
bool huron::driver::can::SocketCanBus::unsubscribe (
    CanSubscription * handle ) [final], [virtual]
```

Deregisters a callback that was previously registered with [subscribe\(\)](#).

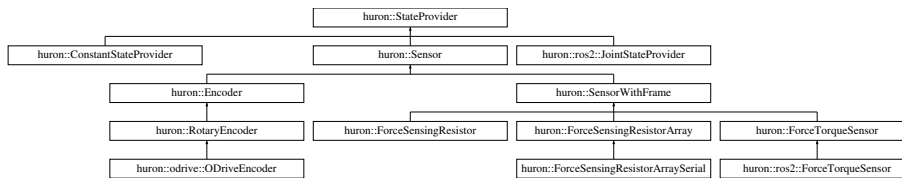
Implements [huron::driver::can::BusBase](#).

The documentation for this class was generated from the following files:

- `/github/workspace/driver/can/include/huron/driver/can/socket_can_bus.h`
- `/github/workspace/driver/can/src/socket_can_bus.cc`

## 8.60 `huron::StateProvider` Class Reference

Inheritance diagram for `huron::StateProvider`:



## Public Member Functions

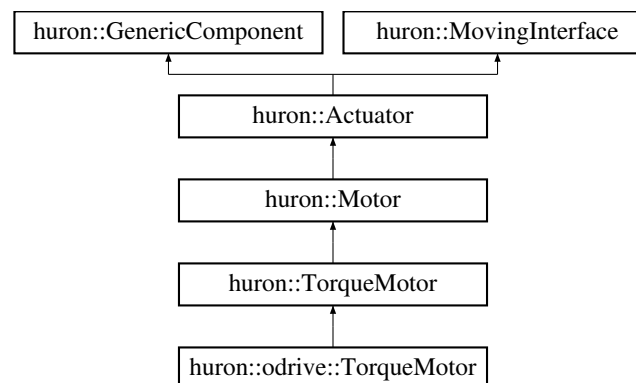
- **StateProvider** (const Eigen::Vector2i &dim)
- **StateProvider** (int rows, int cols)
- **StateProvider** (const [StateProvider](#) &)=delete
- [StateProvider](#) & **operator=** (const [StateProvider](#) &)=delete
- virtual void **RequestStateUpdate** ()=0
- virtual void **GetNewState** (Eigen::Ref< Eigen::MatrixXd > new\_state) const =0
- const Eigen::Vector2i & **dim** () const

The documentation for this class was generated from the following file:

- /github/workspace/system/control\_interfaces/include/huron/control\_interfaces/state\_provider.h

## 8.61 huron::odrive::TorqueMotor Class Reference

Inheritance diagram for huron::odrive::TorqueMotor:



## Public Member Functions

- **TorqueMotor** (std::unique\_ptr< [TorqueMotorConfiguration](#) > config, std::shared\_ptr< [ODrive](#) > odrive, double gear\_ratio)
- **TorqueMotor** (std::shared\_ptr< [ODrive](#) > odrive, double gear\_ratio)
- **TorqueMotor** (std::shared\_ptr< [ODrive](#) > odrive)
- **TorqueMotor** (const [TorqueMotor](#) &)=delete
- [TorqueMotor](#) & **operator=** (const [TorqueMotor](#) &)=delete
- void **Initialize** () override
- void **SetUp** () override
- void **Terminate** () override
- bool **Move** (double value) override
- bool **Move** (const std::vector< double > &values) override
- bool **Move** (const Eigen::VectorXd &values) override
- bool **Stop** () override

## Additional Inherited Members

### 8.61.1 Member Function Documentation

#### 8.61.1.1 Initialize()

`void huron::odrive::TorqueMotor::Initialize ( ) [override], [virtual]`  
 Implements [huron::GenericComponent](#).

#### 8.61.1.2 Move() [1/3]

`bool huron::odrive::TorqueMotor::Move (`  
     `const Eigen::VectorXd & values ) [override], [virtual]`  
 Implements [huron::MovingInterface](#).

#### 8.61.1.3 Move() [2/3]

`bool huron::odrive::TorqueMotor::Move (`  
     `const std::vector< double > & values ) [override], [virtual]`

Moves the component by the specified input vector.

This method can be used if the component needs more than one input. For example, a position controlled motor needs position input, velocity feedforward, and current feedforward.

##### Parameters

<i>value</i>	Input value vector.
--------------	---------------------

##### Returns

true if the operation is successful, false otherwise.

Implements [huron::MovingInterface](#).

#### 8.61.1.4 Move() [3/3]

`bool huron::odrive::TorqueMotor::Move (`  
     `double value ) [override], [virtual]`  
 Implements [huron::Motor](#).

#### 8.61.1.5 SetUp()

`void huron::odrive::TorqueMotor::SetUp ( ) [override], [virtual]`  
 Implements [huron::GenericComponent](#).

#### 8.61.1.6 Stop()

`bool huron::odrive::TorqueMotor::Stop ( ) [override], [virtual]`  
 Stops the component from moving.

##### Returns

true if the operation is successful, false otherwise.

Implements [huron::MovingInterface](#).



### 8.61.1.7 Terminate()

void huron::odrive::TorqueMotor::Terminate ( ) [override], [virtual]

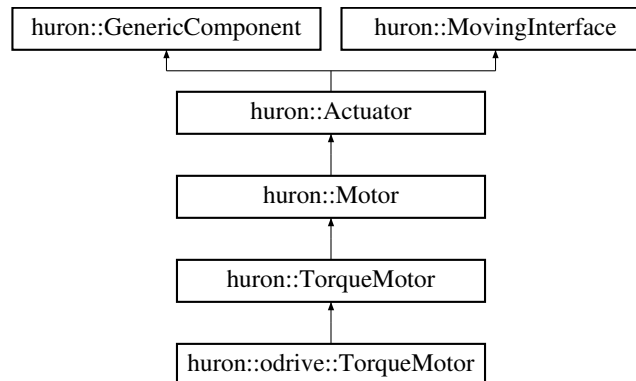
Implements [huron::GenericComponent](#).

The documentation for this class was generated from the following files:

- /github/workspace/system/odrive/include/huron/odrive/odrive\_torque\_motor.h
- /github/workspace/system/odrive/src/odrive\_torque\_motor.cc

## 8.62 huron::TorqueMotor Class Reference

Inheritance diagram for huron::TorqueMotor:



### Public Member Functions

- **TorqueMotor** (std::unique\_ptr< [TorqueMotorConfiguration](#) > config, double gear\_ratio)
- **TorqueMotor** (double gear\_ratio)
- **TorqueMotor** (const [TorqueMotor](#) &)=delete
- [TorqueMotor](#) & **operator=** (const [TorqueMotor](#) &)=delete

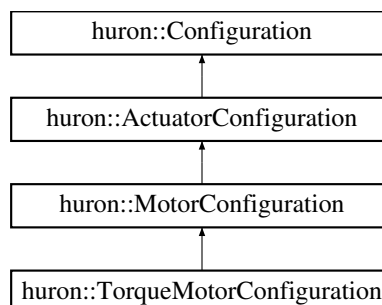
### Additional Inherited Members

The documentation for this class was generated from the following file:

- /github/workspace/system/control\_interfaces/include/huron/control\_interfaces/torque\_motor.h

## 8.63 huron::TorqueMotorConfiguration Class Reference

Inheritance diagram for huron::TorqueMotorConfiguration:



### Public Member Functions

- [TorqueMotorConfiguration](#) (ConfigMap config\_map, std::set< std::string > valid\_keys)

## Additional Inherited Members

### 8.63.1 Constructor & Destructor Documentation

#### 8.63.1.1 TorqueMotorConfiguration()

```
huron::TorqueMotorConfiguration::TorqueMotorConfiguration (
    ConfigMap config_map,
    std::set< std::string > valid_keys ) [inline]
```

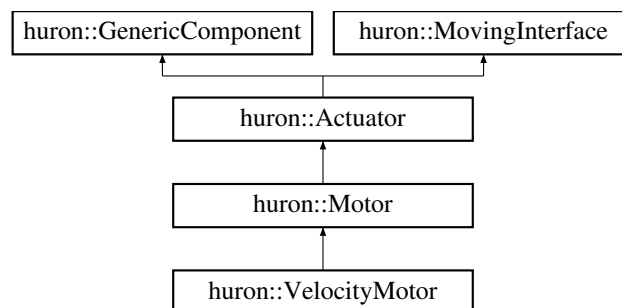
Supports further inheritance.

The documentation for this class was generated from the following file:

- /github/workspace/system/control\_interfaces/include/huron/control\_interfaces/torque\_motor.h

## 8.64 huron::VelocityMotor Class Reference

Inheritance diagram for huron::VelocityMotor:



### Public Member Functions

- **VelocityMotor** (std::unique\_ptr< [VelocityMotorConfiguration](#) > config, double gear\_ratio)
- **VelocityMotor** (double gear\_ratio)
- **VelocityMotor** (const [VelocityMotor](#) &)=delete
- [VelocityMotor](#) & **operator=** (const [VelocityMotor](#) &)=delete

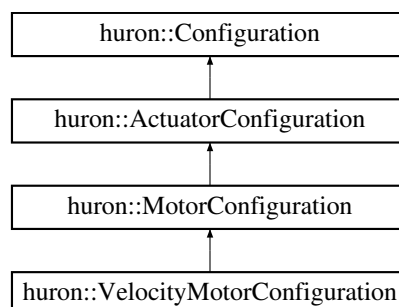
### Additional Inherited Members

The documentation for this class was generated from the following file:

- /github/workspace/system/control\_interfaces/include/huron/control\_interfaces/velocity\_motor.h

## 8.65 huron::VelocityMotorConfiguration Class Reference

Inheritance diagram for huron::VelocityMotorConfiguration:



## Public Member Functions

- [VelocityMotorConfiguration](#) (ConfigMap config\_map, std::set< std::string > valid\_keys)

## Additional Inherited Members

### 8.65.1 Constructor & Destructor Documentation

#### 8.65.1.1 VelocityMotorConfiguration()

```
huron::VelocityMotorConfiguration::VelocityMotorConfiguration (
    ConfigMap config_map,
    std::set< std::string > valid_keys ) [inline]
```

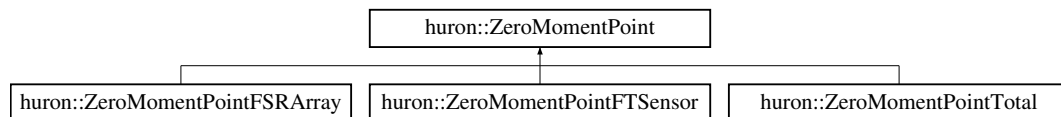
Supports further inheritance.

The documentation for this class was generated from the following file:

- /github/workspace/system/control\_interfaces/include/huron/control\_interfaces/velocity\_motor.h

## 8.66 huron::ZeroMomentPoint Class Reference

Inheritance diagram for huron::ZeroMomentPoint:



## Public Member Functions

- **ZeroMomentPoint** (std::weak\_ptr< const [multibody::Frame](#) > zmp\_frame, double normal\_force\_threshold)
- **ZeroMomentPoint** (const [ZeroMomentPoint](#) &)=delete
- [ZeroMomentPoint](#) & **operator=** (const [ZeroMomentPoint](#) &)=delete
- virtual Eigen::Vector2d **Eval** (double &fz)=0
- Eigen::Vector2d **Eval** ()
- Eigen::Affine3d **ZmpToWorld** (const Eigen::Vector2d &zmp) const

## Protected Attributes

- std::weak\_ptr< const [multibody::Frame](#) > **zmp\_frame\_**
- double **normal\_force\_threshold\_**

### 8.66.1 Member Function Documentation

#### 8.66.1.1 Eval()

```
virtual Eigen::Vector2d huron::ZeroMomentPoint::Eval (
    double & fz ) [pure virtual]
```

Evaluate the zero moment point in the ZMP frame based on the current sensor and joint states.

#### Parameters

<i>zmp</i>	The zero moment point in the ZMP frame.
<i>fz</i>	The normal force.

Implemented in [huron::ZeroMomentPointFSRArray](#), [huron::ZeroMomentPointFTSensor](#), and [huron::ZeroMomentPointTotal](#).

### 8.66.1.2 ZmpToWorld()

```
Eigen::Affine3d huron::ZeroMomentPoint::ZmpToWorld (
    const Eigen::Vector2d & zmp ) const
```

Convert the zero moment point from the 2D ZMP frame to the world frame.

#### Parameters

<i>zmp</i>	The zero moment point in the ZMP frame.
------------	---

#### Returns

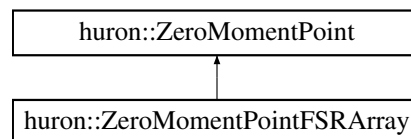
The zero moment point in the world frame.

The documentation for this class was generated from the following files:

- [/github/workspace/system/locomotion/include/huron/locomotion/zero\\_moment\\_point.h](#)
- [/github/workspace/system/locomotion/src/zero\\_moment\\_point.cc](#)

## 8.67 huron::ZeroMomentPointFSRArray Class Reference

Inheritance diagram for [huron::ZeroMomentPointFSRArray](#):



### Public Member Functions

- **ZeroMomentPointFSRArray** (std::weak\_ptr< const [multibody::Frame](#) > zmp\_frame, double normal\_force\_threshold, std::shared\_ptr< [ForceSensingResistorArray](#) > fsr\_array)
- **ZeroMomentPointFSRArray** (const [ZeroMomentPointFSRArray](#) &)=delete
- **ZeroMomentPointFSRArray** & **operator=** (const [ZeroMomentPointFSRArray](#) &)=delete
- Eigen::Vector2d **Eval** (double &fz) override

### Additional Inherited Members

#### 8.67.1 Member Function Documentation

##### 8.67.1.1 Eval()

```
Eigen::Vector2d huron::ZeroMomentPointFSRArray::Eval (
    double & fz ) [override], [virtual]
```

Evaluate the zero moment point in the ZMP frame based on the current sensor and joint states.

#### Parameters

<i>zmp</i>	The zero moment point in the ZMP frame.
<i>fz</i>	The normal force.

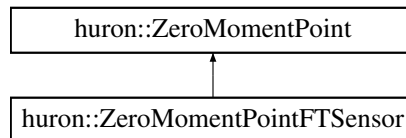
Implements [huron::ZeroMomentPoint](#).

The documentation for this class was generated from the following files:

- /github/workspace/system/locomotion/include/huron/locomotion/zero\_moment\_point\_fsr\_array.h
- /github/workspace/system/locomotion/src/zero\_moment\_point\_fsr\_array.cc

## 8.68 huron::ZeroMomentPointFTSensor Class Reference

Inheritance diagram for huron::ZeroMomentPointFTSensor:



### Public Member Functions

- **ZeroMomentPointFTSensor** (std::weak\_ptr< const [multibody::Frame](#) > zmp\_frame, double normal\_force\_threshold, const std::vector< std::shared\_ptr< [ForceTorqueSensor](#) > > &ft\_sensors)
- **ZeroMomentPointFTSensor** (const [ZeroMomentPointFTSensor](#) &)=delete
- [ZeroMomentPointFTSensor](#) & **operator=** (const [ZeroMomentPointFTSensor](#) &)=delete
- Eigen::Vector2d **Eval** (double &fz) override

### Additional Inherited Members

#### 8.68.1 Member Function Documentation

##### 8.68.1.1 Eval()

```

Eigen::Vector2d huron::ZeroMomentPointFTSensor::Eval (
    double & fz ) [override], [virtual]
  
```

Evaluate the zero moment point in the ZMP frame based on the current sensor and joint states.

#### Parameters

<i>zmp</i>	The zero moment point in the ZMP frame.
<i>fz</i>	The normal force.

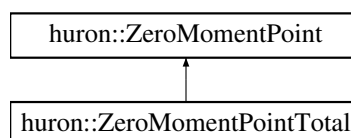
Implements [huron::ZeroMomentPoint](#).

The documentation for this class was generated from the following files:

- /github/workspace/system/locomotion/include/huron/locomotion/zero\_moment\_point\_ft\_sensor.h
- /github/workspace/system/locomotion/src/zero\_moment\_point\_ft\_sensor.cc

## 8.69 huron::ZeroMomentPointTotal Class Reference

Inheritance diagram for huron::ZeroMomentPointTotal:



## Public Member Functions

- **ZeroMomentPointTotal** (std::weak\_ptr< const [multibody::Frame](#) > zmp\_frame, const std::vector< std::shared\_ptr< [ZeroMomentPoint](#) > > &zmp\_vector)
- **ZeroMomentPointTotal** (const [ZeroMomentPointTotal](#) &)=delete
- **ZeroMomentPointTotal** & **operator=** (const [ZeroMomentPointTotal](#) &)=delete
- Eigen::Vector2d **Eval** (double &fz) override

## Additional Inherited Members

### 8.69.1 Member Function Documentation

#### 8.69.1.1 Eval()

```
Eigen::Vector2d huron::ZeroMomentPointTotal::Eval (
    double & fz ) [override], [virtual]
```

Evaluate the zero moment point in the ZMP frame based on the current sensor and joint states.

#### Parameters

<i>zmp</i>	The zero moment point in the ZMP frame.
<i>fz</i>	The normal force.

Implements [huron::ZeroMomentPoint](#).

The documentation for this class was generated from the following files:

- /github/workspace/system/locomotion/include/huron/locomotion/zero\_moment\_point\_total.h
- /github/workspace/system/locomotion/src/zero\_moment\_point\_total.cc

## Chapter 9

# File Documentation

### 9.1 enable\_protected\_make\_shared.h

```
1 #pragma once
2
3 #include <memory>
4 #include <utility>
5
6 namespace huron {
7
8     template <typename ClassWithProtectedCtor>
9     class enable_protected_make_shared {
10     protected:
11         template <typename... Args>
12         static std::shared_ptr<ClassWithProtectedCtor> make_shared(Args &&... args) {
13             class make_shared_enabler : public ClassWithProtectedCtor {
14             public:
15                 // Ensures that the constructor is not public.
16                 static_assert(!std::is_constructible_v<ClassWithProtectedCtor, Args...>);
17                 explicit make_shared_enabler(Args &&... args)
18                     : ClassWithProtectedCtor(std::forward<Args>(args)...) {}
19             };
20             return std::make_shared<make_shared_enabler>(std::forward<Args>(args)...)
21         }
22     };
23 }
24
25 // namespace huron
```

### 9.2 enable\_protected\_make\_unique.h

```
1 #pragma once
2
3 #include <memory>
4 #include <utility>
5
6 namespace huron {
7
8     template <typename ClassWithProtectedCtor>
9     class enable_protected_make_unique {
10     protected:
11         template <typename... Args>
12         static std::unique_ptr<ClassWithProtectedCtor> make_unique(Args &&... args) {
13             class make_unique_enabler : public ClassWithProtectedCtor {
14             public:
15                 // Ensures that the constructor is not public.
16                 static_assert(!std::is_constructible_v<ClassWithProtectedCtor, Args...>);
17                 explicit make_unique_enabler(Args &&... args)
18                     : ClassWithProtectedCtor(std::forward<Args>(args)...) {}
19             };
20             return std::make_unique<make_unique_enabler>(std::forward<Args>(args)...)
21         }
22     };
23 }
24
25 // namespace huron
```

### 9.3 types.h

```
1 #pragma once
2
3 #include <eigen3/Eigen/Core>
```

```

4
5 namespace huron {
6
7 typedef Eigen::Matrix< double, 6, 1 > Vector6d;
8 typedef Eigen::Matrix< double, 6, 6 > Matrix6d;
9 typedef Eigen::Matrix< double, 6, Eigen::Dynamic > Matrix6Xd;
10
11 } // namespace huron

```

## 9.4 push\_recovery.h

