Mumei

v0.1

Generated by Doxygen 1.9.5

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:

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

Chapter 5

Class Index

5.1 Class List

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

Chapter 6

File Index

6.1 File List

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

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

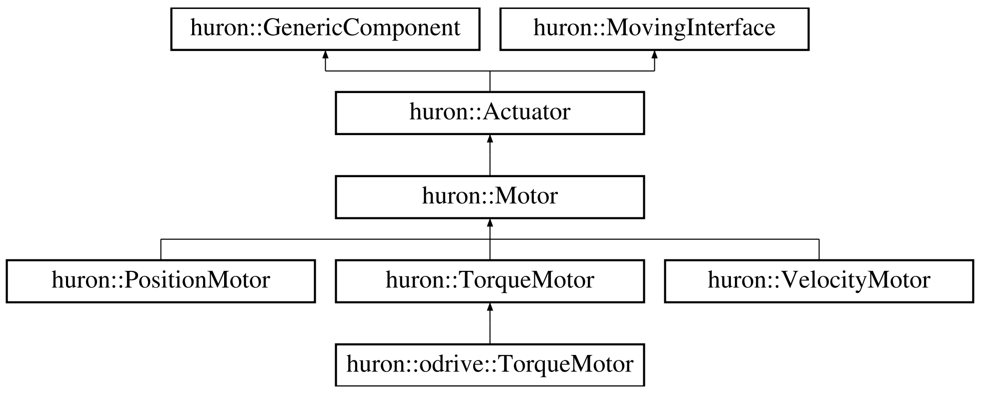
Source: `https://stackoverflow.com/questions/24469927/does-c-have-an-equivalent-to-nets-n`
See `https://docs.odriverobotics.com/v/0.5.6/can-protocol.html` for more information about this CAN API.

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

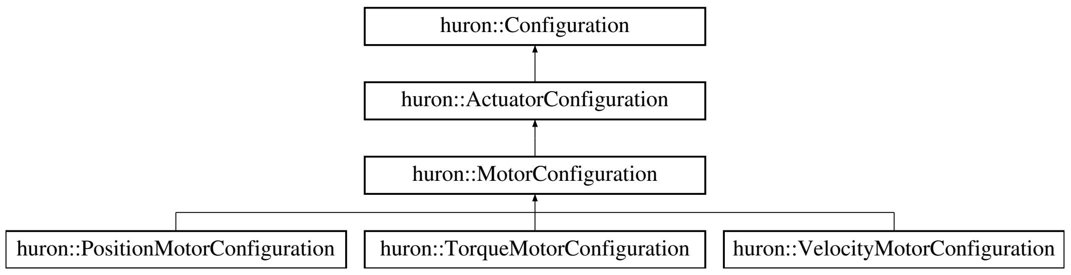
Additional Inherited Members

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:


huron::Configuration
huron::ActuatorConfiguration
huron::MotorConfiguration
huron::PositionMotorConfiguration
huron::TorqueMotorConfiguration
huron::VelocityMotorConfiguration

Public Member Functions

- **ActuatorConfiguration** (ConfigMap config_map, std::set< std::string > valid_keys)
- **ActuatorConfiguration** (ConfigMap config_map)

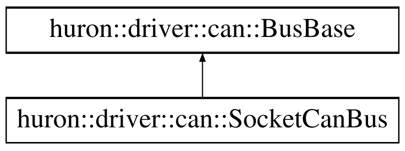
Additional Inherited Members

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

handle	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:

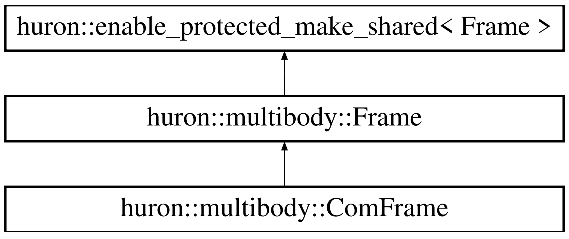
- /github/workspace/driver/can/include/huron/driver/can/canbus.h

8.8 huron::multibody::ComFrame Class Reference

Robot center of mass frame.