```

1 #pragma once
2
3 #include <eigen3/Eigen/Dense>
4
5 #include <iostream>
6 #include <complex>
7 #include <vector>
8
9 class PushRecoveryControl {
10 private:
11     static const inline std::complex<double> i{0.0, 1.0};
12     // EOM of 3 DOF model
13     // Mass in kg, length in meter
14     static constexpr double alpha = 0.7;
15     static constexpr double m1 = 5.9117,
16         m2 = 4.2554,
17         m3 = 10.19329;
18
19     static constexpr double lc1 = 0.15149,
20         lc2 = 0.24517,
21         lc3 = 0.1585;
22
23     static constexpr double l1 = 0.3715,
24         l2 = 0.49478,
25         l3 = 0.32662;
26
27     static constexpr double g = 9.81;
28     static constexpr double I1 = 0.0222,
29         I2 = 0.01009,
30         I3 = 0.0219;
31
32     // Desired position, velocity and acceleration of location of the com
33
34     double theta1_d = 0,
35         theta2_d = 0,
36         theta3_d = 0;
37
38     double theta1_dot_d = 0,
39         theta2_dot_d = 0,
40         theta3_dot_d = 0;
41
42     double theta1_ddd = 0,
43         theta2_ddd = 0,
44         theta3_ddd = 0;
45
46     double x_com_d = 0,
47         x_com_dd = 0,
48         x_com_ddd = 0;
49
50     // Values
51     double theta1, theta2, theta3 = 0;
52     double theta1_dot, theta2_dot, theta3_dot = 0;
53     double X_COM, X_dot_COM = 0;
54     Eigen::MatrixXd ModelCalculation();
55     Eigen::MatrixXd CalculateCOM();
56     template <typename T>
57     int sign (const T &val) { return (val > 0) - (val < 0); }
58     Eigen::MatrixXd SMCController(const Eigen::Vector2d& cop,
59         const Eigen::MatrixXd& J_X_COM,
60         const Eigen::MatrixXd& J_X_COM_dot);
61     Eigen::MatrixXd SMCPostureCorrection();
62     double constrainAngle(double x);
63
64 public:
65     Eigen::MatrixXd GetTorque(const Eigen::Vector2d& cop,
66         const Eigen::VectorXd& position,
67         const Eigen::VectorXd& velocity);
68 };

```



## 9.5 can\_helpers.h

```

1  /*
2  * Original source from: https://github.com/odriverobotics/ODrive/tree/master
3  */
4  #pragma once
5
6  #include <stdint.h>
7  #include <algorithm>
8  #include <cstring>
9  #include <iterator>
10
11 struct can_Message_t {
12     uint32_t id = 0x000; // 11-bit max is 0x7ff, 29-bit max is 0x1FFFFFFF
13     bool isExt = false;
14     bool rtr = false;
15     uint8_t len = 8;
16     uint8_t buf[8] = {0, 0, 0, 0, 0, 0, 0, 0};
17 };
18
19 struct can_Signal_t {
20     const uint8_t startBit;
21     const uint8_t length;
22     const bool isIntel;
23     const float factor;
24     const float offset;
25 };
26
27 struct can_Cyclic_t {
28     uint32_t cycleTime_ms;
29     uint32_t lastTime_ms;
30 };
31
32 template <typename T>
33 constexpr T can_getSignal(can_Message_t msg, const uint8_t startBit,
34                          const uint8_t length, const bool isIntel) {
35     uint64_t tempVal = 0;
36     uint64_t mask = length < 64 ? (1ULL « length) - 1ULL : -1ULL;
37
38     if (isIntel) {
39         std::memcpy(&tempVal, msg.buf, sizeof(tempVal));
40         tempVal = (tempVal « startBit) & mask;
41     } else {
42         std::reverse(std::begin(msg.buf), std::end(msg.buf));
43         std::memcpy(&tempVal, msg.buf, sizeof(tempVal));
44         tempVal = (tempVal « (64 - startBit - length)) & mask;
45     }
46
47     T retVal;
48     std::memcpy(&retVal, &tempVal, sizeof(T));
49     return retVal;
50 }
51
52 template <typename T>
53 constexpr void can_setSignal(can_Message_t& msg, const T& val,
54                             const uint8_t startBit, const uint8_t length,
55                             const bool isIntel) {
56     uint64_t valAsBits = 0;
57     std::memcpy(&valAsBits, &val, sizeof(val));
58
59     uint64_t mask = length < 64 ? (1ULL « length) - 1ULL : -1ULL;
60
61     if (isIntel) {
62         uint64_t data = 0;
63         std::memcpy(&data, msg.buf, sizeof(data));
64
65         data &= ~(mask « startBit);
66         data |= valAsBits « startBit;
67
68         std::memcpy(msg.buf, &data, sizeof(data));
69     } else {
70         uint64_t data = 0;
71         std::reverse(std::begin(msg.buf), std::end(msg.buf));
72         std::memcpy(&data, msg.buf, sizeof(data));
73
74         data &= ~(mask « (64 - startBit - length));
75         data |= valAsBits « (64 - startBit - length);
76
77         std::memcpy(msg.buf, &data, sizeof(data));
78         std::reverse(std::begin(msg.buf), std::end(msg.buf));
79     }
80 }
81
82 template<typename T>
83 void can_setSignal(can_Message_t& msg, const T& val, const uint8_t startBit,
84                  const uint8_t length, const bool isIntel, const float factor,
85                  const float offset) {

```

```

86   T scaledVal = static_cast<T>((val - offset) / factor);
87   can_setSignal<T>(msg, scaledVal, startBit, length, isIntel);
88 }
89
90 template<typename T>
91 float can_getSignal(can_Message_t msg, const uint8_t startBit,
92                    const uint8_t length, const bool isIntel,
93                    const float factor, const float offset) {
94   T retVal = can_getSignal<T>(msg, startBit, length, isIntel);
95   return (retVal * factor) + offset;
96 }
97
98 template <typename T>
99 float can_getSignal(can_Message_t msg, const can_Signal_t& signal) {
100    return can_getSignal<T>(msg, signal.startBit, signal.length, signal.isIntel,
101                             signal.factor, signal.offset);
102 }
103
104 template <typename T>
105 void can_setSignal(can_Message_t& msg, const T& val,
106                  const can_Signal_t& signal) {
107    can_setSignal(msg, val, signal.startBit, signal.length, signal.isIntel,
108                  signal.factor, signal.offset);
109 }

```

## 9.6 canbus.h

```

1  /*
2  * Original source from: https://github.com/odriverobotics/ODrive/tree/master
3  * */
4  #pragma once
5
6  #include <variant>
7  #include "can_helpers.h"
8
9  namespace huron {
10 namespace driver {
11 namespace can {
12
13 struct MsgIdFilterSpecs {
14     std::variant<uint16_t, uint32_t> id;
15     uint32_t mask;
16 };
17
18 class BusBase {
19 public:
20     typedef void(*on_can_message_cb_t)(void* ctx, const can_Message_t& message);
21     struct CanSubscription {};
22
23     BusBase() = default;
24     BusBase(const BusBase&) = delete;
25     BusBase& operator=(const BusBase&) = delete;
26     virtual ~BusBase() = default;
27
28     virtual bool send_message(const can_Message_t& message) = 0;
29
30     virtual bool recv_message(can_Message_t& message,
31                               uint32_t timeout = UINT32_MAX) = 0;
32
33     virtual bool subscribe(const MsgIdFilterSpecs& filter,
34                             on_can_message_cb_t callback, void* ctx,
35                             CanSubscription** handle) = 0;
36
37     virtual bool unsubscribe(CanSubscription* handle) = 0;
38 };
39
40 } // namespace can
41 } // namespace driver
42 } // namespace huron

```

## 9.7 socket\_can\_bus.h

```

1  #pragma once
2
3  #include <string>
4  #include <sockcanpp/CanDriver.hpp>
5
6  #include "canbus.h"
7
8  #define CAN_CLK_HZ (16000000)
9  #define CAN_CLK_MHZ (16)
10

```

```

11 namespace huron {
12 namespace driver {
13 namespace can {
14
15 // Anonymous enum for defining the most common CAN baud rates
16 enum {
17     CAN_BAUD_125K = 125000,
18     CAN_BAUD_250K = 250000,
19     CAN_BAUD_500K = 500000,
20     CAN_BAUD_1000K = 1000000,
21     CAN_BAUD_1M = 1000000
22 };
23
24 class SocketCanBus : public BusBase {
25 public:
26     static constexpr sockcanpp::milliseconds kRecvTimeout{1000};
27     // struct Config_t {
28     //     uint32_t baud_rate = CAN_BAUD_250K;
29     //     Protocol protocol = PROTOCOL_SIMPLE;
30     //
31     //     Bus* parent = nullptr; // set in apply_config()
32     //     void set_baud_rate(uint32_t value) { parent->set_baud_rate(value); }
33     // };
34
35     SocketCanBus(std::string can_if, uint32_t axis_id)
36         : can_if_(can_if), axis_id_(axis_id), recv_timeout_(kRecvTimeout) {}
37     SocketCanBus(const SocketCanBus&) = delete;
38     SocketCanBus& operator=(const SocketCanBus&) = delete;
39     ~SocketCanBus() = default;
40
41     std::string can_if_;
42     uint32_t axis_id_;
43     sockcanpp::milliseconds recv_timeout_;
44     sockcanpp::CanDriver can_driver_{can_if_, CAN_RAW};
45
46     static const uint8_t kCanFifoNone = 0xff;
47
48     bool send_message(const can_Message_t& message) final;
49     bool recv_message(can_Message_t& message,
50                     uint32_t timeout = UINT32_MAX) final;
51     bool subscribe(const MsgIdFilterSpecs& filter,
52                 on_can_message_cb_t callback, void* ctx,
53                 CanSubscription** handle) final;
54     bool unsubscribe(CanSubscription* handle) final;
55 };
56
57 } // namespace can
58 } // namespace driver
59 } // namespace huron

```

## 9.8 config.h

```

1 #pragma once
2
3 #include "odrive_config.h"

```

## 9.9 odrive\_config.h

```

1 #pragma once
2
3 #define AXIS_COUNT (2)
4
5 #define ODRIVE_VELOCITY_LIMIT (15.0)
6 #define ODRIVE_CURRENT_LIMIT (70.0)

```

## 9.10 serial.h

```

1 #pragma once
2
3 #include <cstdint>
4 #include <cstdint>
5 #include <vector>
6 #include <string>
7
8 namespace huron {
9 namespace driver {
10 namespace serial {
11
12 enum class Parity {

```

```

13  None = 0,
14  Odd = 1,
15  Even = 2,
16  Mark = 3,
17  Space = 4
18  };
19
20  enum class StopBits {
21      One = 1,
22      Two = 2,
23      OnePointFive
24  };
25
26  enum class FlowControl {
27      None = 0,
28      Software,
29      Hardware
30  };
31
32  class SerialBase {
33  public:
34      SerialBase(std::string port,
35                 uint32_t baudrate,
36                 Parity parity,
37                 StopBits stopbits,
38                 FlowControl flowcontrol) :
39          port_(port),
40          baudrate_(baudrate),
41          parity_(parity),
42          stopbits_(stopbits),
43          flowcontrol_(flowcontrol) {}
44      SerialBase(const SerialBase&) = delete;
45      SerialBase& operator=(const SerialBase&) = delete;
46      virtual ~SerialBase() = default;
47
48      virtual void Open() = 0;
49      virtual bool IsOpen() = 0;
50      virtual void Close() = 0;
51      virtual size_t Available() = 0;
52      virtual bool WaitReadable() = 0;
53      virtual size_t Read(uint8_t *buffer, size_t nbytes) = 0;
54      virtual size_t Read(std::vector<uint8_t> &buffer, size_t nbytes = 1) = 0;
55      virtual size_t Read(std::string &buffer, size_t nbytes = 1) = 0;
56      virtual std::string Read(size_t nbytes = 1) = 0;
57      virtual size_t ReadLine(std::string &buffer,
58                             size_t nbytes = 65536,
59                             std::string eol = "\n") = 0;
60      virtual std::string ReadLine(size_t nbytes = 65536,
61                                  std::string eol = "\n") = 0;
62      virtual std::vector<std::string> ReadLines(size_t nbytes = 65536,
63                                                std::string eol = "\n") = 0;
64      virtual size_t Write(const uint8_t *data, size_t nbytes) = 0;
65      virtual size_t Write(const std::vector<uint8_t> &data) = 0;
66      virtual size_t Write(const std::string &data) = 0;
67      virtual void SetPort(const std::string &port) = 0;
68      virtual std::string GetPort() const = 0;
69      virtual void SetTimeout(uint32_t inter_byte_timeout,
70                             uint32_t read_timeout_constant,
71                             uint32_t read_timeout_multiplier,
72                             uint32_t write_timeout_constant,
73                             uint32_t write_timeout_multiplier) = 0;
74      virtual void SetBaudrate(uint32_t baudrate) = 0;
75      virtual uint32_t GetBaudrate() const = 0;
76      virtual void SetParity(Parity parity) = 0;
77      virtual Parity GetParity() const = 0;
78      virtual void SetStopbits(StopBits stopbits) = 0;
79      virtual StopBits GetStopbits() const = 0;
80      virtual void SetFlowcontrol(FlowControl flowcontrol) = 0;
81      virtual FlowControl GetFlowcontrol() const = 0;
82      virtual void Flush() = 0;
83      virtual void FlushInput() = 0;
84      virtual void FlushOutput() = 0;
85      virtual void SendBreak(int duration) = 0;
86
87  protected:
88      std::string port_;
89      uint32_t baudrate_;
90      Parity parity_;
91      StopBits stopbits_;
92      FlowControl flowcontrol_;
93  };
94
95 } // namespace serial
96 } // namespace driver
97 } // namespace huron

```

## 9.11 wjwwood\_serial.h

```

1 #pragma once
2
3 #include <serial/serial.h>
4
5 #include <memory>
6 #include <vector>
7 #include <string>
8
9 #include "serial.h"
10
11 namespace huron {
12 namespace driver {
13 namespace serial {
14
15 class Serial : public SerialBase{
16 public:
17     Serial(std::string port,
18           uint32_t baudrate,
19           Parity parity,
20           StopBits stopbits,
21           FlowControl flowcontrol);
22     Serial(const Serial&) = delete;
23     Serial& operator=(const Serial&) = delete;
24     virtual ~Serial() = default;
25
26     void Open() override;
27     bool IsOpen() override;
28     void Close() override;
29     size_t Available() override;
30     bool WaitReadable() override;
31     size_t Read(uint8_t *buffer, size_t nbytes) override;
32     size_t Read(std::vector<uint8_t> &buffer, size_t nbytes = 1) override;
33     size_t Read(std::string &buffer, size_t nbytes = 1) override;
34     std::string Read(size_t nbytes = 1) override;
35     size_t ReadLine(std::string &buffer,
36                   size_t nbytes = 65536,
37                   std::string eol = "\n") override;
38     std::string ReadLine(size_t nbytes = 65536,
39                       std::string eol = "\n") override;
40     std::vector<std::string> ReadLines(size_t nbytes = 65536,
41                                     std::string eol = "\n") override;
42     size_t Write(const uint8_t *data, size_t nbytes) override;
43     size_t Write(const std::vector<uint8_t> &data) override;
44     size_t Write(const std::string &data) override;
45     void SetPort(const std::string &port) override;
46     std::string GetPort() const override;
47     void SetTimeout(uint32_t inter_byte_timeout,
48                   uint32_t read_timeout_constant,
49                   uint32_t read_timeout_multiplier,
50                   uint32_t write_timeout_constant,
51                   uint32_t write_timeout_multiplier) override;
52     void SetBaudrate(uint32_t baudrate) override;
53     uint32_t GetBaudrate() const override;
54     void SetParity(Parity parity) override;
55     Parity GetParity() const override;
56     void SetStopbits(StopBits stopbits) override;
57     StopBits GetStopbits() const override;
58     void SetFlowcontrol(FlowControl flowcontrol) override;
59     FlowControl GetFlowcontrol() const override;
60     void Flush() override;
61     void FlushInput() override;
62     void FlushOutput() override;
63     void SendBreak(int duration) override;
64
65 private:
66     std::unique_ptr<::serial::Serial> wjwwood_serial_;
67
68     static ::serial::flowcontrol_t ConvertFlowControl(FlowControl flowcontrol) {
69         return static_cast<::serial::flowcontrol_t>(flowcontrol);
70     }
71     static ::serial::parity_t ConvertParity(Parity parity) {
72         return static_cast<::serial::parity_t>(parity);
73     }
74     static ::serial::stopbits_t ConvertStopBits(StopBits stopbits) {
75         return static_cast<::serial::stopbits_t>(stopbits);
76     }
77 };
78
79 } // namespace serial
80 } // namespace driver
81 } // namespace huron

```

## 9.12 rotation.h

```

1 #pragma once
2
3 #include <eigen3/Eigen/Core>
4 #include <cmath>
5
6 namespace huron {
7 namespace math {
8
9 Eigen::Vector3d ZyxToRpy(
10     const Eigen::Ref<const Eigen::Vector3d>& zyx);
11
12 Eigen::Vector3d RpyToZyx(
13     const Eigen::Ref<const Eigen::Vector3d>& zyx);
14
15 Eigen::Matrix3d ZyxToRotationMatrix(
16     const Eigen::Ref<const Eigen::Vector3d>& zyx);
17
18 Eigen::Vector3d RotationMatrixToZyx(
19     const Eigen::Ref<const Eigen::Matrix3d>& rotation_matrix);
20
21 Eigen::Matrix3d RpyToRotationMatrix(
22     const Eigen::Ref<const Eigen::Vector3d>& rpy);
23
24 Eigen::Vector3d RotationMatrixToRpy(
25     const Eigen::Ref<const Eigen::Matrix3d>& rotation_matrix);
26
27
28 } // namespace math
29 } // namespace huron

```

## 9.13 com\_frame.h

```

1 #pragma once
2
3 #include <string>
4 #include <memory>
5
6 #include "huron/multibody/frame.h"
7
8 namespace huron {
9 namespace multibody {
10
11 class ComFrame : public Frame {
12 public:
13     ComFrame(FrameIndex index,
14             const std::string& name,
15             bool is_user_defined,
16             std::weak_ptr<const Model> model,
17             FrameIndex parent_frame_index);
18
19     ComFrame(const ComFrame&) = delete;
20     ComFrame& operator=(const ComFrame&) = delete;
21     ~ComFrame() override = default;
22
23     Eigen::Affine3d GetTransformInWorld() const override;
24     Eigen::Affine3d GetTransformFromFrame(const Frame& other) const override;
25     Eigen::Affine3d GetTransformFromFrame(FrameIndex other) const override;
26     Eigen::Affine3d GetTransformToFrame(const Frame& other) const override;
27     Eigen::Affine3d GetTransformToFrame(FrameIndex other) const override;
28
29 private:
30     FrameIndex parent_frame_index_;
31
32     Eigen::Affine3d ParentToThisTransform() const;
33 };
34
35 } // namespace multibody
36 } // namespace huron

```

## 9.14 frame.h

```

1 #pragma once
2
3 #include <eigen3/Eigen/Dense>
4
5 #include <memory>
6 #include <string>
7
8 #include "huron/enable_protected_make_shared.h"
9

```

```

10 namespace huron {
11 namespace multibody {
12
13 class Model;
14
15 using FrameIndex = size_t;
16
17 enum class FrameType {
18     kLogical,
19     kFixed,
20     kJoint,
21     kSensor,
22     kPhysical,
23 };
24
25 class Frame : public enable_protected_make_shared<Frame> {
26 public:
27     friend class Model;
28
29     Frame(const Frame&) = delete;
30     Frame& operator=(const Frame&) = delete;
31     virtual ~Frame() = default;
32
33     virtual Eigen::Affine3d GetTransformInWorld() const;
34     virtual Eigen::Affine3d GetTransformFromFrame(const Frame& other) const;
35     virtual Eigen::Affine3d GetTransformFromFrame(FrameIndex other) const;
36     virtual Eigen::Affine3d GetTransformToFrame(const Frame& other) const;
37     virtual Eigen::Affine3d GetTransformToFrame(FrameIndex other) const;
38
39     const std::string& name() const { return name_; }
40     FrameIndex index() const { return index_; }
41     FrameType type() const { return type_; }
42     bool is_user_defined() const { return is_user_defined_; }
43
44 protected:
45     Frame(FrameIndex index,
46           const std::string& name,
47           FrameType type,
48           bool is_user_defined,
49           std::weak_ptr<const Model> model);
50
51     const FrameIndex index_;
52     const std::string name_;
53     const FrameType type_;
54     bool is_user_defined_;
55     const std::weak_ptr<const Model> model_;
56 };
57
58
59 } // namespace multibody
60 } // namespace huron

```

## 9.15 joint\_common.h