`#include <com_frame.h>`

Inheritance diagram for huron::multibody::ComFrame:


huron::enable_protected_make_shared< Frame >
huron::multibody::Frame
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.

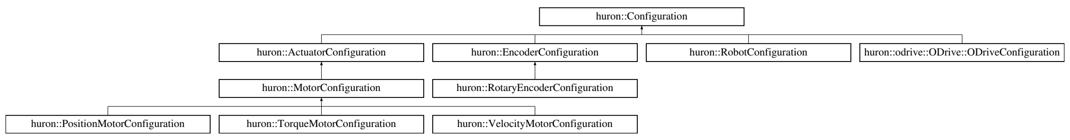
The documentation for this class was generated from the following files:

- /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:


huron::Configuration
huron::ActuatorConfiguration
huron::EncoderConfiguration
huron::RobotConfiguration
huron::odrive::ODrive::ODriveConfiguration
huron::MotorConfiguration
huron::RotaryEncoderConfiguration
huron::PositionMotorConfiguration
huron::TorqueMotorConfiguration
huron::VelocityMotorConfiguration

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

InvalidConfigurationException	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 overriden by concrete configuration classes.

8.9.2.3 ValidateKey()

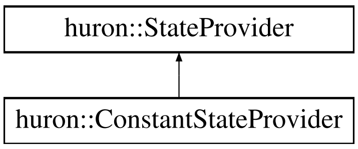
```
bool huron::Configuration::ValidateKey (
            std::string config_key )  [inline], [protected]
```

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

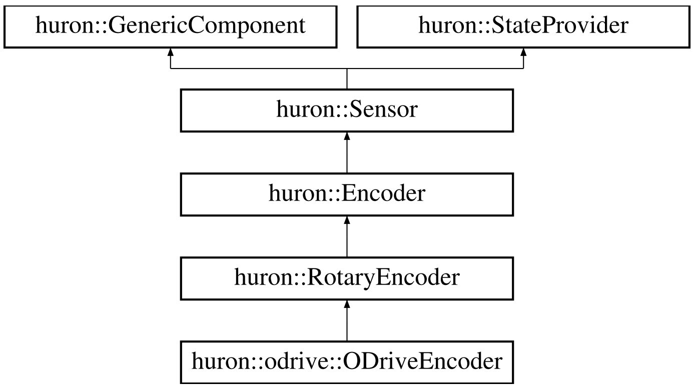
template<typename ClassWithProtectedCtor>
class huron::enable_protected_make_unique< ClassWithProtectedCtor >

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]
```

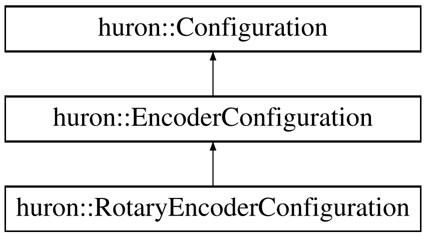
Implemented in huron::RotaryEncoder.

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:


huron::Configuration
huron::EncoderConfiguration
huron::RotaryEncoderConfiguration

Public Member Functions

- **EncoderConfiguration** (ConfigMap config_map, std::set< std::string > valid_keys)
- **EncoderConfiguration** (ConfigMap config_map)

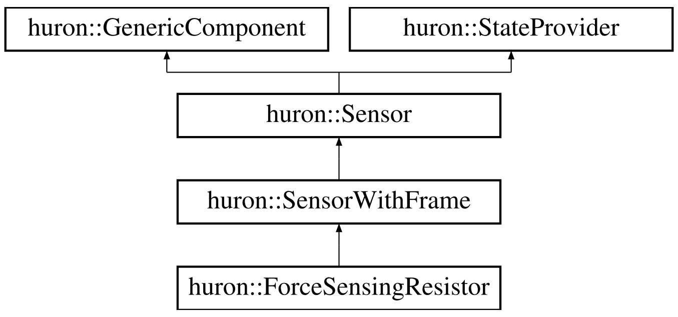
Additional Inherited Members

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

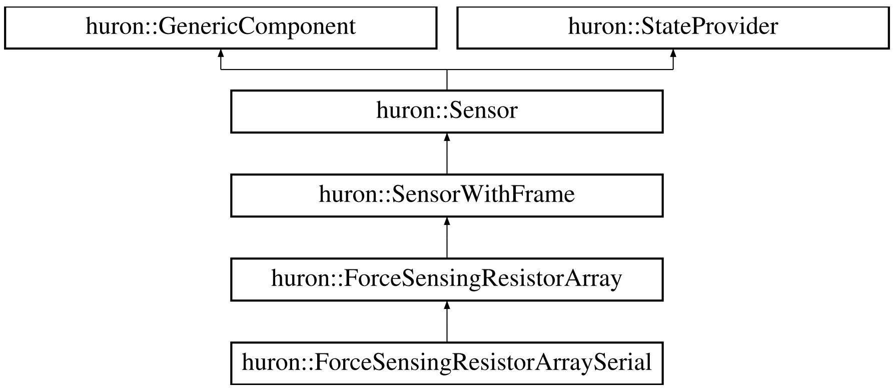
Additional Inherited Members

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.

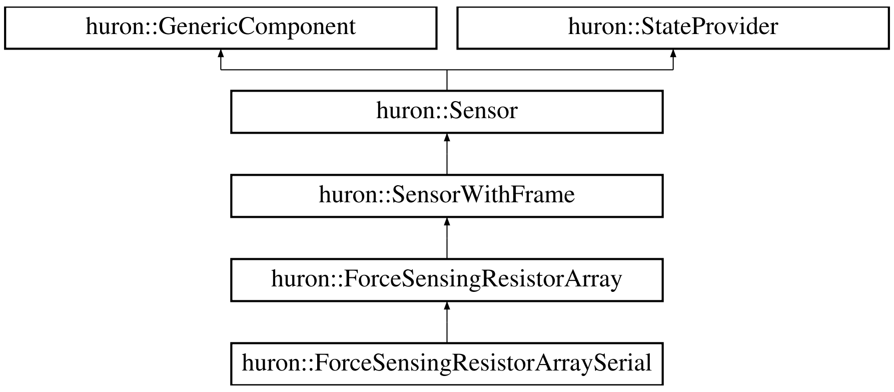
The documentation for this class was generated from the following files:

- /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.

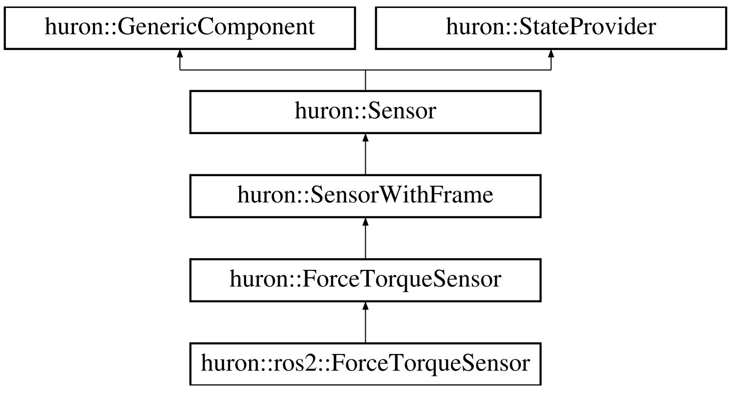
8.17.2.6 Terminate()

```
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 overriden.
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 $[Fx, Fy, Fz, Tx, Ty, Tz]^T$.

Reimplemented from huron::Sensor.

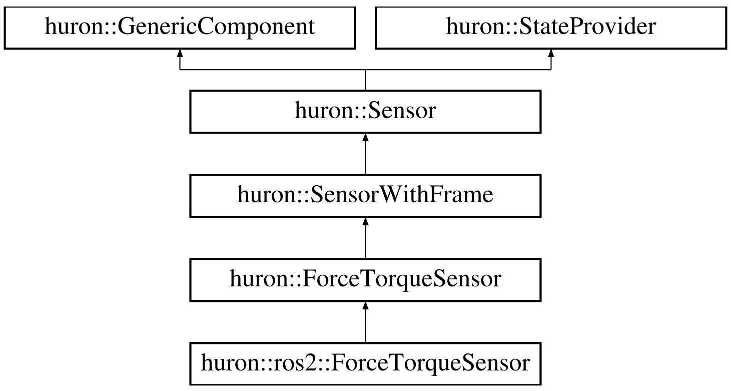
8.18.1.4 RequestStateUpdate()

```
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 overriden.

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]
```