```

1 #pragma once
2
3 #include <eigen3/Eigen/Dense>
4
5 #include <stddef.h>
6 #include <string>
7 #include <cassert>
8 #include <limits>
9
10 #include "frame.h"
11
12 namespace huron {
13 namespace multibody {
14
15 using JointIndex = size_t;
16
17 enum class JointType {
18     kUnknown = 0,
19     kFixed,
20     kRevolute,
21     kPrismatic,
22     kPlanar,
23     kSpherical,
24     kFreeFlyer
25 };
26
27 struct JointDescription {
28 public:
29     // TODO(dtbpkmt): Properly implement JointIndex/FrameIndex and casts.
30     JointDescription(size_t id, const std::string& name,
31                     size_t parent_frame_id,
32                     size_t child_frame_id,

```

```

33         size_t num_positions, size_t num_velocities,
34         JointType type,
35         const Eigen::VectorXd& min_position,
36         const Eigen::VectorXd& max_position,
37         const Eigen::VectorXd& min_velocity,
38         const Eigen::VectorXd& max_velocity,
39         const Eigen::VectorXd& min_acceleration,
40         const Eigen::VectorXd& max_acceleration,
41         const Eigen::VectorXd& min_torque,
42         const Eigen::VectorXd& max_torque,
43         const Eigen::VectorXd& friction,
44         const Eigen::VectorXd& damping)
45     : id_((JointIndex) id),
46       name_(name),
47       parent_frame_id_((FrameIndex) parent_frame_id),
48       child_frame_id_((FrameIndex) child_frame_id),
49       num_positions_(num_positions),
50       num_velocities_(num_velocities),
51       type_(type),
52       min_position_(min_position),
53       max_position_(max_position),
54       min_velocity_(min_velocity),
55       max_velocity_(max_velocity),
56       min_acceleration_(min_acceleration),
57       max_acceleration_(max_acceleration),
58       min_torque_(min_torque),
59       max_torque_(max_torque),
60       friction_(friction),
61       damping_(damping) {
62     assert(min_position.size() == num_positions_);
63     assert(max_position.size() == num_positions_);
64     assert((max_position.array() >= min_position.array()).all());
65
66     assert(min_velocity.size() == num_velocities_);
67     assert(max_velocity.size() == num_velocities_);
68     assert((max_velocity.array() >= min_velocity.array()).all());
69
70     assert(min_acceleration.size() == num_velocities_);
71     assert(max_acceleration.size() == num_velocities_);
72     assert((max_acceleration.array() >= min_acceleration.array()).all());
73
74     assert(min_torque.size() == num_velocities_);
75     assert(max_torque.size() == num_velocities_);
76     assert((max_torque.array() >= min_torque.array()).all());
77
78     assert(friction.size() == num_velocities_);
79     assert(damping.size() == num_velocities_);
80 }
81
82 JointDescription(size_t id, const std::string& name,
83                 size_t parent_frame_id,
84                 size_t child_frame_id,
85                 size_t num_positions, size_t num_velocities,
86                 JointType type,
87                 const Eigen::VectorXd& min_position,
88                 const Eigen::VectorXd& max_position,
89                 const Eigen::VectorXd& min_velocity,
90                 const Eigen::VectorXd& max_velocity,
91                 const Eigen::VectorXd& min_acceleration,
92                 const Eigen::VectorXd& max_acceleration,
93                 const Eigen::VectorXd& min_torque,
94                 const Eigen::VectorXd& max_torque)
95     : JointDescription(id, name,
96                       parent_frame_id, child_frame_id,
97                       num_positions, num_velocities,
98                       type,
99                       min_position, max_position,
100                      min_velocity, max_velocity,
101                      min_acceleration, max_acceleration,
102                      min_torque, max_torque,
103                      Eigen::VectorXd::Zero(num_velocities),
104                      Eigen::VectorXd::Zero(num_velocities)) {}
105
106 JointDescription(size_t id, const std::string& name,
107                 size_t parent_frame_id,
108                 size_t child_frame_id,
109                 size_t num_positions,
110                 size_t num_velocities,
111                 JointType type)
112     : JointDescription(
113       id, name,
114       parent_frame_id, child_frame_id,
115       num_positions, num_velocities,
116       type,
117       Eigen::VectorXd::Constant(num_positions,
118                                -std::numeric_limits<double>::infinity()),
119       Eigen::VectorXd::Constant(num_positions,

```



```

120         std::numeric_limits<double>::infinity()),
121         Eigen::VectorXd::Constant(num_velocities,
122         -std::numeric_limits<double>::infinity()),
123         Eigen::VectorXd::Constant(num_velocities,
124         std::numeric_limits<double>::infinity()),
125         Eigen::VectorXd::Constant(num_velocities,
126         -std::numeric_limits<double>::infinity()),
127         Eigen::VectorXd::Constant(num_velocities,
128         std::numeric_limits<double>::infinity()),
129         Eigen::VectorXd::Constant(num_velocities,
130         -std::numeric_limits<double>::infinity()),
131         Eigen::VectorXd::Constant(num_velocities,
132         std::numeric_limits<double>::infinity()),
133         Eigen::VectorXd::Zero(num_velocities),
134         Eigen::VectorXd::Zero(num_velocities)) {}
135
136     JointIndex id() const { return id_; }
137     const std::string& name() const { return name_; }
138     FrameIndex parent_frame_id() const { return parent_frame_id_; }
139     FrameIndex child_frame_id() const { return child_frame_id_; }
140     size_t num_positions() const { return num_positions_; }
141     size_t num_velocities() const { return num_velocities_; }
142     JointType type() const { return type_; }
143     const Eigen::VectorXd& min_position() const { return min_position_; }
144     const Eigen::VectorXd& max_position() const { return max_position_; }
145     const Eigen::VectorXd& min_velocity() const { return min_velocity_; }
146     const Eigen::VectorXd& max_velocity() const { return max_velocity_; }
147     const Eigen::VectorXd& min_acceleration() const { return min_acceleration_; }
148     const Eigen::VectorXd& max_acceleration() const { return max_acceleration_; }
149     const Eigen::VectorXd& min_torque() const { return min_torque_; }
150     const Eigen::VectorXd& max_torque() const { return max_torque_; }
151     const Eigen::VectorXd& friction() const { return friction_; }
152     const Eigen::VectorXd& damping() const { return damping_; }
153
154 private:
155     JointIndex id_;
156     std::string name_;
157     FrameIndex parent_frame_id_;
158     FrameIndex child_frame_id_;
159     size_t num_positions_;
160     size_t num_velocities_;
161     JointType type_;
162     Eigen::VectorXd min_position_;
163     Eigen::VectorXd max_position_;
164     Eigen::VectorXd min_velocity_;
165     Eigen::VectorXd max_velocity_;
166     Eigen::VectorXd min_acceleration_;
167     Eigen::VectorXd max_acceleration_;
168     Eigen::VectorXd min_torque_;
169     Eigen::VectorXd max_torque_;
170     Eigen::VectorXd friction_;
171     Eigen::VectorXd damping_;
172 };
173
174 std::ostream& operator<<(std::ostream &os, const JointDescription &jd);
175
176 } // namespace multibody
177 } // namespace huron

```

## 9.16 logical\_frame.h

```

1 #pragma once
2
3 #include <string>
4 #include <memory>
5
6 #include "huron/multibody/frame.h"
7
8 namespace huron {
9 namespace multibody {
10
11     class LogicalFrame : public Frame, enable_protected_make_shared<LogicalFrame> {
12     public:
13         friend class Model;
14
15         LogicalFrame(const LogicalFrame&) = delete;
16         LogicalFrame& operator=(const LogicalFrame&) = delete;
17         ~LogicalFrame() override = default;
18
19         Eigen::Affine3d GetTransformInWorld() const override;
20         Eigen::Affine3d GetTransformFromFrame(const Frame& other) const override;
21         Eigen::Affine3d GetTransformFromFrame(FrameIndex other) const override;
22         Eigen::Affine3d GetTransformToFrame(const Frame& other) const override;
23         Eigen::Affine3d GetTransformToFrame(FrameIndex other) const override;
24     };
25 }
26 }

```

```

42 protected:
43     LogicalFrame(FrameIndex index,
44                 const std::string& name,
45                 bool is_user_defined,
46                 std::weak_ptr<const Model> model,
47                 FrameIndex parent_frame_index,
48                 std::function<Eigen::Affine3d(const Eigen::Affine3d&)>
49                 transform_function);
50
51 private:
52     FrameIndex parent_frame_index_;
53     const std::function<Eigen::Affine3d(const Eigen::Affine3d&)>
54     transform_function_;
55 };
56
57 } // namespace multibody
58 } // namespace huron

```

## 9.17 model.h

```

1 #pragma once
2
3 #include <utility>
4 #include <memory>
5 #include <vector>
6 #include <string>
7 #include <unordered_map>
8
9 #include "huron/multibody/model_impl_types.h"
10 #include "huron/multibody/model_impl_interface.h"
11 #include "huron/multibody/frame.h"
12
13 namespace huron {
14 namespace multibody {
15
16 class Model : public std::enable_shared_from_this<Model> {
17     using ModelImplInterface = internal::ModelImplInterface;
18
19 public:
20     Model();
21     Model(const Model&) = delete;
22     Model& operator=(const Model&) = delete;
23     ~Model() = default;
24
25     void AddModelImpl(ModelImplType type, bool set_as_default = false);
26
27     template<typename ...Args>
28     void AddJoint(JointIndex index,
29                 Args&&... args) {
30         assert(!is_constructed_);
31         assert(!is_finalized_);
32         if (joints_[index] != nullptr) {
33             // TODO(dtbpkmt): provide index information in the error message.
34             throw std::runtime_error("Joint already exists at this index.");
35         }
36         joints_[index] = std::make_shared<Joint>(std::forward<Args>(args)...);
37     }
38
39     Joint* const GetJoint(JointIndex index);
40     Joint* const GetJoint(const std::string& name);
41
42     void SetJointStateProvider(JointIndex index,
43                               std::shared_ptr<StateProvider> state_provider);
44
45     JointIndex GetJointIndex(const std::string& joint_name) const;
46     // template<typename FrameImpl, typename ...Args>
47     // std::weak_ptr<const Frame>
48     // AddFrame(const std::string& name, Args&&... args);
49
50     template<typename FrameImpl, typename ...Args>
51     std::weak_ptr<const Frame> AddFrame(const std::string& name, Args&&... args) {
52         static_assert(
53             std::is_base_of_v<Frame, FrameImpl>,
54             "Invalid frame type.");
55         static_assert(
56             std::is_base_of_v<enable_protected_make_shared<FrameImpl>, FrameImpl>,
57             "Frame-derived class must also derive from "
58             "enable_protected_make_shared.");
59         assert(is_constructed_);
60         assert(!is_finalized_);
61
62         DoAddFrame<FrameImpl>(name, true, std::forward<Args>(args)...);
63         return frames_.back();
64     }
65 }
66
67 }

```

```

91  std::weak_ptr<const Frame> GetFrame(FrameIndex index) const;
92  std::weak_ptr<const Frame> GetFrame(const std::string& name) const;
93
94  void BuildFromUrdf(const std::string& urdf_path,
95                    JointType root_joint_type = JointType::kFixed);
96
104 void Finalize(const Eigen::VectorXd& initial_state);
105 void Finalize();
106
110 void UpdateJointStates();
111
112 void SetDefaultModelImpl(size_t index) {
113     default_impl_index_ = index;
114 }
115 size_t GetDefaultModelImpl() const {
116     return default_impl_index_;
117 }
118
119
120 // Kinematics and Dynamics wrapper functions
121 Eigen::Affine3d GetJointTransformInWorld(size_t joint_index) const;
122
123 FrameIndex GetFrameIndex(const std::string& frame_name) const;
124 const std::string& GetFrameName(FrameIndex frame_index) const;
125 Eigen::Affine3d GetFrameTransform(FrameIndex from_frame,
126                                   FrameIndex to_frame) const;
127 Eigen::Affine3d GetFrameTransformInWorld(FrameIndex frame) const;
128
129 Eigen::VectorXd NeutralConfiguration() const;
130
131 Eigen::Vector3d EvalCenterOfMassPosition();
132 Eigen::Vector3d GetCenterOfMassPosition() const;
133
134 const Eigen::VectorBlock<const Eigen::VectorXd> GetPositions() const;
135
136 const Eigen::VectorBlock<const Eigen::VectorXd> GetVelocities() const;
137
138 const Eigen::VectorXd& GetAccelerations() const;
139
140 const Eigen::VectorXd& GetTorques() const;
141
142 const Eigen::MatrixXd& GetMassMatrix() const;
143
144 const Eigen::MatrixXd& GetCoriolisMatrix() const;
145
146 const Eigen::VectorXd& GetNonlinearEffects() const;
147
148 const Eigen::VectorXd& GetGravity() const;
149
150 const huron::Vector6d& GetSpatialMomentum() const;
151
152 huron::Vector6d GetCentroidalMomentum() const;
153
154 const huron::Matrix6Xd& GetCentroidalMatrix() const;
155
156 void ComputeAll();
157
158 void ForwardKinematics();
159
160 bool is_finalized() const {
161     return is_finalized_;
162 }
163
164 size_t num_positions() const {
165     return num_positions_;
166 }
167
168 size_t num_velocities() const {
169     return num_velocities_;
170 }
171
172 size_t num_joints() const {
173     return joints_.size();
174 }
175
176 size_t num_frames() const {
177     return frames_.size();
178 }
179
180 protected:
181 ModelImplInterface const * GetModelImpl(size_t index) const;
182
183 // template<typename FrameImpl, typename ...Args>
184 // void DoAddFrame(const std::string& name, bool is_user_defined,
185 //                 Args&&... args);
186 // template<typename FrameImpl, typename ...Args>
187 void DoAddFrame(const std::string& name, bool is_user_defined,

```

```

219             Args&&... args) {
220         // Check if the frame name already exists
221         if (frame_name_to_index_.find(name) != frame_name_to_index_.end()) {
222             throw std::runtime_error("Frame name already exists.");
223         }
224         frames_.push_back(FrameImpl::make_shared(
225             frames_.size(), // frame index
226             name, // frame name
227             is_user_defined,
228             weak_from_this(), // model
229             std::forward<Args>(args)...));
230         frame_name_to_index_[name] = frames_.size() - 1;
231     }
232
233     void DoAddFrameFromModelDescription(FrameIndex idx,
234                                         const std::string& name,
235                                         FrameType type);
236
237     size_t default_impl_index_ = 0;
238     std::vector<std::unique_ptr<ModelImplInterface> > impls_;
239     std::vector<std::shared_ptr<Joint> > joints_;
240     Eigen::VectorX<double> states_;
241     size_t num_positions_ = 0;
242     size_t num_velocities_ = 0;
243
244     std::vector<std::shared_ptr<Frame> > frames_;
245     std::unordered_map<std::string, FrameIndex> frame_name_to_index_;
246
247     bool is_constructed_ = false;
248     bool is_finalized_ = false;
249 };
250
251 } // namespace multibody
252 } // namespace huron

```

## 9.18 model\_composite.h

```

1 #pragma once
2
3 #include <vector>
4 #include <memory>
5
6 #include "model.h"
7
8 namespace huron {
9 namespace multibody {
10
11 class ModelComposite final : public class Model {
12 public:
13     ModelComposite();
14
15     void RegisterModel(std::unique_ptr<Model> model);
16
17 private:
18     std::vector<std::unique_ptr<Model> > models_;
19 };
20
21 } // namespace multibody
22 } // namespace huron

```

## 9.19 model\_impl\_factory.h

```

1 #pragma once
2
3 #include <memory>
4
5 #include "model_impl_types.h"
6 #include "model_impl_interface.h"
7 #include "pinocchio_model_impl.h"
8
9 namespace huron {
10 namespace multibody {
11 namespace internal {
12
13 class ModelImplFactory final {
14     friend class multibody::Model;
15 public:
16     ModelImplFactory() = delete;
17     ModelImplFactory(const ModelImplFactory&) = delete;
18     ModelImplFactory& operator=(const ModelImplFactory&) = delete;
19     ~ModelImplFactory() = default;
20 private:

```

```

21 static std::unique_ptr<internal::ModelImplInterface>
22 Create(ModelImplType type) {
23     switch (type) {
24         case ModelImplType::kPinocchio:
25             return std::make_unique<internal::PinocchioModelImpl>();
26         default:
27             throw std::runtime_error("ModelImplType not implemented.");
28     }
29 }
30 };
31
32 } // namespace internal
33 } // namespace multibody
34 } // namespace huron

```

## 9.20 model\_impl\_interface.h

```

1 #pragma once
2
3 #include <vector>
4 #include <memory>
5 #include <string>
6
7 #include "huron/types.h"
8 #include "huron/control_interfaces/joint.h"
9 #include "joint_common.h"
10
11 namespace huron {
12 namespace multibody {
13 namespace internal {
14
15 class ModelImplInterface {
16 public:
17     ModelImplInterface() = default;
18     ModelImplInterface(const ModelImplInterface&) = delete;
19     ModelImplInterface& operator=(const ModelImplInterface&) = delete;
20     virtual ~ModelImplInterface() = default;
21
22     virtual void BuildFromUrdF(const std::string& urdf_path,
23                               JointType root_joint_type);
24
25     virtual const std::vector<std::string>& GetJointNames() const;
26     virtual std::weak_ptr<Joint> GetJoint(const std::string& name) const;
27     virtual std::weak_ptr<Joint> GetJoint(size_t joint_index) const;
28
29     virtual JointType GetJointType(size_t joint_index) const;
30     virtual JointIndex GetJointIndex(const std::string& joint_name) const = 0;
31
32     virtual std::unique_ptr<JointDescription> GetJointDescription(
33         JointIndex joint_index) const;
34     virtual std::unique_ptr<JointDescription> GetJointDescription(
35         const std::string& joint_name) const;
36
37     virtual Eigen::Affine3d
38         GetJointTransformInWorld(size_t joint_index) const;
39
40     virtual FrameIndex GetFrameIndex(
41         const std::string& frame_name) const;
42     virtual const std::string& GetFrameName(FrameIndex frame_index) const;
43     virtual FrameType GetFrameType(FrameIndex frame_index) const;
44     virtual Eigen::Affine3d GetFrameTransform(FrameIndex from_frame,
45                                               FrameIndex to_frame) const;
46     virtual Eigen::Affine3d GetFrameTransformInWorld(FrameIndex frame) const;
47
48     virtual Eigen::Vector3d EvalCenterOfMassPosition();
49     virtual Eigen::Vector3d GetCenterOfMassPosition() const;
50
51     virtual Eigen::VectorXd NeutralConfiguration() const;
52
53     virtual const Eigen::VectorXd& GetAccelerations() const;
54
55     virtual const Eigen::VectorXd& GetTorques() const;
56
57     virtual const Eigen::MatrixXd& GetMassMatrix() const;
58
59     virtual const Eigen::MatrixXd& GetCoriolisMatrix() const;
60
61     virtual const Eigen::VectorXd& GetNonlinearEffects() const;
62
63     virtual const Eigen::VectorXd& GetGravity() const;
64
65     virtual const huron::Vector6d& GetSpatialMomentum() const;
66
67     virtual huron::Vector6d GetCentroidalMomentum() const;
68
69 };
70
71 } // namespace internal
72 } // namespace multibody
73 } // namespace huron

```

```

96  virtual const huron::Matrix6Xd& GetCentroidalMatrix() const;
97
98  virtual void ComputeAll(
99      const Eigen::Ref<const Eigen::VectorXd>& q,
100      const Eigen::Ref<const Eigen::VectorXd>& v);
101
102  virtual void ForwardKinematics(
103      const Eigen::Ref<const Eigen::VectorXd>& q);
104  virtual void ForwardKinematics(
105      const Eigen::Ref<const Eigen::VectorXd>& q,
106      const Eigen::Ref<const Eigen::VectorXd>& v);
107  virtual void ForwardKinematics(
108      const Eigen::Ref<const Eigen::VectorXd>& q,
109      const Eigen::Ref<const Eigen::VectorXd>& v,
110      const Eigen::Ref<const Eigen::VectorXd>& a);
111
112  virtual bool is_built() const;
113
114  virtual size_t num_positions() const;
115  virtual size_t num_velocities() const;
116  virtual size_t num_joints() const;
117  virtual size_t num_frames() const;
118 };
119
120 } // namespace internal
121 } // namespace multibody
122 } // namespace huron

```

## 9.21 model\_impl\_types.h