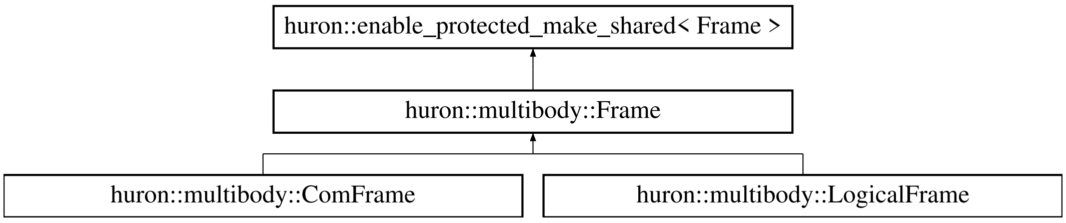
Implements huron::GenericComponent.

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:


huron::enable_protected_make_shared< Frame >
huron::multibody::Frame
huron::multibody::ComFrame
huron::multibody::LogicalFrame

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

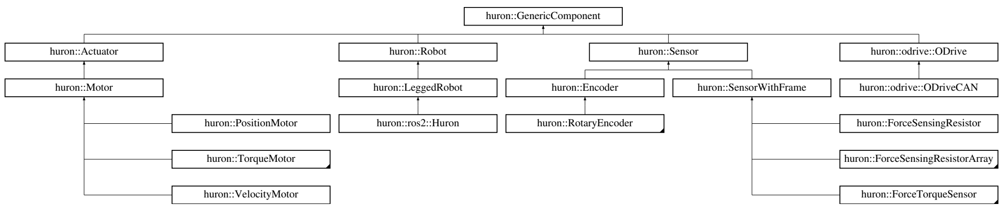
The documentation for this class was generated from the following files:

- /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()

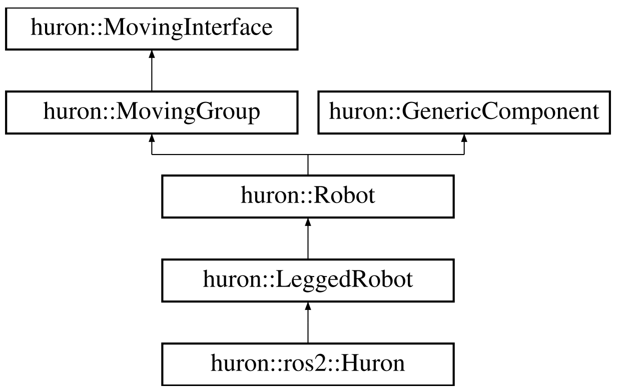
```
virtual void huron::GenericComponent::Initialize ( )  [pure virtual]
```
Implemented in huron::odrive::ODrive.

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()

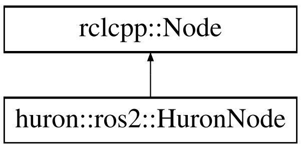
```
void huron::ros2::Huron::Terminate ( )  [override], [virtual]
```
Implements huron::GenericComponent.

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

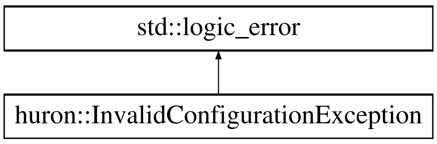
- int **dummy**
- pinocchio::Model **model_**
- pinocchio::Data **data_**

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 specfied 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

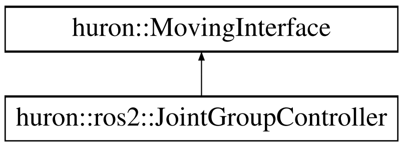
- const Eigen::VectorXd & **min_position** () const
- const Eigen::VectorXd & **max_position** () const
- const Eigen::VectorXd & **min_velocity** () const
- const Eigen::VectorXd & **max_velocity** () const
- const Eigen::VectorXd & **min_acceleration** () const
- const Eigen::VectorXd & **max_acceleration** () 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

value	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

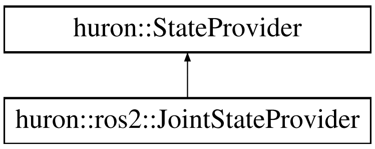
true if the operation is successful, false otherwise.