```

1  #pragma once
2
3  namespace huron {
4  namespace multibody {
5
6  enum class ModelImplType {
7      kPinocchio,
8  };
9
10 } // namespace multibody
11 } // namespace huron

```

## 9.22 pinocchio\_model\_impl.h

```

1  #pragma once
2
3  #include <experimental/propagate_const>
4  #include <vector>
5  #include <string>
6  #include <memory>
7
8  #include "huron/multibody/model_impl_interface.h"
9  #include "huron/multibody/joint_common.h"
10 #include "huron/exceptions/not_implemented_exception.h"
11
12 namespace huron {
13 namespace multibody {
14 namespace internal {
15
16 class PinocchioModelImpl : public ModelImplInterface {
17 public:
18     PinocchioModelImpl();
19     PinocchioModelImpl(const PinocchioModelImpl&) = delete;
20     PinocchioModelImpl& operator=(const PinocchioModelImpl&) = delete;
21     ~PinocchioModelImpl() override;
22
23     static bool IsAvailable() { return true; }
24
25     void BuildFromUrdof(const std::string& urdf_path,
26                        JointType root_joint_type) override;
27
28     const std::vector<std::string>& GetJointNames() const override;
29     std::weak_ptr<Joint> GetJoint(const std::string& name) const override;
30     std::weak_ptr<Joint> GetJoint(size_t joint_index) const override;
31
32     JointType GetJointType(size_t joint_index) const override;
33     JointIndex GetJointIndex(const std::string& joint_name) const override;
34
35     std::unique_ptr<JointDescription> GetJointDescription(
36         JointIndex joint_index) const override;
37     std::unique_ptr<JointDescription> GetJointDescription(
38         const std::string& joint_name) const override;

```

```

39
40 Eigen::Affine3d
41 GetJointTransformInWorld(size_t joint_index) const override;
42
43 FrameIndex GetFrameIndex(const std::string& frame_name) const override;
44 const std::string& GetFrameName(FrameIndex frame_index) const override;
45 FrameType GetFrameType(FrameIndex frame_index) const override;
46 Eigen::Affine3d GetFrameTransform(FrameIndex from_frame,
47                                   FrameIndex to_frame) const override;
48 Eigen::Affine3d
49 GetFrameTransformInWorld(FrameIndex frame) const override;
50
51 Eigen::Vector3d EvalCenterOfMassPosition() override;
52 Eigen::Vector3d GetCenterOfMassPosition() const override;
53
54 Eigen::VectorXd NeutralConfiguration() const override;
55
56 const Eigen::VectorXd& GetAccelerations() const override;
57 const Eigen::VectorXd& GetTorques() const override;
58 const Eigen::MatrixXd& GetMassMatrix() const override;
59 const Eigen::MatrixXd& GetCoriolisMatrix() const override;
60 const Eigen::VectorXd& GetNonlinearEffects() const override;
61 const Eigen::VectorXd& GetGravity() const override;
62 const huron::Vector6d& GetSpatialMomentum() const override;
63 huron::Vector6d GetCentroidalMomentum() const override;
64 const huron::Matrix6Xd& GetCentroidalMatrix() const override;
65
66 void ComputeAll(
67     const Eigen::Ref<const Eigen::VectorXd>& q,
68     const Eigen::Ref<const Eigen::VectorXd>& v) override;
69
70 void ForwardKinematics(
71     const Eigen::Ref<const Eigen::VectorXd>& q) override;
72 void ForwardKinematics(
73     const Eigen::Ref<const Eigen::VectorXd>& q,
74     const Eigen::Ref<const Eigen::VectorXd>& v) override;
75 void ForwardKinematics(
76     const Eigen::Ref<const Eigen::VectorXd>& q,
77     const Eigen::Ref<const Eigen::VectorXd>& v,
78     const Eigen::Ref<const Eigen::VectorXd>& a) override;
79
80 bool is_built() const override { return is_built_; }
81 size_t num_positions() const override { return num_positions_; }
82 size_t num_velocities() const override { return num_velocities_; }
83 size_t num_joints() const override { return num_joints_; }
84 size_t num_frames() const override { return num_frames_; }
85
86 private:
87 struct Impl;
88 std::experimental::propagate_const<std::unique_ptr<Impl>> impl_;
89
90 bool is_built_ = false;
91 size_t num_positions_ = 0;
92 size_t num_velocities_ = 0;
93 size_t num_joints_ = 0;
94 size_t num_frames_ = 0;
95 };
96
97 } // namespace internal
98 } // namespace multibody
99 } // namespace huron

```

## 9.23 force\_torque\_sensor.h

```

1 #pragma once
2
3 #include <memory>
4 #include <utility>
5
6 #include "huron/sensors/force_torque_sensor.h"
7
8 namespace huron {
9 namespace ros2 {
10
11 class HuronNode;
12
13 class ForceTorqueSensor : public huron::ForceTorqueSensor {
14     friend class HuronNode;
15 public:
16     ForceTorqueSensor(bool reverse_wrench_direction,
17                       std::weak_ptr<const multibody::Frame> frame);
18     ForceTorqueSensor(const ForceTorqueSensor&) = delete;
19     ForceTorqueSensor& operator=(const ForceTorqueSensor&) = delete;
20     ~ForceTorqueSensor() override = default;
21

```

```

22 void Initialize() override;
23 void SetUp() override;
24 void Terminate() override;
25
26 protected:
27   Vector6d DoGetWrenchRaw() override;
28
29 private:
30   size_t index_;
31   std::weak_ptr<const HuronNode> node_;
32
33   void SetNode(std::weak_ptr<HuronNode> node) {
34     node_ = std::move(node);
35   }
36
37   void SetIndex(size_t index) {
38     index_ = index;
39   }
40 };
41
42 } // namespace ros2
43 } // namespace huron

```

## 9.24 force\_torque\_sensor.h

```

1 #pragma once
2
3 #include <eigen3/Eigen/Core>
4
5 #include <memory>
6
7 #include "huron/control_interfaces/sensor_with_frame.h"
8 #include "huron/types.h"
9
10 namespace huron {
11
12 class ForceTorqueSensor : public SensorWithFrame {
13 public:
14   ForceTorqueSensor(bool reverse_wrench_direction,
15                     std::weak_ptr<const multibody::Frame> frame);
16   ForceTorqueSensor(bool reverse_wrench_direction,
17                     std::weak_ptr<const multibody::Frame> frame,
18                     std::unique_ptr<Configuration> config);
19   ForceTorqueSensor(const ForceTorqueSensor&) = delete;
20   ForceTorqueSensor& operator=(const ForceTorqueSensor&) = delete;
21   ~ForceTorqueSensor() override = default;
22
23   void RequestStateUpdate() override;
24
25   void GetNewState(Eigen::Ref<Eigen::MatrixXd> new_state) const override;
26   Eigen::VectorXd GetValue() const override;
27
28 protected:
29   virtual Vector6d DoGetWrenchRaw() = 0;
30
31   bool reverse_wrench_direction_;
32
33 private:
34   huron::Vector6d wrench_;
35 };
36
37 } // namespace huron

```

## 9.25 huron.h

```

1 #pragma once
2
3 #include "huron_node.h"
4
5 #include <string>
6 #include <vector>
7 #include <memory>
8
9 #include "huron/control_interfaces/legged_robot.h"
10
11 namespace huron {
12 namespace ros2 {
13
14 class Huron : public huron::LeggedRobot {
15 public:
16   Huron(std::shared_ptr<HuronNode> node,
17         std::unique_ptr<huron::RobotConfiguration> config);

```



```

18  explicit Huron(std::shared_ptr<HuronNode> node);
19
20  Huron(const Huron&) = delete;
21  Huron& operator=(const Huron&) = delete;
22  ~Huron() override = default;
23
24  // GenericComponent interface
25  void Initialize() override;
26  void SetUp() override;
27  void Terminate() override;
28
29  // ROS-specific
30  void Loop();
31
32 private:
33  std::shared_ptr<HuronNode> node_;
34 };
35
36 } // namespace ros2
37 } // namespace huron

```

## 9.26 huron\_node.h

```

1  #pragma once
2
3  #include <eigen3/Eigen/Core>
4
5  #include <memory>
6  #include <vector>
7  #include <string>
8
9  #include <rclcpp/rclcpp.hpp>
10 #include <sensor_msgs/msg/joint_state.hpp>
11 #include <std_msgs/msg/float64_multi_array.hpp>
12 #include <geometry_msgs/msg/wrench_stamped.hpp>
13 #include <nav_msgs/msg/odometry.hpp>
14
15 #include "huron/types.h"
16
17 namespace huron {
18 namespace ros2 {
19
20 class ForceTorqueSensor;
21 class JointGroupController;
22 class JointStateProvider;
23
24 class HuronNode : public rclcpp::Node {
25 public:
26  HuronNode();
27  ~HuronNode() override = default;
28
29  void JointStateCallback(
30      std::shared_ptr<const sensor_msgs::msg::JointState> msg);
31  void OdomCallback(
32      std::shared_ptr<const nav_msgs::msg::Odometry> msg);
33  void WrenchStampedCallback(
34      size_t idx,
35      std::shared_ptr<const geometry_msgs::msg::WrenchStamped> msg);
36  void PublishFloat64MultiArray(size_t idx, const std::vector<double>& values);
37
38  const huron::Vector6d& GetWrench(size_t idx) const {
39      return wrenches_[idx];
40  }
41
42  void AddJointStateProvider(
43      std::shared_ptr<JointStateProvider> jsp,
44      const std::string& topic,
45      size_t nq, size_t nv,
46      bool is_odom = false);
47  void AddForceTorqueSensor(
48      std::shared_ptr<ForceTorqueSensor> ft_sensor,
49      const std::string& topic);
50  void AddJointGroupController(
51      std::shared_ptr<JointGroupController> jgc,
52      const std::string& topic);
53
54  void Finalize();
55
56  Eigen::VectorXd GetJointState(size_t id_q, size_t dim_q,
57                                size_t id_v, size_t dim_v) const {
58      Eigen::VectorXd state(dim_q + dim_v);
59      state.segment(0, dim_q) = joint_state_.segment(id_q, dim_q);
60      state.segment(dim_q, dim_v) = joint_state_.segment(nq + id_v,
61                                                           dim_v);
62  }
63
64 }
65
66 } // namespace ros2
67 } // namespace huron

```

```

71     return state;
72 }
73
74 private:
75 bool finalized_ = false;
76 size_t nq_ = 0;
77 size_t nv_ = 0;
78
79 rclcpp::Subscription<nav_msgs::msg::Odometry>::SharedPtr odom_sub_;
80 rclcpp::Subscription<sensor_msgs::msg::JointState>::SharedPtr
81   joint_state_sub_;
82 std::vector<rclcpp::Publisher<std_msgs::msg::Float64MultiArray>::SharedPtr>
83   float64_multi_array_pubs_;
84 std::vector<rclcpp::Subscription<geometry_msgs::msg::WrenchStamped>
85   ::SharedPtr> wrench_stamped_subs_;
86
87 Eigen::VectorXd joint_state_;
88 std::vector<huron::Vector6d> wrenches_;
89 };
90
91 // namespace ros2
92 // namespace huron

```

## 9.27 joint\_group\_controller.h

```

1 #pragma once
2
3 #include <vector>
4 #include <memory>
5 #include <utility>
6
7 #include "huron/control_interfaces/moving_interface.h"
8
9 namespace huron {
10 namespace ros2 {
11
12 class HuronNode;
13
14 class JointGroupController : public huron::MovingInterface {
15   friend class HuronNode;
16 public:
17   explicit JointGroupController(size_t dim);
18   JointGroupController(const JointGroupController&) = delete;
19   JointGroupController& operator=(const JointGroupController&) = delete;
20   ~JointGroupController() override = default;
21
22   bool Move(const std::vector<double>& values) override;
23   bool Move(const Eigen::VectorXd& values) override;
24   bool Stop() override;
25
26 private:
27   std::weak_ptr<HuronNode> node_;
28   size_t dim_;
29   size_t pub_idx_;
30
31   void SetNode(std::weak_ptr<HuronNode> node) {
32     node_ = std::move(node);
33   }
34
35   void SetPubIdx(size_t pub_idx) {
36     pub_idx_ = pub_idx;
37   }
38 };
39
40 // namespace ros2
41 // namespace huron

```

## 9.28 joint\_state\_provider.h

```

1 #pragma once
2
3 #include <memory>
4 #include <utility>
5
6 #include "huron/control_interfaces/state_provider.h"
7
8 namespace huron {
9 namespace ros2 {
10
11 class HuronNode;
12
13 class JointStateProvider : public huron::StateProvider {

```

```

14 friend class HuronNode;
15 public:
16   JointStateProvider(size_t id_q, size_t nq, size_t id_v, size_t nv);
17
18   void RequestStateUpdate() override;
19   void GetNewState(Eigen::Ref<Eigen::MatrixXd> new_state) const override;
20
21 private:
22   std::weak_ptr<HuronNode> node_;
23   size_t nq_;
24   size_t nv_;
25   size_t id_q_;
26   size_t id_v_;
27
28   void SetNode(std::weak_ptr<HuronNode> node) {
29     node_ = std::move(node);
30   }
31 };
32
33 } // namespace ros2
34 } // namespace huron

```

## 9.29 actuator.h

```

1 #pragma once
2
3 #include <vector>
4 #include <set>
5 #include <string>
6 #include <utility>
7 #include <memory>
8
9 #include "huron/control_interfaces/moving_interface.h"
10 #include "huron/control_interfaces/generic_component.h"
11
12 namespace huron {
13
14 class ActuatorConfiguration : public Configuration {
15 private:
16   static const inline std::set<std::string> kActuatorValidKeys{};
17
18 public:
19   ActuatorConfiguration(ConfigMap config_map,
20                         std::set<std::string> valid_keys)
21     : Configuration(config_map, [&valid_keys]() {
22         std::set<std::string> tmp(kActuatorValidKeys);
23         tmp.merge(valid_keys);
24         return tmp;
25       }) {}
26
27   explicit ActuatorConfiguration(ConfigMap config_map)
28     : ActuatorConfiguration(config_map, {}) {}
29
30   ActuatorConfiguration()
31     : ActuatorConfiguration({}, {}) {}
32 };
33
34 class Actuator : public GenericComponent, public MovingInterface {
35 public:
36   Actuator(size_t dim, std::unique_ptr<ActuatorConfiguration> config)
37     : GenericComponent(std::move(config)), MovingInterface(dim) {}
38   explicit Actuator(size_t dim)
39     : Actuator(dim, std::make_unique<ActuatorConfiguration>()) {}
40   Actuator(const Actuator&) = delete;
41   Actuator& operator=(const Actuator&) = delete;
42   ~Actuator() override = default;
43 };
44
45 } // namespace huron

```

## 9.30 configuration.h

```

1 #pragma once
2
3 #include <string>
4 #include <unordered_map>
5 #include <set>
6 #include <any>
7
8 #include "huron/exceptions/not_implemented_exception.h"
9
10 namespace huron {

```

```

11
12 typedef std::unordered_map<std::string, std::any> ConfigMap;
13
14 class Configuration {
15 protected:
16     const std::set<std::string> valid_keys_;
17     ConfigMap config_map_;
18
19     bool ValidateKey(std::string config_key) {
20         return valid_keys_.count(config_key);
21     }
22     ConfigMap ValidateMap(ConfigMap config_map);
23     virtual std::any GetFromComponent(std::string config_key) {
24         throw NotImplementedException(__func__);
25     }
26
27 public:
28     Configuration(ConfigMap config_map, std::set<std::string> valid_keys);
29     explicit Configuration(ConfigMap config_map);
30     Configuration(const Configuration&) = delete;
31     Configuration& operator=(const Configuration&) = delete;
32     virtual ~Configuration() = default;
33
34     std::any Get(std::string config_key, bool renew = false);
35     bool Set(std::string config_key, std::any config_value);
36     bool Set(ConfigMap config_map);
37 };
38
39 // namespace huron

```

## 9.31 constant\_state\_provider.h

```

1 #pragma once
2
3 #include "state_provider.h"
4
5 namespace huron {
6
7 class ConstantStateProvider : public StateProvider {
8 public:
9     explicit ConstantStateProvider(const Eigen::MatrixXd& state)
10         : StateProvider(state.rows(), state.cols()),
11           state_(state) {}
12     ConstantStateProvider(const ConstantStateProvider&) = delete;
13     ConstantStateProvider& operator=(const ConstantStateProvider&) = delete;
14     ~ConstantStateProvider() override = default;
15
16     void RequestStateUpdate() override {}
17     void GetNewState(Eigen::Ref<Eigen::MatrixXd> new_state) const override {
18         new_state = state_;
19     }
20
21     void SetState(const Eigen::MatrixXd& state) {
22         assert(state.rows() == dim()[0] && state.cols() == dim()[1]);
23         state_ = state;
24     }
25
26 private:
27     Eigen::MatrixXd state_;
28 };
29
30 // namespace huron

```

## 9.32 encoder.h

```

1 #pragma once
2
3 #include <set>
4 #include <string>
5 #include <utility>
6 #include <memory>
7
8 #include "huron/control_interfaces/sensor.h"
9
10 namespace huron {
11
12 class EncoderConfiguration : public Configuration {
13 private:
14     static const inline std::set<std::string> kEncoderValidKeys{};
15
16 public:
17     EncoderConfiguration(ConfigMap config_map,

```

```

18         std::set<std::string> valid_keys)
19     : Configuration(config_map, [&valid_keys]() {
20         std::set<std::string> tmp(kEncoderValidKeys);
21         tmp.merge(valid_keys);
22         return tmp;
23     }) {}
24
25     explicit EncoderConfiguration(ConfigMap config_map)
26     : EncoderConfiguration(config_map, {}) {}
27
28     EncoderConfiguration()
29     : EncoderConfiguration({}, {}) {}
30 };
31
32 class Encoder : public Sensor {
33 public:
34     Encoder(double gear_ratio, std::unique_ptr<EncoderConfiguration> config)
35     : Sensor(2, 1, std::move(config)), gear_ratio_(gear_ratio) {}
36     explicit Encoder(double gear_ratio)
37     : Encoder(gear_ratio, std::make_unique<EncoderConfiguration>()) {}
38     explicit Encoder(std::unique_ptr<EncoderConfiguration> config)
39     : Encoder(1.0, std::move(config)) {}
40     Encoder() : Encoder(1.0) {}
41     Encoder(const Encoder&) = delete;
42     Encoder& operator=(const Encoder&) = delete;
43     virtual ~Encoder() = default;
44
45     void GetNewState(Eigen::Ref<Eigen::MatrixXd> new_state) const override {
46         new_state = Eigen::Vector2d(GetPosition(), GetVelocity());
47     }
48
49     virtual double GetPosition() const = 0;
50     virtual double GetVelocity() const = 0;
51
52     virtual void Reset() = 0;
53
54 protected:
55     double gear_ratio_;
56 };
57
58 } // namespace huron

```

## 9.33 generic\_component.h

```

1 #pragma once
2
3 #include <memory>
4 #include <unordered_map>
5 #include <string>
6 #include <utility>
7
8 #include "huron/control_interfaces/configuration.h"
9
10 namespace huron {
11
12 class GenericComponent {
13 protected:
14     std::unique_ptr<Configuration> config_;
15
16     virtual void ConfigureKey(std::string config_key, std::any config_value) {}
17
18     virtual void ConfigureMap(const ConfigMap& config_map) {
19         for (auto& pair : config_map) {
20             ConfigureKey(pair.first, pair.second);
21         }
22     }
23
24 public:
25     explicit GenericComponent(std::unique_ptr<Configuration> config)
26     : config_(std::move(config)) {}
27     GenericComponent()
28     : GenericComponent(std::make_unique<Configuration>(ConfigMap())) {}
29     GenericComponent(const GenericComponent&) = delete;
30     GenericComponent& operator=(const GenericComponent&) = delete;
31     virtual ~GenericComponent() = default;
32
33     void Configure(std::string config_key, std::any config_value) {
34         if (config_>Set(config_key, config_value)) {
35             return ConfigureKey(config_key, config_value);
36         }
37     }
38
39     void Configure(ConfigMap config) {
40         if (config_>Set(config)) {
41             return ConfigureMap(config);
42         }
43     }
44 }

```



```

84     }
85
86     size_t nv() const {
87         return num_velocities();
88     }
89
90 protected:
91     std::unique_ptr<JointDescription> jd_;
92     Eigen::VectorXd positions_;
93     Eigen::VectorXd velocities_;
94     size_t id_q_;
95     size_t id_v_;
96
97     std::shared_ptr<StateProvider> state_provider_;
98 };
99
100 } // namespace huron

```

## 9.35 legged\_robot.h

```

1 #pragma once
2
3 #include <eigen3/Eigen/Core>
4
5 #include <memory>
6
7 #include "robot.h"
8 #include "huron/locomotion/zero_moment_point.h"
9
10 namespace huron {
11
12 class LeggedRobot : public Robot {
13 public:
14     explicit LeggedRobot(std::unique_ptr<RobotConfiguration> config);
15     LeggedRobot();
16     LeggedRobot(const LeggedRobot&) = delete;
17     LeggedRobot& operator=(const LeggedRobot&) = delete;
18     ~LeggedRobot() override = default;
19
20     void InitializeZmp(std::shared_ptr<ZeroMomentPoint> zmp);
21     Eigen::Vector2d EvalZeroMomentPoint();
22
23 private:
24     std::shared_ptr<ZeroMomentPoint> zmp_;
25 };
26
27 } // namespace huron

```

## 9.36 limb.h

```

1 #pragma once
2
3 #include <vector>
4
5 #include "joint.h"
6 #include "moving_group_component.h"
7
8 namespace huron {
9
10 class Limb : public MovingGroupComponent {
11 public:
12     void Init(std::vector<Joint> joints);
13     void AddJoint(Joint& joint);
14
15 private:
16     std::vector<Joint> joints_;
17 };
18
19 } // namespace huron

```

## 9.37 motor.h

```

1 #pragma once
2
3 #include <vector>
4 #include <set>
5 #include <string>
6 #include <utility>
7 #include <memory>

```

```

8
9 #include "huron/control_interfaces/actuator.h"
10
11 namespace huron {
12
13 class MotorConfiguration : public ActuatorConfiguration {
14 private:
15     static const inline std::set<std::string> kMotorValidKeys{};
16
17 public:
18     MotorConfiguration(ConfigMap config_map,
19                         std::set<std::string> valid_keys)
20         : ActuatorConfiguration(config_map, [&valid_keys]() {
21             std::set<std::string> tmp(kMotorValidKeys);
22             tmp.merge(valid_keys);
23             return tmp;
24         }) {}
25
26     explicit MotorConfiguration(ConfigMap config_map)
27         : MotorConfiguration(config_map, {}) {}
28
29     MotorConfiguration()
30         : MotorConfiguration({}, {}) {}
31 };
32
33 class Motor : public Actuator {
34 public:
35     explicit Motor(std::unique_ptr<MotorConfiguration> config,
36                   double gear_ratio = 1.0)
37         : Actuator(1, std::move(config)) {}
38     explicit Motor(double gear_ratio)
39         : Motor(std::make_unique<MotorConfiguration>(), gear_ratio) {}
40     Motor() : Motor(1.0) {}
41     Motor(const Motor&) = delete;
42     Motor& operator=(const Motor&) = delete;
43     ~Motor() override = default;
44
45     virtual bool Move(double value) = 0;
46
47 private:
48     double gear_ratio_;
49 };
50
51 } // namespace huron

```

## 9.38 moving\_group.h

```

1 #pragma once
2
3 #include <eigen3/Eigen/Dense>
4
5 #include <memory>
6 #include <utility>
7 #include <vector>
8
9 #include "moving_interface.h"
10
11 namespace huron {
12
13 class MovingGroup : public MovingInterface {
14 public:
15     MovingGroup();
16     MovingGroup(const MovingGroup&) = delete;
17     MovingGroup& operator=(const MovingGroup&) = delete;
18     ~MovingGroup() override = default;
19
20     virtual void AddToGroup(std::shared_ptr<MovingInterface> component);
21
22     bool Move(const std::vector<double>& values) override;
23
24     bool Move(const Eigen::VectorXd& values) override;
25
26     bool Stop() override;
27
28 protected:
29     std::vector<std::shared_ptr<MovingInterface>> moving_components_;
30     std::vector<size_t> moving_interface_dims_;
31 };
32
33 } // namespace huron

```



## 9.39 moving\_interface.h

```

1 #pragma once
2
3 #include <eigen3/Eigen/Dense>
4
5 #include <vector>
6 #include <memory>
7 #include <utility>
8
9 namespace huron {
10
11     class MovingInterface {
12     public:
13         explicit MovingInterface(size_t dim) : dim_(dim) {}
14         MovingInterface(const MovingInterface&) = delete;
15         MovingInterface& operator=(const MovingInterface&) = delete;
16         virtual ~MovingInterface() = default;
17
18         virtual bool Move(const std::vector<double>& values) = 0;
19
20         virtual bool Move(const Eigen::VectorXd& values) = 0;
21
22         virtual bool Stop() = 0;
23
24         size_t dim() const { return dim_; }
25
26     protected:
27         size_t dim_;
28     };
29
30 } // namespace huron

```

## 9.40 position\_motor.h

```

1 #pragma once
2
3 #include <set>
4 #include <string>
5 #include <utility>
6 #include <memory>
7
8 #include "huron/control_interfaces/motor.h"
9
10 namespace huron {
11
12     class PositionMotorConfiguration : public MotorConfiguration {
13     public:
14         PositionMotorConfiguration(ConfigMap config_map,
15                                     std::set<std::string> valid_keys)
16             : MotorConfiguration(config_map,
17                                   [&valid_keys]() {
18                                       std::set<std::string> tmp(kPositionMotorValidKeys);
19                                       tmp.merge(valid_keys);
20                                       return tmp;
21                                   }) {}
22
23         PositionMotorConfiguration()
24             : PositionMotorConfiguration({}, {}) {}
25
26     private:
27         static const inline std::set<std::string> kPositionMotorValidKeys{};
28     };
29
30     class PositionMotor : public Motor {
31     public:
32         explicit PositionMotor(std::unique_ptr<PositionMotorConfiguration> config,
33                                 double gear_ratio)
34             : Motor(std::move(config), gear_ratio) {}
35         explicit PositionMotor(double gear_ratio)
36             : Motor(gear_ratio) {}
37
38         PositionMotor() : Motor() {}
39         PositionMotor(const PositionMotor&) = delete;
40         PositionMotor& operator=(const PositionMotor&) = delete;
41         virtual ~PositionMotor() = default;
42     };
43
44 } // namespace huron

```

## 9.41 robot.h

```

1 #pragma once

```

```

2
3 #include <vector>
4 #include <set>
5 #include <string>
6 #include <utility>
7 #include <memory>
8
9 #include "huron/control_interfaces/configuration.h"
10 #include "huron/control_interfaces/generic_component.h"
11 #include "huron/control_interfaces/moving_group.h"
12 #include "huron/control_interfaces/joint.h"
13 #include "huron/multibody/model.h"
14 #include "huron/multibody/joint_common.h"
15
16 namespace huron {
17
18 class RobotConfiguration : public Configuration {
19 private:
20     static const inline std::set<std::string> kRobotValidKeys{};
21
22 public:
23     RobotConfiguration(ConfigMap config_map,
24                        std::set<std::string> valid_keys)
25         : Configuration(config_map, [&valid_keys]() {
26             std::set<std::string> tmp(kRobotValidKeys);
27             tmp.merge(valid_keys);
28             return tmp;
29         }) {}
30
31     explicit RobotConfiguration(ConfigMap config_map)
32         : RobotConfiguration(config_map, {}) {}
33
34     RobotConfiguration()
35         : RobotConfiguration({}, {}) {}
36 };
37
38 class Robot : public MovingGroup, public GenericComponent {
39     using Model = multibody::Model;
40
41 public:
42     explicit Robot(std::unique_ptr<RobotConfiguration> config);
43     Robot();
44     Robot(const Robot&) = delete;
45     Robot& operator=(const Robot&) = delete;
46     ~Robot() override = default;
47
48     Model* const GetModel() { return model_.get(); }
49
50     void RegisterStateProvider(std::shared_ptr<StateProvider> state_provider,
51                              bool is_joint_state_provider = false);
52
53     void UpdateAllStates();
54
55     void UpdateJointStates();
56
57     const Eigen::VectorBlock<const Eigen::VectorXd> GetJointPositions() const;
58
59     const Eigen::VectorBlock<const Eigen::VectorXd> GetJointVelocities() const;
60
61 protected:
62     Robot(std::unique_ptr<RobotConfiguration> config,
63          std::shared_ptr<Model> model);
64     explicit Robot(std::shared_ptr<Model> model);
65
66     std::shared_ptr<Model> model_;
67     std::vector<std::shared_ptr<StateProvider>> non_joint_state_providers_;
68     std::vector<std::shared_ptr<StateProvider>> joint_state_providers_;
69 };
70
71 } // namespace huron

```

## 9.42 rotary\_encoder.h

```

1 #pragma once
2
3 #include <cmath>
4 #include <set>
5 #include <string>
6 #include <utility>
7 #include <memory>
8
9 #include "encoder.h"
10
11 namespace huron {
12

```

```

13 class RotaryEncoderConfiguration : public EncoderConfiguration {
14 public:
15     RotaryEncoderConfiguration(ConfigMap config_map,
16                               std::set<std::string> valid_keys)
17         : EncoderConfiguration(config_map,
18                               [&valid_keys]() {
19                                   std::set<std::string> tmp(kRotEncValidKeys);
20                                   tmp.merge(valid_keys);
21                                   return tmp;
22                               }) {}
23
24     explicit RotaryEncoderConfiguration(double cpr)
25         : RotaryEncoderConfiguration(
26             ConfigMap({{"cpr", cpr}}), {}) {}
27
28 private:
29     static const inline std::set<std::string> kRotEncValidKeys{"cpr"};
30 };
31
32 class RotaryEncoder : public Encoder {
33 public:
34     RotaryEncoder(double gear_ratio,
35                  std::unique_ptr<RotaryEncoderConfiguration> config)
36         : Encoder(gear_ratio, std::move(config)) {
37         cpr_ = std::any_cast<double>(config.get()->Get("cpr"));
38     }
39     RotaryEncoder(double gear_ratio, double cpr)
40         : RotaryEncoder(gear_ratio,
41                        std::make_unique<RotaryEncoderConfiguration>(cpr)) {}
42
43     explicit RotaryEncoder(double cpr)
44         : RotaryEncoder(1.0, cpr) {}
45     RotaryEncoder(const RotaryEncoder&) = delete;
46     RotaryEncoder& operator=(const RotaryEncoder&) = delete;
47     ~RotaryEncoder() override = default;
48
49     void RequestStateUpdate() final {
50         prev_count_ = count_;
51         prev_velocity_ = velocity_;
52         DoUpdateState();
53     }
54
55     double GetCount() const {
56         return count_;
57     }
58
59     double GetVelocityCount() const {
60         return velocity_;
61     }
62
63     double GetPrevCount() const {
64         return prev_count_;
65     }
66
67     double GetCPR() const {
68         return cpr_;
69     }
70
71     double GetPosition() const override {
72         return count_ / cpr_ * 2.0 * M_PI / gear_ratio_;
73     }
74
75     double GetAngleDegree() const {
76         return count_ / cpr_ * 360.0 / gear_ratio_;
77     }
78
79     double GetVelocity() const override {
80         return velocity_ / cpr_ * 2 * M_PI / gear_ratio_;
81     }
82
83     double GetVelocityDegree() const {
84         return velocity_ / cpr_ * 360.0 / gear_ratio_;
85     }
86
87     void Reset() override {
88         count_ = 0.0;
89         prev_count_ = 0.0;
90     }
91
92 protected:
93     virtual void DoUpdateState() = 0;
94
95     double velocity_ = 0.0;
96     double prev_velocity_ = 0.0;
97     double count_ = 0.0;
98     double prev_count_ = 0.0;
99     double cpr_;
100 };

```

```

149
150 } // namespace huron

```

## 9.43 sensor.h

```

1 #pragma once
2
3 #include <eigen3/Eigen/Core>
4
5 #include <memory>
6
7 #include "huron/control_interfaces/generic_component.h"
8 #include "huron/control_interfaces/state_provider.h"
9
10 namespace huron {
11
12 class Sensor : public GenericComponent, public StateProvider {
13 public:
14     Sensor(const Eigen::Vector2i& dim,
15            std::unique_ptr<Configuration> config);
16     explicit Sensor(const Eigen::Vector2i& dim);
17     Sensor(int rows, int cols,
18            std::unique_ptr<Configuration> config);
19     Sensor(int rows, int cols);
20     Sensor(const Sensor&) = delete;
21     Sensor& operator=(const Sensor&) = delete;
22     virtual ~Sensor() = default;
23
24
27     virtual Eigen::VectorXd GetValue() const;
28     virtual Eigen::VectorXd ReloadAndGetValue();
29 };
30
31 } // namespace huron

```

## 9.44 sensor\_with\_frame.h

```

1 #pragma once
2
3 #include <eigen3/Eigen/Core>
4
5 #include <memory>
6
7 #include "huron/control_interfaces/sensor.h"
8 #include "huron/multibody/frame.h"
9
10 namespace huron {
11
12 class SensorWithFrame : public Sensor {
13     using Frame = multibody::Frame;
14
15 public:
16     SensorWithFrame(const Eigen::Vector2i& dim,
17                    std::weak_ptr<const Frame> frame);
18     SensorWithFrame(const Eigen::Vector2i& dim,
19                    std::weak_ptr<const Frame> frame,
20                    std::unique_ptr<Configuration> config);
21     SensorWithFrame(int rows, int cols,
22                    std::weak_ptr<const Frame> frame);
23     SensorWithFrame(int rows, int cols,
24                    std::weak_ptr<const Frame> frame,
25                    std::unique_ptr<Configuration> config);
26     SensorWithFrame(const SensorWithFrame&) = delete;
27     SensorWithFrame& operator=(const SensorWithFrame&) = delete;
28     ~SensorWithFrame() override = default;
29
30
33     std::weak_ptr<const Frame> GetSensorFrame() const {
34         return frame_;
35     }
36
37 private:
38     std::weak_ptr<const Frame> frame_;
39 };
40
41 } // namespace huron

```

## 9.45 state\_provider.h

```

1 #pragma once
2

```

```

3 #include <eigen3/Eigen/Dense>
4
5 namespace huron {
6
7 class StateProvider {
8 public:
9     explicit StateProvider(const Eigen::Vector2i& dim)
10         : dim_(dim) {}
11     StateProvider(int rows, int cols)
12         : dim_(rows, cols) {}
13     StateProvider(const StateProvider&) = delete;
14     StateProvider& operator=(const StateProvider&) = delete;
15     virtual ~StateProvider() = default;
16
17     virtual void RequestStateUpdate() = 0;
18     virtual void GetNewState(Eigen::Ref<Eigen::MatrixXd> new_state) const = 0;
19
20     const Eigen::Vector2i& dim() const { return dim_; }
21
22 private:
23     const Eigen::Vector2i dim_;
24 };
25
26 } // namespace huron

```

## 9.46 torque\_motor.h

```

1 #pragma once
2
3 #include <set>
4 #include <string>
5 #include <utility>
6 #include <memory>
7
8 #include "huron/control_interfaces/motor.h"
9
10 namespace huron {
11
12 class TorqueMotorConfiguration : public MotorConfiguration {
13 public:
14     TorqueMotorConfiguration(ConfigMap config_map,
15                             std::set<std::string> valid_keys)
16         : MotorConfiguration(config_map,
17                             [&valid_keys]() {
18                                 std::set<std::string> tmp(kTorqueMotorValidKeys);
19                                 tmp.merge(valid_keys);
20                                 return tmp;
21                             }) {}
22
23     TorqueMotorConfiguration()
24         : TorqueMotorConfiguration({}, {}) {}
25
26 private:
27     static const inline std::set<std::string> kTorqueMotorValidKeys{};
28 };
29
30 class TorqueMotor : public Motor {
31 public:
32     TorqueMotor(std::unique_ptr<TorqueMotorConfiguration> config,
33                double gear_ratio)
34         : Motor(std::move(config), gear_ratio) {}
35     explicit TorqueMotor(double gear_ratio)
36         : Motor(gear_ratio) {}
37     TorqueMotor() : Motor() {}
38     TorqueMotor(const TorqueMotor&) = delete;
39     TorqueMotor& operator=(const TorqueMotor&) = delete;
40     ~TorqueMotor() override = default;
41 };
42
43 } // namespace huron

```

## 9.47 velocity\_motor.h

```

1 #pragma once
2
3 #include <set>
4 #include <string>
5 #include <utility>
6 #include <memory>
7
8 #include "huron/control_interfaces/motor.h"
9

```

```

10 namespace huron {
11
12 class VelocityMotorConfiguration : public MotorConfiguration {
13 public:
14     VelocityMotorConfiguration(ConfigMap config_map,
15                               std::set<std::string> valid_keys)
16         : MotorConfiguration(config_map,
17                               [&valid_keys]() {
18                                   std::set<std::string> tmp(kVelocityMotorValidKeys);
19                                   tmp.merge(valid_keys);
20                                   return tmp;
21                               }) {}
22
23     VelocityMotorConfiguration()
24         : VelocityMotorConfiguration({}, {}) {}
25
26 private:
27     static const inline std::set<std::string> kVelocityMotorValidKeys{};
28 };
29
30 class VelocityMotor : public Motor {
31 public:
32     VelocityMotor(std::unique_ptr<VelocityMotorConfiguration> config,
33                  double gear_ratio)
34         : Motor(std::move(config), gear_ratio) {}
35     explicit VelocityMotor(double gear_ratio)
36         : Motor(gear_ratio) {}
37     VelocityMotor() : Motor() {}
38     VelocityMotor(const VelocityMotor&) = delete;
39     VelocityMotor& operator=(const VelocityMotor&) = delete;
40     ~VelocityMotor() override = default;
41 };
42
43 } // namespace huron

```

## 9.48 invalid\_configuration\_exception.h

```

1
4 #pragma once
5
6 #include <stdexcept>
7 #include <string>
8
9 namespace huron {
10
11 class InvalidConfigurationException : public std::logic_error {
12 private:
13     std::string _text;
14
15     InvalidConfigurationException(const char* message, const char* function)
16         : std::logic_error("Invalid Configuration provided.") {
17         _text = message;
18         _text += " : ";
19         _text += function;
20     }
21
22 public:
23     InvalidConfigurationException()
24         : InvalidConfigurationException(
25             "Invalid Configuration provided.", __FUNCTION__) {}
26
27     explicit InvalidConfigurationException(const char* message)
28         : InvalidConfigurationException(message, __FUNCTION__) {}
29
30     virtual const char *what() const throw() {
31         return _text.c_str();
32     }
33 };
34
35 } // namespace huron

```

## 9.49 not\_implemented\_exception.h

```

1
4 #pragma once
5
6 #include <stdexcept>
7 #include <string>
8
9 namespace huron {
10
11 class NotImplementedException : public std::logic_error {

```

```

12 private:
13     std::string _text;
14
15     NotImplementedException(const char* message, const char* function)
16         : std::logic_error("Not Implemented") {
17         _text = message;
18         _text += " : ";
19         _text += function;
20     }
21
22 public:
23     NotImplementedException()
24         : NotImplementedException("Not Implemented", __FUNCTION__) {}
25
26     explicit NotImplementedException(const char* function)
27         : NotImplementedException("Not Implemented", function) {}
28
29     virtual const char *what() const throw() {
30         return _text.c_str();
31     }
32 };
33
34 } // namespace huron

```

## 9.50 zero\_moment\_point.h

```

1 #pragma once
2
3 #include <Eigen3/Eigen/Dense>
4
5 #include <memory>
6
7 #include "huron/sensors/force_torque_sensor.h"
8 #include "huron/sensors/force_sensing_resistor_array.h"
9 #include "huron/multibody/logical_frame.h"
10
11 namespace huron {
12
13 class ZeroMomentPoint {
14 public:
15     ZeroMomentPoint(std::weak_ptr<const multibody::Frame> zmp_frame,
16                     double normal_force_threshold);
17     ZeroMomentPoint(const ZeroMomentPoint&) = delete;
18     ZeroMomentPoint& operator=(const ZeroMomentPoint&) = delete;
19     virtual ~ZeroMomentPoint() = default;
20
21     virtual Eigen::Vector2d Eval(double& fz) = 0;
22     Eigen::Vector2d Eval() {
23         double fz;
24         return Eval(fz);
25     }
26
27     Eigen::Affine3d ZmpToWorld(const Eigen::Vector2d& zmp) const;
28
29 protected:
30     std::weak_ptr<const multibody::Frame> zmp_frame_;
31     double normal_force_threshold_;
32 };
33
34 } // namespace huron

```

## 9.51 zero\_moment\_point\_fsr\_array.h

```

1 #pragma once
2
3 #include <memory>
4
5 #include "huron/locomotion/zero_moment_point.h"
6
7 namespace huron {
8
9 class ZeroMomentPointFSRArray : public ZeroMomentPoint {
10 public:
11     ZeroMomentPointFSRArray(
12         std::weak_ptr<const multibody::Frame> zmp_frame,
13         double normal_force_threshold,
14         std::shared_ptr<ForceSensingResistorArray> fsr_array);
15     ZeroMomentPointFSRArray(const ZeroMomentPointFSRArray&) = delete;
16     ZeroMomentPointFSRArray& operator=(const ZeroMomentPointFSRArray&) = delete;
17     ~ZeroMomentPointFSRArray() override = default;
18
19     Eigen::Vector2d Eval(double& fz) override;

```

```

20
21 private:
22     std::shared_ptr<ForceSensingResistorArray> fsr_array_;
23 };
24
25 } // namespace huron

```

## 9.52 zero\_moment\_point\_ft\_sensor.h

```

1 #pragma once
2
3 #include <memory>
4 #include <vector>
5
6 #include "huron/locomotion/zero_moment_point.h"
7
8 namespace huron {
9
10 class ZeroMomentPointFTSensor : public ZeroMomentPoint {
11 public:
12     ZeroMomentPointFTSensor(
13         std::weak_ptr<const multibody::Frame> zmp_frame,
14         double normal_force_threshold,
15         const std::vector<std::shared_ptr<ForceTorqueSensor>& ft_sensors);
16     ZeroMomentPointFTSensor(const ZeroMomentPointFTSensor&) = delete;
17     ZeroMomentPointFTSensor& operator=(const ZeroMomentPointFTSensor&) = delete;
18     ~ZeroMomentPointFTSensor() override = default;
19
20     Eigen::Vector2d Eval(double& fz) override;
21
22 private:
23     std::vector<std::shared_ptr<ForceTorqueSensor>& ft_sensors_;
24 };
25
26 } // namespace huron

```

## 9.53 zero\_moment\_point\_total.h

```

1 #pragma once
2
3 #include <memory>
4 #include <vector>
5
6 #include "huron/locomotion/zero_moment_point.h"
7
8 namespace huron {
9
10 class ZeroMomentPointTotal : public ZeroMomentPoint {
11 public:
12     ZeroMomentPointTotal(
13         std::weak_ptr<const multibody::Frame> zmp_frame,
14         const std::vector<std::shared_ptr<ZeroMomentPoint>& zmp_vector);
15     ZeroMomentPointTotal(const ZeroMomentPointTotal&) = delete;
16     ZeroMomentPointTotal& operator=(const ZeroMomentPointTotal&) = delete;
17     ~ZeroMomentPointTotal() override = default;
18
19     Eigen::Vector2d Eval(double& fz) override;
20
21 private:
22     std::vector<std::shared_ptr<ZeroMomentPoint>& zmp_vector_;
23 };
24
25 } // namespace huron

```

## 9.54 odrive.h

```

1 #pragma once
2
3 #include <cstdint>
4 #include <set>
5 #include <string>
6 #include <utility>
7 #include <memory>
8
9 #include "odrive_enums.h"
10 #include "huron/control_interfaces/generic_component.h"
11
12 namespace huron {
13 namespace odrive {

```



```

14
15 class ODrive : public huron::GenericComponent {
16 protected:
17     static const uint32_t kGetTimeout = 100; // ms
18
19     uint32_t get_timeout_;
20     bool is_calibrated_ = false;
21
22 public:
23     class ODriveConfiguration : public huron::Configuration {
24     public:
25         ODriveConfiguration(ConfigMap config_map,
26                             std::set<std::string> valid_keys)
27             : huron::Configuration(config_map,
28                                     [&valid_keys]() {
29                                         std::set<std::string> tmp(kODriveKeys);
30                                         tmp.merge(valid_keys);
31                                         return tmp;
32                                     }) {}
33
34         explicit ODriveConfiguration(ConfigMap config_map)
35             : ODriveConfiguration(config_map, std::set<std::string>()) {}
36
37         ODriveConfiguration()
38             : ODriveConfiguration(ConfigMap(), std::set<std::string>()) {}
39
40     private:
41         static const inline std::set<std::string> kODriveKeys{
42             "velocity_limit",
43             "current_limit",
44             "traj_vel_limit",
45             "traj_accel_limit",
46             "traj_decel_limit",
47             "traj_inertia"};
48     };
49
50     ODrive(std::unique_ptr<ODriveConfiguration> config, uint32_t get_timeout)
51         : huron::GenericComponent(std::move(config)),
52           get_timeout_(get_timeout) {}
53     explicit ODrive(uint32_t get_timeout = kGetTimeout)
54         : ODrive(std::make_unique<ODriveConfiguration>(),
55                 get_timeout) {}
56
57     ODrive(const ODrive&) = delete;
58     ODrive& operator=(const ODrive&) = delete;
59     ~ODrive() override = default;
60
61     void Initialize() override;
62
63     bool Calibrate();
64
65     void ConfigureKey(std::string config_key, std::any config_value) override;
66
67     // Get functions (msg.rtr bit must be set)
68     virtual bool GetMotorError(uint64_t& motor_error) = 0;
69     virtual bool GetEncoderError(uint32_t& encoder_error) = 0;
70     virtual bool GetControllerError(uint32_t& controller_error) = 0;
71     virtual bool GetSensorlessError(uint32_t& sensorless_error) = 0;
72     virtual bool GetEncoderEstimates(float& pos, float& vel) = 0;
73     virtual bool GetEncoderCount(int32_t& shadow_cnt, int32_t& cnt_cpr) = 0;
74     virtual bool GetIq(float& iq_setpoint, float& iq_measured) = 0;
75     virtual bool GetSensorlessEstimates(float& pos, float& vel) = 0;
76     virtual bool GetBusVoltageCurrent(float& bus_voltage, float& bus_current) = 0;
77     // msg.rtr bit must NOT be set
78     virtual bool GetAdcVoltage(float& adc_voltage) = 0;
79
80     // Set functions
81     virtual bool SetAxisNodeid(uint32_t axis_id) = 0;
82     virtual bool SetAxisRequestedState(uint32_t state) = 0;
83     virtual bool SetAxisStartupConfig() = 0;
84     virtual bool SetInputPos(float input_pos, int16_t vel_ff,
85                             int16_t torque_ff) = 0;
86     virtual bool SetInputVel(float input_vel, float torque_ff) = 0;
87     virtual bool SetInputTorque(float input_torque) = 0;
88     virtual bool SetControllerModes(int32_t control_mode, int32_t input_mode) = 0;
89     virtual bool SetLimits(float velocity_limit, float current_limit) = 0;
90     virtual bool SetTrajVelLimit(float traj_vel_limit) = 0;
91     virtual bool SetTrajAccelLimits(float traj_accel_limit,
92                                     float traj_decel_limit) = 0;
93     virtual bool SetTrajInertia(float traj_inertia) = 0;
94     virtual bool SetLinearCount(int32_t position) = 0;
95     virtual bool SetPosGain(float pos_gain) = 0;
96     virtual bool SetVelGains(float vel_gain, float vel_interator_gain) = 0;
97
98     // Other functions
99     virtual bool Nmt() = 0;
100     virtual bool Estop() = 0;
101     virtual bool ClearErrors() = 0;

```

```

120 virtual bool StartAnticogging() = 0;
121 };
122
123 } // namespace odrive
124 } // namespace huron

```

## 9.55 odrive\_can.h

```

1 #pragma once
2
3 #include <memory>
4 #include <string>
5
6 #include "odrive.h"
7 #include "huron/driver/can/canbus.h"
8
9 namespace huron {
10 namespace odrive {
11
12 class ODriveCAN : public ODrive {
13 private:
14     static const uint32_t kRecvTimeout = 100; // ms
15
16 public:
17     enum {
18         MSG_CO_NMT_CTRL = 0x000, // CANOpen NMT Message REC
19         MSG_ODRIVE_HEARTBEAT,
20         MSG_ODRIVE_ESTOP,
21         MSG_GET_MOTOR_ERROR, // Errors
22         MSG_GET_ENCODER_ERROR,
23         MSG_GET_SENSORLESS_ERROR,
24         MSG_SET_AXIS_NODE_ID,
25         MSG_SET_AXIS_REQUESTED_STATE,
26         MSG_SET_AXIS_STARTUP_CONFIG,
27         MSG_GET_ENCODER_ESTIMATES,
28         MSG_GET_ENCODER_COUNT,
29         MSG_SET_CONTROLLER_MODES,
30         MSG_SET_INPUT_POS,
31         MSG_SET_INPUT_VEL,
32         MSG_SET_INPUT_TORQUE,
33         MSG_SET_LIMITS,
34         MSG_START_ANTICOGGING,
35         MSG_SET_TRAJ_VEL_LIMIT,
36         MSG_SET_TRAJ_ACCEL_LIMITS,
37         MSG_SET_TRAJ_INERTIA,
38         MSG_GET_IQ,
39         MSG_GET_SENSORLESS_ESTIMATES,
40         MSG_RESET_ODRIVE,
41         MSG_GET_BUS_VOLTAGE_CURRENT,
42         MSG_CLEAR_ERRORS,
43         MSG_SET_LINEAR_COUNT,
44         MSG_SET_POS_GAIN,
45         MSG_SET_VEL_GAINS,
46         MSG_GET_ADC_VOLTAGE,
47         MSG_GET_CONTROLLER_ERROR,
48         MSG_CO_HEARTBEAT_CMD = 0x700, // CANOpen NMT Heartbeat SEND
49     };
50
51 ODriveCAN(huron::driver::can::BusBase* canbus,
52            uint32_t axis_id,
53            std::unique_ptr<ODriveConfiguration> config,
54            uint32_t get_timeout = kGetTimeout);
55 ODriveCAN(const ODriveCAN&) = delete;
56 ODriveCAN& operator=(const ODriveCAN&) = delete;
57 virtual ~ODriveCAN() = default;
58
59 // GenericComponent interface
60 void SetUp() override;
61 void Terminate() override;
62
63 // Get functions (msg.rtr bit must be set)
64 bool GetMotorError(uint64_t& motor_error) override;
65 bool GetEncoderError(uint32_t& encoder_error) override;
66 bool GetControllerError(uint32_t& controller_error) override;
67 bool GetSensorlessError(uint32_t& sensorless_error) override;
68 bool GetEncoderEstimates(float& pos, float& vel) override;
69 bool GetEncoderCount(int32_t& shadow_cnt, int32_t& cnt_cpr) override;
70 bool GetIq(float& iq_setpoint, float& iq_measured) override;
71 bool GetSensorlessEstimates(float& pos, float& vel) override;
72 bool GetBusVoltageCurrent(float& bus_voltage, float& bus_current) override;
73 // msg.rtr bit must NOT be set
74 bool GetAdcVoltage(float& adc_voltage) override;
75
76 // Set functions
77 bool SetAxisNodeid(uint32_t axis_id) override;
78 bool SetAxisRequestedState(uint32_t state) override;

```

```

89  bool SetAxisStartupConfig() override;
90  bool SetInputPos(float input_pos, int16_t vel_ff,
91      int16_t torque_ff) override;
92  bool SetInputVel(float input_vel, float torque_ff) override;
93  bool SetInputTorque(float input_torque) override;
94  bool SetControllerModes(int32_t control_mode, int32_t input_mode) override;
95  bool SetLimits(float velocity_limit, float current_limit) override;
96  bool SetTrajVelLimit(float traj_vel_limit) override;
97  bool SetTrajAccelLimits(float traj_accel_limit,
98      float traj_decel_limit) override;
99  bool SetTrajInertia(float traj_inertia) override;
100 bool SetLinearCount(int32_t position) override;
101 bool SetPosGain(float pos_gain) override;
102 bool SetVelGains(float vel_gain, float vel_interator_gain) override;
103
104 // Other functions
105 bool Nmt() override;
106 bool Estop() override;
107 bool ClearErrors() override;
108 bool StartAnticogging() override;
109
110 static constexpr uint8_t NUM_NODE_ID_BITS = 6;
111 static constexpr uint8_t NUM_CMD_ID_BITS = 11 - NUM_NODE_ID_BITS;
112
113 // Utility functions
114 static constexpr uint32_t GetNodeId(uint32_t msgID) {
115     return (msgID >> NUM_CMD_ID_BITS); // Upper 6 or more bits
116 }
117
118 static constexpr uint8_t GetCmdId(uint32_t msgID) {
119     return (msgID & 0x01F); // Bottom 5 bits
120 }
121
122 private:
123     huron::driver::can::BusBase* canbus_;
124     uint32_t can_id_;
125     uint32_t axis_id_;
126     bool is_ext_ = false;
127 };
128
129 } // namespace odrive
130 } // namespace huron

```

## 9.56 odrive\_enums.h

```

1  /*
2  * Original source from: https://github.com/odriverobotics/ODrive/tree/master
3  */
4  #pragma once
5
6  /* TODO: This file is dangerous because the enums could potentially change between API versions. Should
   * transmit as part of the JSON.
7  ** To regenerate this file, nagivate to the top level of the ODrive repository and run:
8  ** python Firmware/interface_generator_stub.py --definitions Firmware/odrive-interface.yaml --template
   * tools/arduino_enums_template.j2 --output Arduino/ODriveArduino/ODriveEnums.h
9  */
10
11 // ODrive.GpioMode
12 enum GpioMode {
13     GPIO_MODE_DIGITAL = 0,
14     GPIO_MODE_DIGITAL_PULL_UP = 1,
15     GPIO_MODE_DIGITAL_PULL_DOWN = 2,
16     GPIO_MODE_ANALOG_IN = 3,
17     GPIO_MODE_UART_A = 4,
18     GPIO_MODE_UART_B = 5,
19     GPIO_MODE_UART_C = 6,
20     GPIO_MODE_CAN_A = 7,
21     GPIO_MODE_I2C_A = 8,
22     GPIO_MODE_SPI_A = 9,
23     GPIO_MODE_PWM = 10,
24     GPIO_MODE_ENCO = 11,
25     GPIO_MODE_ENC1 = 12,
26     GPIO_MODE_ENC2 = 13,
27     GPIO_MODE_MECH_BRAKE = 14,
28     GPIO_MODE_STATUS = 15,
29 };
30
31 // ODrive.StreamProtocolType
32 enum StreamProtocolType {
33     STREAM_PROTOCOL_TYPE_FIBRE = 0,
34     STREAM_PROTOCOL_TYPE_ASCII = 1,
35     STREAM_PROTOCOL_TYPE_STDOUT = 2,
36     STREAM_PROTOCOL_TYPE_ASCII_AND_STDOUT = 3,
37 };
38

```

```

39 // ODrive.Can.Protocol
40 enum Protocol {
41     PROTOCOL_SIMPLE = 0x00000001,
42 };
43
44 // ODrive.Axis.AxisState
45 enum AxisState {
46     AXIS_STATE_UNDEFINED = 0,
47     AXIS_STATE_IDLE = 1,
48     AXIS_STATE_STARTUP_SEQUENCE = 2,
49     AXIS_STATE_FULL_CALIBRATION_SEQUENCE = 3,
50     AXIS_STATE_MOTOR_CALIBRATION = 4,
51     AXIS_STATE_ENCODER_INDEX_SEARCH = 6,
52     AXIS_STATE_ENCODER_OFFSET_CALIBRATION = 7,
53     AXIS_STATE_CLOSED_LOOP_CONTROL = 8,
54     AXIS_STATE_LOCKIN_SPIN = 9,
55     AXIS_STATE_ENCODER_DIR_FIND = 10,
56     AXIS_STATE_HOMING = 11,
57     AXIS_STATE_ENCODER_HALL_POLARITY_CALIBRATION = 12,
58     AXIS_STATE_ENCODER_HALL_PHASE_CALIBRATION = 13,
59 };
60
61 // ODrive.Encoder.Mode
62 enum EncoderMode {
63     ENCODER_MODE_INCREMENTAL = 0,
64     ENCODER_MODE_HALL = 1,
65     ENCODER_MODE_SINCOS = 2,
66     ENCODER_MODE_SPI_ABS_CUI = 256,
67     ENCODER_MODE_SPI_ABS_AMS = 257,
68     ENCODER_MODE_SPI_ABS_AEAT = 258,
69     ENCODER_MODE_SPI_ABS_RLS = 259,
70     ENCODER_MODE_SPI_ABS_MA732 = 260,
71 };
72
73 // ODrive.Controller.ControlMode
74 enum ControlMode {
75     CONTROL_MODE_VOLTAGE_CONTROL = 0,
76     CONTROL_MODE_TORQUE_CONTROL = 1,
77     CONTROL_MODE_VELOCITY_CONTROL = 2,
78     CONTROL_MODE_POSITION_CONTROL = 3,
79 };
80
81 // ODrive.Controller.InputMode
82 enum InputMode {
83     INPUT_MODE_INACTIVE = 0,
84     INPUT_MODE_PASSTHROUGH = 1,
85     INPUT_MODE_VEL_RAMP = 2,
86     INPUT_MODE_POS_FILTER = 3,
87     INPUT_MODE_MIX_CHANNELS = 4,
88     INPUT_MODE_TRAP_TRAJ = 5,
89     INPUT_MODE_TORQUE_RAMP = 6,
90     INPUT_MODE_MIRROR = 7,
91     INPUT_MODE_TUNING = 8,
92 };
93
94 // ODrive.Motor.MotorType
95 enum MotorType {
96     MOTOR_TYPE_HIGH_CURRENT = 0,
97     MOTOR_TYPE_GIMBAL = 2,
98     MOTOR_TYPE_ACIM = 3,
99 };
100
101 // ODrive.Error
102 enum ODriveError {
103     ODRIVE_ERROR_NONE = 0x00000000,
104     ODRIVE_ERROR_CONTROL_ITERATION_MISSED = 0x00000001,
105     ODRIVE_ERROR_DC_BUS_UNDER_VOLTAGE = 0x00000002,
106     ODRIVE_ERROR_DC_BUS_OVER_VOLTAGE = 0x00000004,
107     ODRIVE_ERROR_DC_BUS_OVER_REGEN_CURRENT = 0x00000008,
108     ODRIVE_ERROR_DC_BUS_OVER_CURRENT = 0x00000010,
109     ODRIVE_ERROR_BRAKE_DEADTIME_VIOLATION = 0x00000020,
110     ODRIVE_ERROR_BRAKE_DUTY_CYCLE_NAN = 0x00000040,
111     ODRIVE_ERROR_INVALID_BRAKE_RESISTANCE = 0x00000080,
112 };
113
114 // ODrive.Can.Error
115 enum CanError {
116     CAN_ERROR_NONE = 0x00000000,
117     CAN_ERROR_DUPLICATE_CAN_IDS = 0x00000001,
118 };
119
120 // ODrive.Axis.Error
121 enum AxisError {
122     AXIS_ERROR_NONE = 0x00000000,
123     AXIS_ERROR_INVALID_STATE = 0x00000001,
124     AXIS_ERROR_MOTOR_FAILED = 0x00000040,
125     AXIS_ERROR_SENSORLESS_ESTIMATOR_FAILED = 0x00000080,

```

```

126  AXIS_ERROR_ENCODER_FAILED                = 0x00000100,
127  AXIS_ERROR_CONTROLLER_FAILED             = 0x00000200,
128  AXIS_ERROR_WATCHDOG_TIMER_EXPIRED       = 0x00000800,
129  AXIS_ERROR_MIN_ENDSTOP_PRESSED          = 0x00001000,
130  AXIS_ERROR_MAX_ENDSTOP_PRESSED          = 0x00002000,
131  AXIS_ERROR_ESTOP_REQUESTED              = 0x00004000,
132  AXIS_ERROR_HOMING_WITHOUT_ENDSTOP        = 0x00020000,
133  AXIS_ERROR_OVER_TEMP                    = 0x00040000,
134  AXIS_ERROR_UNKNOWN_POSITION              = 0x00080000,
135 };
136
137 // ODrive.Motor.Error
138 enum MotorError {
139     MOTOR_ERROR_NONE                      = 0x00000000,
140     MOTOR_ERROR_PHASE_RESISTANCE_OUT_OF_RANGE = 0x00000001,
141     MOTOR_ERROR_PHASE_INDUCTANCE_OUT_OF_RANGE = 0x00000002,
142     MOTOR_ERROR_DRV_FAULT                 = 0x00000008,
143     MOTOR_ERROR_CONTROL_DEADLINE_MISSED   = 0x00000010,
144     MOTOR_ERROR_MODULATION_MAGNITUDE      = 0x00000080,
145     MOTOR_ERROR_CURRENT_SENSE_SATURATION   = 0x00000400,
146     MOTOR_ERROR_CURRENT_LIMIT_VIOLATION    = 0x00001000,
147     MOTOR_ERROR_MODULATION_IS_NAN         = 0x00010000,
148     MOTOR_ERROR_MOTOR_THERMISTOR_OVER_TEMP = 0x00020000,
149     MOTOR_ERROR_FET_THERMISTOR_OVER_TEMP  = 0x00040000,
150     MOTOR_ERROR_TIMER_UPDATE_MISSED       = 0x00080000,
151     MOTOR_ERROR_CURRENT_MEASUREMENT_UNAVAILABLE = 0x00100000,
152     MOTOR_ERROR_CONTROLLER_FAILED         = 0x00200000,
153     MOTOR_ERROR_I_BUS_OUT_OF_RANGE        = 0x00400000,
154     MOTOR_ERROR_BRAKE_RESISTOR_DISARMED    = 0x00800000,
155     MOTOR_ERROR_SYSTEM_LEVEL              = 0x01000000,
156     MOTOR_ERROR_BAD_TIMING                 = 0x02000000,
157     MOTOR_ERROR_UNKNOWN_PHASE_ESTIMATE     = 0x04000000,
158     MOTOR_ERROR_UNKNOWN_PHASE_VEL         = 0x08000000,
159     MOTOR_ERROR_UNKNOWN_TORQUE             = 0x10000000,
160     MOTOR_ERROR_UNKNOWN_CURRENT_COMMAND    = 0x20000000,
161     MOTOR_ERROR_UNKNOWN_CURRENT_MEASUREMENT = 0x40000000,
162     MOTOR_ERROR_UNKNOWN_VBUS_VOLTAGE       = 0x80000000,
163     MOTOR_ERROR_UNKNOWN_VOLTAGE_COMMAND    = 0x100000000,
164     MOTOR_ERROR_UNKNOWN_GAINS              = 0x200000000,
165     MOTOR_ERROR_CONTROLLER_INITIALIZING     = 0x400000000,
166     MOTOR_ERROR_UNBALANCED_PHASES          = 0x800000000,
167 };
168
169 // ODrive.Controller.Error
170 enum ControllerError {
171     CONTROLLER_ERROR_NONE                  = 0x00000000,
172     CONTROLLER_ERROR_OVERSPEED             = 0x00000001,
173     CONTROLLER_ERROR_INVALID_INPUT_MODE    = 0x00000002,
174     CONTROLLER_ERROR_UNSTABLE_GAIN         = 0x00000004,
175     CONTROLLER_ERROR_INVALID_MIRROR_AXIS   = 0x00000008,
176     CONTROLLER_ERROR_INVALID_LOAD_ENCODER  = 0x00000010,
177     CONTROLLER_ERROR_INVALID_ESTIMATE      = 0x00000020,
178     CONTROLLER_ERROR_INVALID_CIRCULAR_RANGE = 0x00000040,
179     CONTROLLER_ERROR_SPINOUT_DETECTED      = 0x00000080,
180 };
181
182 // ODrive.Encoder.Error
183 enum EncoderError {
184     ENCODER_ERROR_NONE                     = 0x00000000,
185     ENCODER_ERROR_UNSTABLE_GAIN            = 0x00000001,
186     ENCODER_ERROR_CPR_POLEPAIRS_MISMATCH  = 0x00000002,
187     ENCODER_ERROR_NO_RESPONSE              = 0x00000004,
188     ENCODER_ERROR_UNSUPPORTED_ENCODER_MODE = 0x00000008,
189     ENCODER_ERROR_ILLEGAL_HALL_STATE       = 0x00000010,
190     ENCODER_ERROR_INDEX_NOT_FOUND_YET      = 0x00000020,
191     ENCODER_ERROR_ABS_SPI_TIMEOUT           = 0x00000040,
192     ENCODER_ERROR_ABS_SPI_COM_FAIL         = 0x00000080,
193     ENCODER_ERROR_ABS_SPI_NOT_READY        = 0x00000100,
194     ENCODER_ERROR_HALL_NOT_CALIBRATED_YET  = 0x00000200,
195 };
196
197 // ODrive.SensorlessEstimator.Error
198 enum SensorlessEstimatorError {
199     SENSORLESS_ESTIMATOR_ERROR_NONE        = 0x00000000,
200     SENSORLESS_ESTIMATOR_ERROR_UNSTABLE_GAIN = 0x00000001,
201     SENSORLESS_ESTIMATOR_ERROR_UNKNOWN_CURRENT_MEASUREMENT = 0x00000002,
202 };
203

```

## 9.57 odrive\_rotary\_encoder.h

```

1  #pragma once
2
3  #include <memory>
4  #include "huron/control_interfaces/rotary_encoder.h"

```

```

5 #include "huron/odrive/odrive.h"
6
7 namespace huron {
8 namespace odrive {
9
10 class ODriveEncoder final : public huron::RotaryEncoder {
11 public:
12     ODriveEncoder(double gear_ratio,
13                  std::unique_ptr<RotaryEncoderConfiguration> config,
14                  std::shared_ptr<ODrive> odrive);
15     ODriveEncoder(double gear_ratio, double cpr, std::shared_ptr<ODrive> odrive);
16     ODriveEncoder(double cpr, std::shared_ptr<ODrive> odrive);
17     ODriveEncoder(const ODriveEncoder&) = delete;
18     ODriveEncoder& operator=(const ODriveEncoder&) = delete;
19     ~ODriveEncoder() override = default;
20
21     void Initialize() override;
22     void SetUp() override;
23     void Terminate() override;
24
25 protected:
26     void DoUpdateState() override;
27
28 private:
29     std::shared_ptr<ODrive> odrive_;
30 };
31
32 } // namespace odrive
33 } // namespace huron

```

## 9.58 odrive\_torque\_motor.h

```

1 #pragma once
2
3 #include <memory>
4 #include <vector>
5
6 #include "huron/control_interfaces/torque_motor.h"
7 #include "huron/odrive/odrive_can.h"
8
9 namespace huron {
10 namespace odrive {
11
12 class TorqueMotor : public huron::TorqueMotor {
13 public:
14     TorqueMotor(
15         std::unique_ptr<TorqueMotorConfiguration> config,
16         std::shared_ptr<ODrive> odrive,
17         double gear_ratio);
18     TorqueMotor(
19         std::shared_ptr<ODrive> odrive,
20         double gear_ratio);
21     explicit TorqueMotor(std::shared_ptr<ODrive> odrive);
22     TorqueMotor(const TorqueMotor&) = delete;
23     TorqueMotor& operator=(const TorqueMotor&) = delete;
24     ~TorqueMotor() = default;
25
26     // GenericComponent methods
27     void Initialize() override;
28     void SetUp() override;
29     void Terminate() override;
30
31     bool Move(double value) override;
32     bool Move(const std::vector<double>& values) override;
33     bool Move(const Eigen::VectorXd& values) override;
34     bool Stop() override;
35
36 private:
37     std::shared_ptr<ODrive> odrive_;
38 };
39
40 } // namespace odrive
41 } // namespace huron

```

## 9.59 force\_sensing\_resistor.h

```

1 #pragma once
2
3 #include <eigen3/Eigen/Core>
4 #include <memory>
5 #include "huron/control_interfaces/sensor_with_frame.h"
6

```

```

7 namespace huron {
8
9 class ForceSensingResistor : public SensorWithFrame {
10 public:
11     explicit ForceSensingResistor(std::weak_ptr<const multibody::Frame> frame);
12     ForceSensingResistor(std::weak_ptr<const multibody::Frame> frame,
13                          std::unique_ptr<Configuration> config);
14     ForceSensingResistor(const ForceSensingResistor&) = delete;
15     ForceSensingResistor& operator=(const ForceSensingResistor&) = delete;
16     ~ForceSensingResistor() override = default;
17 };
18
19 } // namespace huron

```

## 9.60 force\_sensing\_resistor\_array.h

```

1 #pragma once
2
3 #include <eigen3/Eigen/Core>
4
5 #include <string>
6 #include <memory>
7 #include <vector>
8
9 #include "huron/control_interfaces/sensor_with_frame.h"
10 #include "huron/sensors/force_sensing_resistor.h"
11
12 namespace huron {
13
14 class ForceSensingResistorArray : public SensorWithFrame {
15 public:
16     ForceSensingResistorArray(
17         const std::string& name,
18         std::weak_ptr<const multibody::Frame> frame,
19         const std::vector<std::shared_ptr<ForceSensingResistor>& fsr_array);
20     ForceSensingResistorArray(
21         const std::string& name,
22         std::weak_ptr<const multibody::Frame> frame,
23         const std::vector<std::shared_ptr<ForceSensingResistor>& fsr_array,
24         std::unique_ptr<Configuration> config);
25
26     ForceSensingResistorArray(const ForceSensingResistorArray&) = delete;
27     ForceSensingResistorArray&
28         operator=(const ForceSensingResistorArray&) = delete;
29     ~ForceSensingResistorArray() override = default;
30
31     void RequestStateUpdate() override;
32
33     void GetNewState(Eigen::Ref<Eigen::MatrixXd> new_state) const override;
34
35     Eigen::Affine3d GetSensorPose(size_t index) const;
36
37     size_t num_sensors() const { return fsr_array_.size(); }
38
39 protected:
40     std::string name_;
41     Eigen::VectorXd values_;
42     std::vector<std::shared_ptr<ForceSensingResistor>> fsr_array_;
43 };
44
45 } // namespace huron

```

## 9.61 force\_sensing\_resistor\_array\_serial.h

```

1 #pragma once
2
3 #include <memory>
4 #include <vector>
5 #include <string>
6
7 #include "huron/sensors/force_sensing_resistor_array.h"
8 #include "huron/driver/serial/serial.h"
9
10 namespace huron {
11
12 class ForceSensingResistorArraySerial : public ForceSensingResistorArray {
13 public:
14     ForceSensingResistorArraySerial(
15         const std::string& name,
16         std::weak_ptr<const multibody::Frame> frame,
17         const std::vector<std::shared_ptr<ForceSensingResistor>& fsr_array,
18         std::shared_ptr<driver::serial::SerialBase> serial);
19
20 };
21
22 } // namespace huron

```

```

26 ForceSensingResistorArraySerial(
27     const std::string& name,
28     std::weak_ptr<const multibody::Frame> frame,
29     const std::vector<std::shared_ptr<ForceSensingResistor>>& fsr_array,
30     std::shared_ptr<driver::serial::SerialBase> serial,
31     std::unique_ptr<Configuration> config);
32
33 ForceSensingResistorArraySerial(
34     const ForceSensingResistorArraySerial&) = delete;
35
36 ForceSensingResistorArraySerial&
37     operator=(const ForceSensingResistorArraySerial&) = delete;
38
39 ~ForceSensingResistorArraySerial() override = default;
40
41 void RequestStateUpdate() override;
42
43 Eigen::VectorXd GetValue() const override;
44 Eigen::VectorXd ReloadAndGetValue() override;
45
46 void Initialize() override;
47 void SetUp() override;
48 void Terminate() override;
49
50 private:
51     static inline const std::string delimiter = ",";
52     std::shared_ptr<driver::serial::SerialBase> serial_;
53 };
54
55 } // namespace huron

```

## 9.62 string.h

```

1 #pragma once
2
3 #include <string>
4 #include <vector>
5
6 namespace huron {
7 namespace utils {
8
9 std::vector<std::string> split(const std::string& str,
10                             const std::string& delimiter);
11 } // namespace utils
12 } // namespace huron

```

## 9.63 time.h

```

1 #pragma once
2
3 #include <chrono> //NOLINT
4
5 template <
6     class result_t    = std::chrono::milliseconds,
7     class clock_t     = std::chrono::steady_clock,
8     class duration_t  = std::chrono::milliseconds
9 >
10 auto since(std::chrono::time_point<clock_t, duration_t> const& start) {
11     return std::chrono::duration_cast<result_t>(clock_t::now() - start);
12 }
13
14
15

```



# Index

/github/workspace/common/include/huron/enable_protected_joint_group.h,	108	/github/workspace/ros2/src/huron_ros2/include/huron_ros2/joint_group_co
/github/workspace/common/include/huron/enable_protected_joint_state_pro	108	
/github/workspace/common/include/huron/types.h,	89	/github/workspace/system/control_interfaces/include/huron/control_interfa
/github/workspace/control/include/huron/control/push_recovery.h,	109	
	90	/github/workspace/system/control_interfaces/include/huron/control_interfa
/github/workspace/driver/can/include/huron/driver/can/can_helpers.h,	109	
	91	/github/workspace/system/control_interfaces/include/huron/control_interfa
/github/workspace/driver/can/include/huron/driver/can/canbus.h,	110	
	92	/github/workspace/system/control_interfaces/include/huron/control_interfa
/github/workspace/driver/can/include/huron/driver/can/socket_can_bus.h,	110	
	92	/github/workspace/system/control_interfaces/include/huron/control_interfa
/github/workspace/driver/config/include/huron/driver/config/config.h,	111	
	93	/github/workspace/system/control_interfaces/include/huron/control_interfa
/github/workspace/driver/config/include/huron/driver/config/odrive_conf.h,	111	
	93	/github/workspace/system/control_interfaces/include/huron/control_interfa
/github/workspace/driver/serial/include/huron/driver/serial/serial.h,	113	
	93	/github/workspace/system/control_interfaces/include/huron/control_interfa
/github/workspace/driver/serial/include/huron/driver/serial/wjwood_serial.h,	113	
	95	/github/workspace/system/control_interfaces/include/huron/control_interfa
/github/workspace/math/include/huron/math/rotation.h,	113	
	96	/github/workspace/system/control_interfaces/include/huron/control_interfa
/github/workspace/multibody/include/huron/multibody/com_frame.h,	114	
	96	/github/workspace/system/control_interfaces/include/huron/control_interfa
/github/workspace/multibody/include/huron/multibody/frame.h,	115	
	96	/github/workspace/system/control_interfaces/include/huron/control_interfa
/github/workspace/multibody/include/huron/multibody/joint_common.h,	115	
	97	/github/workspace/system/control_interfaces/include/huron/control_interfa
/github/workspace/multibody/include/huron/multibody/logical_frame.h,	115	
	99	/github/workspace/system/control_interfaces/include/huron/control_interfa
/github/workspace/multibody/include/huron/multibody/model.h,	116	
	100	/github/workspace/system/control_interfaces/include/huron/control_interfa
/github/workspace/multibody/include/huron/multibody/model_composi	116	
	102	/github/workspace/system/control_interfaces/include/huron/control_interfa
/github/workspace/multibody/include/huron/multibody/model_impl_fact	116	
	102	/github/workspace/system/control_interfaces/include/huron/control_interfa
/github/workspace/multibody/include/huron/multibody/model_impl_inte	116	
	103	/github/workspace/system/control_interfaces/include/huron/control_interfa
/github/workspace/multibody/include/huron/multibody/model_impl_type	116	
	104	/github/workspace/system/control_interfaces/include/huron/control_interfa
/github/workspace/multibody/include/huron/multibody/pinocchio_model	119	
	104	/github/workspace/system/exceptions/include/huron/exceptions/invalid_co
/github/workspace/ros2/src/huron_ros2/include/huron_ros2/force_tor	120	
	105	/github/workspace/system/exceptions/include/huron/exceptions/not_imple
/github/workspace/ros2/src/huron_ros2/include/huron_ros2/huron.h,	120	
	106	/github/workspace/system/locomotion/include/huron/locomotion/zero_mon
/github/workspace/ros2/src/huron_ros2/include/huron_ros2/huron_not	121	
	107	/github/workspace/system/locomotion/include/huron/locomotion/zero_mon

- 121
- /github/workspace/system/locomotion/include/huron/locomotion/DriveState.h, 122
- /github/workspace/system/locomotion/include/huron/locomotion/huron\_RotaryEncoder.h, 122
- /github/workspace/system/odrive/include/huron/odrive/odrive.h, 122
- /github/workspace/system/odrive/include/huron/odrive/odrive\_can.h, 124
- /github/workspace/system/odrive/include/huron/odrive/odrive\_enums.h, 125
- /github/workspace/system/odrive/include/huron/odrive/odrive\_rotary\_encoder.h, 127
- /github/workspace/system/odrive/include/huron/odrive/odrive\_torque\_motor.h, 128
- /github/workspace/system/sensors/include/huron/sensors/force\_sensing\_resistor.h, 128
- /github/workspace/system/sensors/include/huron/sensors/force\_sensing\_resistor\_array.h, 129
- /github/workspace/system/sensors/include/huron/sensors/force\_sensing\_resistor\_array\_serial.h, 129
- /github/workspace/system/sensors/include/huron/sensors/force\_torque\_sensor.h, 106
- /github/workspace/utils/include/huron/utils/string.h, 130
- /github/workspace/utils/include/huron/utils/time.h, 130
- AddFrame
  - huron::multibody::Model, 42
- AddJoint
  - huron::multibody::Model, 42
- AddModelImpl
  - huron::multibody::Model, 43
- Available
  - huron::driver::serial::Serial, 74
- BuildFromUrdf
  - huron::multibody::internal::PinocchioModelImpl, 62
- Calibrate
  - huron::odrive::ODrive, 52
- can\_Cyclic\_t, 17
- can\_Message\_t, 18
- can\_Signal\_t, 18
- ClearErrors
  - huron::odrive::ODriveCAN, 54
- Close
  - huron::driver::serial::Serial, 74
- ComputeAll
  - huron::multibody::internal::PinocchioModelImpl, 62
- Configure
  - huron::GenericComponent, 31, 32
- ConfigureKey
  - huron::GenericComponent, 32
  - huron::odrive::ODrive, 52
- ConfigureMap
  - huron::GenericComponent, 32
- DoGetWrenchRaw
  - huron::ForceTorqueSensor, 28
- huron::ros2::ForceTorqueSensor, 29
- DriveState
  - huron::odrive::ODriveEncoder, 60
  - huron::RotaryEncoder, 61
- EStop
  - huron::odrive::ODriveCAN, 55
- Eval
  - huron::ZeroMomentPoint, 85
  - huron::ZeroMomentPointFSRArray, 86
  - huron::ZeroMomentPointFTSensor, 87
  - huron::ZeroMomentPointTotal, 88
- EvalCenterOfMassPosition
  - huron::multibody::internal::PinocchioModelImpl, 62
- EvalZeroMomentPoint
  - huron::LegeesRobot, 38
- Finalize
  - huron::multibody::Model, 43
- Flush
  - huron::driver::serial::Serial, 74
- FlushInput
  - huron::driver::serial::Serial, 74
- FlushOutput
  - huron::driver::serial::Serial, 74
- ForwardKinematics
  - huron::multibody::internal::PinocchioModelImpl, 62
- Get
  - huron::Configuration, 20
- GetAccelerations
  - huron::multibody::internal::ModelImplInterface, 45
  - huron::multibody::internal::PinocchioModelImpl, 62
- GetAdcVoltage
  - huron::odrive::ODriveCAN, 55
- GetAngleDegree
  - huron::RotaryEncoder, 70
- GetBaudrate
  - huron::driver::serial::Serial, 74
- GetBusVoltageCurrent
  - huron::odrive::ODriveCAN, 55
- GetCenterOfMassPosition
  - huron::multibody::internal::PinocchioModelImpl, 62
- GetCentroidalMatrix
  - huron::multibody::internal::ModelImplInterface, 46
  - huron::multibody::internal::PinocchioModelImpl, 63
- GetCentroidalMomentum
  - huron::multibody::internal::ModelImplInterface, 46
  - huron::multibody::internal::PinocchioModelImpl, 63
- GetControllerError
  - huron::odrive::ODriveCAN, 55
- GetCoriolisMatrix
  - huron::multibody::internal::ModelImplInterface, 46
  - huron::multibody::internal::PinocchioModelImpl, 63
- GetCount
  - huron::RotaryEncoder, 70
- GetCPR
  - huron::RotaryEncoder, 70
- GetEncoderCount

- huron::odrive::ODriveCAN, 55
- GetEncoderError
  - huron::odrive::ODriveCAN, 55
- GetEncoderEstimates
  - huron::odrive::ODriveCAN, 55
- GetFlowcontrol
  - huron::driver::serial::Serial, 75
- GetFrameIndex
  - huron::multibody::internal::PinocchioModelImpl, 63
- GetFrameName
  - huron::multibody::internal::PinocchioModelImpl, 63
- GetFrameTransform
  - huron::multibody::internal::PinocchioModelImpl, 63
- GetFrameTransformInWorld
  - huron::multibody::internal::PinocchioModelImpl, 63
- GetFrameType
  - huron::multibody::internal::PinocchioModelImpl, 63
- GetFromComponent
  - huron::Configuration, 20
- GetGravity
  - huron::multibody::internal::ModelImplInterface, 46
  - huron::multibody::internal::PinocchioModelImpl, 64
- GetIq
  - huron::odrive::ODriveCAN, 55
- GetJoint
  - huron::multibody::internal::PinocchioModelImpl, 64
- GetJointDescription
  - huron::multibody::internal::PinocchioModelImpl, 64
- GetJointIndex
  - huron::multibody::internal::PinocchioModelImpl, 64
- GetJointNames
  - huron::multibody::internal::PinocchioModelImpl, 64
- GetJointTransformInWorld
  - huron::multibody::internal::PinocchioModelImpl, 64
- GetJointType
  - huron::multibody::internal::PinocchioModelImpl, 65
- GetMassMatrix
  - huron::multibody::internal::ModelImplInterface, 46
  - huron::multibody::internal::PinocchioModelImpl, 65
- GetModelImpl
  - huron::multibody::Model, 43
- GetMotorError
  - huron::odrive::ODriveCAN, 56
- GetNewState
  - huron::ConstantStateProvider, 21
  - huron::Encoder, 23
  - huron::ForceSensingResistorArray, 25
  - huron::ForceTorqueSensor, 28
  - huron::ros2::JointStateProvider, 37
- GetNonlinearEffects
  - huron::multibody::internal::ModelImplInterface, 46
  - huron::multibody::internal::PinocchioModelImpl, 65
- GetParity
  - huron::driver::serial::Serial, 75
- GetPort
  - huron::driver::serial::Serial, 75
- GetPosition
  - huron::Encoder, 23
- huron::RotaryEncoder, 70
- GetPrevCount
  - huron::RotaryEncoder, 70
- GetSensorlessError
  - huron::odrive::ODriveCAN, 56
- GetSensorlessEstimates
  - huron::odrive::ODriveCAN, 56
- GetSpatialMomentum
  - huron::multibody::internal::ModelImplInterface, 46
  - huron::multibody::internal::PinocchioModelImpl, 65
- GetStopbits
  - huron::driver::serial::Serial, 75
- GetTorques
  - huron::multibody::internal::ModelImplInterface, 47
  - huron::multibody::internal::PinocchioModelImpl, 65
- GetTransformFromFrame
  - huron::multibody::ComFrame, 19
  - huron::multibody::LogicalFrame, 40
- GetTransformInWorld
  - huron::multibody::ComFrame, 19
  - huron::multibody::LogicalFrame, 40
- GetTransformToFrame
  - huron::multibody::ComFrame, 19
  - huron::multibody::LogicalFrame, 40
- GetValue
  - huron::ForceSensingResistorArraySerial, 26
  - huron::ForceTorqueSensor, 28
  - huron::Sensor, 72
- GetVelocity
  - huron::Encoder, 23
  - huron::RotaryEncoder, 70
- GetVelocityCount
  - huron::RotaryEncoder, 71
- GetVelocityDegree
  - huron::RotaryEncoder, 71
- huron, 13
- huron::Actuator, 15
- huron::ActuatorConfiguration, 15
- huron::Configuration, 20
  - Get, 20
  - GetFromComponent, 20
  - ValidateKey, 21
- huron::ConstantStateProvider, 21
  - GetNewState, 21
  - RequestStateUpdate, 21
- huron::driver::can::BusBase, 16
  - recv\_message, 16
  - send\_message, 17
  - subscribe, 17
  - unsubscribe, 17
- huron::driver::can::BusBase::CanSubscription, 18
- huron::driver::can::MsgIdFilterSpecs, 50
- huron::driver::can::SocketCanBus, 79
  - recv\_message, 79
  - send\_message, 80
  - subscribe, 80
  - unsubscribe, 80
- huron::driver::serial::Serial, 73

- Available, 74
- Close, 74
- Flush, 74
- FlushInput, 74
- FlushOutput, 74
- GetBaudrate, 74
- GetFlowcontrol, 75
- GetParity, 75
- GetPort, 75
- GetStopbits, 75
- IsOpen, 75
- Open, 75
- Read, 75, 76
- ReadLine, 76
- ReadLines, 76
- SendBreak, 76
- SetBaudrate, 76
- SetFlowcontrol, 76
- SetParity, 77
- SetPort, 77
- SetStopbits, 77
- SetTimeout, 77
- WaitReadable, 77
- Write, 77
- huron::driver::serial::SerialBase, 78
- huron::enable\_protected\_make\_shared< ClassWith-ProtectedCtor >, 21
- huron::enable\_protected\_make\_unique< ClassWith-ProtectedCtor >, 22
- huron::Encoder, 22
  - GetNewState, 23
  - GetPosition, 23
  - GetVelocity, 23
  - Reset, 23
- huron::EncoderConfiguration, 24
- huron::ForceSensingResistor, 24
- huron::ForceSensingResistorArray, 25
  - GetNewState, 25
  - RequestStateUpdate, 25
- huron::ForceSensingResistorArraySerial, 26
  - GetValue, 26
  - Initialize, 27
  - ReloadAndGetValue, 27
  - RequestStateUpdate, 27
  - SetUp, 27
  - Terminate, 27
- huron::ForceTorqueSensor, 27
  - DoGetWrenchRaw, 28
  - GetNewState, 28
  - GetValue, 28
  - RequestStateUpdate, 28
- huron::GenericComponent, 31
  - Configure, 31, 32
  - ConfigureKey, 32
  - ConfigureMap, 32
  - Initialize, 32
- huron::InvalidConfigurationException, 34
- huron::Joint, 35
  - Joint, 35
- huron::LeggedRobot, 38
  - EvalZeroMomentPoint, 38
- huron::Limb, 38
- huron::Motor, 47
- huron::MotorConfiguration, 47
- huron::MovingGroup, 48
  - Move, 48, 49
  - Stop, 49
- huron::MovingInterface, 49
  - Move, 50
  - Stop, 50
- huron::multibody::ComFrame, 18
  - GetTransformFromFrame, 19
  - GetTransformInWorld, 19
  - GetTransformToFrame, 19
- huron::multibody::Frame, 30
- huron::multibody::internal::ModelImplFactory, 44
- huron::multibody::internal::ModelImplInterface, 44
  - GetAccelerations, 45
  - GetCentroidalMatrix, 46
  - GetCentroidalMomentum, 46
  - GetCoriolisMatrix, 46
  - GetGravity, 46
  - GetMassMatrix, 46
  - GetNonlinearEffects, 46
  - GetSpatialMomentum, 46
  - GetTorques, 47
- huron::multibody::internal::PinocchioModelImpl, 60
  - BuildFromUrdf, 62
  - ComputeAll, 62
  - EvalCenterOfMassPosition, 62
  - ForwardKinematics, 62
  - GetAccelerations, 62
  - GetCenterOfMassPosition, 62
  - GetCentroidalMatrix, 63
  - GetCentroidalMomentum, 63
  - GetCoriolisMatrix, 63
  - GetFrameIndex, 63
  - GetFrameName, 63
  - GetFrameTransform, 63
  - GetFrameTransformInWorld, 63
  - GetFrameType, 63
  - GetGravity, 64
  - GetJoint, 64
  - GetJointDescription, 64
  - GetJointIndex, 64
  - GetJointNames, 64
  - GetJointTransformInWorld, 64
  - GetJointType, 65
  - GetMassMatrix, 65
  - GetNonlinearEffects, 65
  - GetSpatialMomentum, 65
  - GetTorques, 65
  - is\_built, 65
  - NeutralConfiguration, 65
  - num\_frames, 65
  - num\_joints, 66

- num\_positions, 66
- num\_velocities, 66
- huron::multibody::internal::PinocchioModelImpl::Impl, 34
- huron::multibody::JointDescription, 35
- huron::multibody::LogicalFrame, 39
  - GetTransformFromFrame, 40
  - GetTransformInWorld, 40
  - GetTransformToFrame, 40
- huron::multibody::Model, 40
  - AddFrame, 42
  - AddJoint, 42
  - AddModelImpl, 43
  - Finalize, 43
  - GetModelImpl, 43
  - UpdateJointStates, 43
- huron::multibody::ModelComposite, 44
- huron::NotImplementedException, 51
- huron::odrive::ODrive, 51
  - Calibrate, 52
  - ConfigureKey, 52
  - Initialize, 52
- huron::odrive::ODrive::ODriveConfiguration, 58
  - ODriveConfiguration, 59
- huron::odrive::ODriveCAN, 53
  - ClearErrors, 54
  - Estop, 55
  - GetAdcVoltage, 55
  - GetBusVoltageCurrent, 55
  - GetControllerError, 55
  - GetEncoderCount, 55
  - GetEncoderError, 55
  - GetEncoderEstimates, 55
  - GetIq, 55
  - GetMotorError, 56
  - GetSensorlessError, 56
  - GetSensorlessEstimates, 56
  - Nmt, 56
  - ODriveCAN, 54
  - SetAxisNodeid, 56
  - SetAxisRequestedState, 56
  - SetAxisStartupConfig, 56
  - SetControllerModes, 56
  - SetInputPos, 57
  - SetInputTorque, 57
  - SetInputVel, 57
  - SetLimits, 57
  - SetLinearCount, 57
  - SetPosGain, 57
  - SetTrajAccelLimits, 57
  - SetTrajInertia, 58
  - SetTrajVelLimit, 58
  - SetUp, 58
  - SetVelGains, 58
  - StartAnticogging, 58
  - Terminate, 58
- huron::odrive::ODriveEncoder, 59
  - DoUpdateState, 60
  - Initialize, 60
  - SetUp, 60
  - Terminate, 60
- huron::odrive::TorqueMotor, 81
  - Initialize, 82
  - Move, 82
  - SetUp, 82
  - Stop, 82
  - Terminate, 82
- huron::PositionMotor, 66
- huron::PositionMotorConfiguration, 67
  - PositionMotorConfiguration, 67
- huron::Robot, 67
  - UpdateAllStates, 68
  - UpdateJointStates, 68
- huron::RobotConfiguration, 69
- huron::ros2::ForceTorqueSensor, 29
  - DoGetWrenchRaw, 29
  - Initialize, 29
  - SetUp, 29
  - Terminate, 29
- huron::ros2::Huron, 32
  - Initialize, 33
  - SetUp, 33
  - Terminate, 33
- huron::ros2::HuronNode, 33
- huron::ros2::JointGroupController, 36
  - Move, 36
  - Stop, 37
- huron::ros2::JointStateProvider, 37
  - GetNewState, 37
  - RequestStateUpdate, 37
- huron::RotaryEncoder, 69
  - DoUpdateState, 70
  - GetAngleDegree, 70
  - GetCount, 70
  - GetCPR, 70
  - GetPosition, 70
  - GetPrevCount, 70
  - GetVelocity, 70
  - GetVelocityCount, 71
  - GetVelocityDegree, 71
  - RequestStateUpdate, 71
  - Reset, 71
- huron::RotaryEncoderConfiguration, 71
  - RotaryEncoderConfiguration, 71
- huron::Sensor, 72
  - GetValue, 72
- huron::SensorWithFrame, 72
- huron::StateProvider, 80
- huron::TorqueMotor, 83
  - TorqueMotorConfiguration, 83
  - TorqueMotorConfiguration, 84
- huron::VelocityMotor, 84
  - VelocityMotorConfiguration, 84
  - VelocityMotorConfiguration, 85
- huron::ZeroMomentPoint, 85
  - Eval, 85

- ZmpToWorld, [86](#)
- huron::ZeroMomentPointFSRArray, [86](#)
  - Eval, [86](#)
- huron::ZeroMomentPointFTSensor, [87](#)
  - Eval, [87](#)
- huron::ZeroMomentPointTotal, [87](#)
  - Eval, [88](#)
- Initialize
  - huron::ForceSensingResistorArraySerial, [27](#)
  - huron::GenericComponent, [32](#)
  - huron::odrive::ODrive, [52](#)
  - huron::odrive::ODriveEncoder, [60](#)
  - huron::odrive::TorqueMotor, [82](#)
  - huron::ros2::ForceTorqueSensor, [29](#)
  - huron::ros2::Huron, [33](#)
- is\_built
  - huron::multibody::internal::PinocchioModelImpl, [65](#)
- IsOpen
  - huron::driver::serial::Serial, [75](#)
- Joint
  - huron::Joint, [35](#)
- Move
  - huron::MovingGroup, [48, 49](#)
  - huron::MovingInterface, [50](#)
  - huron::odrive::TorqueMotor, [82](#)
  - huron::ros2::JointGroupController, [36](#)
- NeutralConfiguration
  - huron::multibody::internal::PinocchioModelImpl, [65](#)
- Nmt
  - huron::odrive::ODriveCAN, [56](#)
- num\_frames
  - huron::multibody::internal::PinocchioModelImpl, [65](#)
- num\_joints
  - huron::multibody::internal::PinocchioModelImpl, [66](#)
- num\_positions
  - huron::multibody::internal::PinocchioModelImpl, [66](#)
- num\_velocities
  - huron::multibody::internal::PinocchioModelImpl, [66](#)
- ODriveCAN
  - huron::odrive::ODriveCAN, [54](#)
- ODriveConfiguration
  - huron::odrive::ODrive::ODriveConfiguration, [59](#)
- Open
  - huron::driver::serial::Serial, [75](#)
- PositionMotorConfiguration
  - huron::PositionMotorConfiguration, [67](#)
- PushRecoveryControl, [67](#)
- Read
  - huron::driver::serial::Serial, [75, 76](#)
- ReadLine
  - huron::driver::serial::Serial, [76](#)
- ReadLines
  - huron::driver::serial::Serial, [76](#)
- recv\_message
  - huron::driver::can::BusBase, [16](#)
  - huron::driver::can::SocketCanBus, [79](#)
- ReloadAndGetVale
  - huron::ForceSensingResistorArraySerial, [27](#)
- RequestStateUpdate
  - huron::ConstantStateProvider, [21](#)
  - huron::ForceSensingResistorArray, [25](#)
  - huron::ForceSensingResistorArraySerial, [27](#)
  - huron::ForceTorqueSensor, [28](#)
  - huron::ros2::JointStateProvider, [37](#)
  - huron::RotaryEncoder, [71](#)
- Reset
  - huron::Encoder, [23](#)
  - huron::RotaryEncoder, [71](#)
- RotaryEncoderConfiguration
  - huron::RotaryEncoderConfiguration, [71](#)
- send\_message
  - huron::driver::can::BusBase, [17](#)
  - huron::driver::can::SocketCanBus, [80](#)
- SendBreak
  - huron::driver::serial::Serial, [76](#)
- SetAxisNodeid
  - huron::odrive::ODriveCAN, [56](#)
- SetAxisRequestedState
  - huron::odrive::ODriveCAN, [56](#)
- SetAxisStartupConfig
  - huron::odrive::ODriveCAN, [56](#)
- SetBaudrate
  - huron::driver::serial::Serial, [76](#)
- SetControllerModes
  - huron::odrive::ODriveCAN, [56](#)
- SetFlowcontrol
  - huron::driver::serial::Serial, [76](#)
- SetInputPos
  - huron::odrive::ODriveCAN, [57](#)
- SetInputTorque
  - huron::odrive::ODriveCAN, [57](#)
- SetInputVel
  - huron::odrive::ODriveCAN, [57](#)
- SetLimits
  - huron::odrive::ODriveCAN, [57](#)
- SetLinearCount
  - huron::odrive::ODriveCAN, [57](#)
- SetParity
  - huron::driver::serial::Serial, [77](#)
- SetPort
  - huron::driver::serial::Serial, [77](#)
- SetPosGain
  - huron::odrive::ODriveCAN, [57](#)
- SetStopbits
  - huron::driver::serial::Serial, [77](#)
- SetTimeout
  - huron::driver::serial::Serial, [77](#)
- SetTrajAccelLimits
  - huron::odrive::ODriveCAN, [57](#)
- SetTrajInertia
  - huron::odrive::ODriveCAN, [58](#)

- SetTrajVelLimit
  - [huron::odrive::ODriveCAN, 58](#)
- SetUp
  - [huron::ForceSensingResistorArraySerial, 27](#)
  - [huron::odrive::ODriveCAN, 58](#)
  - [huron::odrive::ODriveEncoder, 60](#)
  - [huron::odrive::TorqueMotor, 82](#)
  - [huron::ros2::ForceTorqueSensor, 29](#)
  - [huron::ros2::Huron, 33](#)
- SetVelGains
  - [huron::odrive::ODriveCAN, 58](#)
- StartAnticogging
  - [huron::odrive::ODriveCAN, 58](#)
- Stop
  - [huron::MovingGroup, 49](#)
  - [huron::MovingInterface, 50](#)
  - [huron::odrive::TorqueMotor, 82](#)
  - [huron::ros2::JointGroupController, 37](#)
- subscribe
  - [huron::driver::can::BusBase, 17](#)
  - [huron::driver::can::SocketCanBus, 80](#)
- Terminate
  - [huron::ForceSensingResistorArraySerial, 27](#)
  - [huron::odrive::ODriveCAN, 58](#)
  - [huron::odrive::ODriveEncoder, 60](#)
  - [huron::odrive::TorqueMotor, 82](#)
  - [huron::ros2::ForceTorqueSensor, 29](#)
  - [huron::ros2::Huron, 33](#)
- TorqueMotorConfiguration
  - [huron::TorqueMotorConfiguration, 84](#)
- unsubscribe
  - [huron::driver::can::BusBase, 17](#)
  - [huron::driver::can::SocketCanBus, 80](#)
- UpdateAllStates
  - [huron::Robot, 68](#)
- UpdateJointStates
  - [huron::multibody::Model, 43](#)
  - [huron::Robot, 68](#)
- ValidateKey
  - [huron::Configuration, 21](#)
- VelocityMotorConfiguration
  - [huron::VelocityMotorConfiguration, 85](#)
- WaitReadable
  - [huron::driver::serial::Serial, 77](#)
- Write
  - [huron::driver::serial::Serial, 77](#)
- ZmpToWorld
  - [huron::ZeroMomentPoint, 86](#)