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]
```

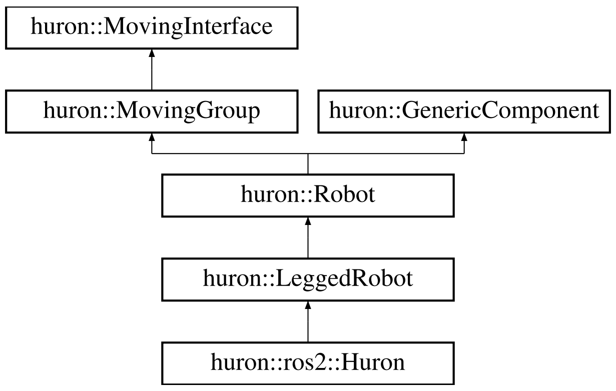
Implements huron::StateProvider.

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 ( )
```

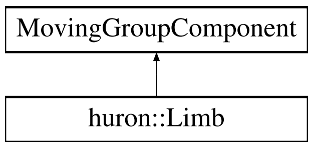
Computes the Center of Mass in Base frame.

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

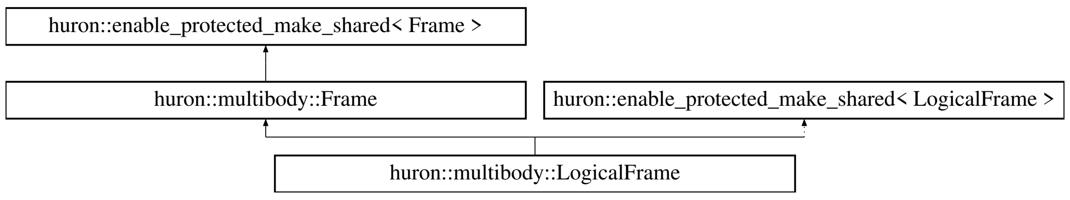
- void **Init** (std::vector< Joint > joints)
- void **AddJoint** (Joint &joint)

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:


huron::enable_protected_make_shared< Frame >
huron::multibody::Frame
huron::enable_protected_make_shared< LogicalFrame >
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

index	The index of this frame.
name	The name of this frame.
model	The model that this frame is a part of.
parent_frame_index	The index of the frame that this frame is defined relative to.
transform_function	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]
```

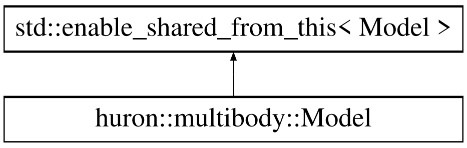
Reimplemented from huron::multibody::Frame.

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

- ModelImplInterface const ∗ GetModelImpl (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< ModelImplInterface > > **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

index	The index of the frame.
args	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

index	The index of the joint.
args	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

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

Exceptions

std::runtime_error	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

std::runtime_error	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()

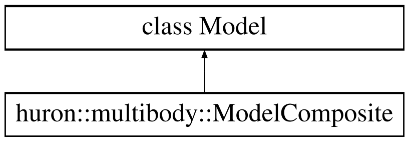
```
void huron::multibody::Model::UpdateJointStates ( )
```

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

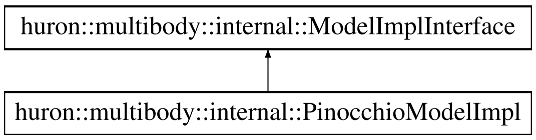
- class **multibody::Model**

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:


huron::multibody::internal::ModelImplInterface
huron::multibody::internal::PinocchioModelImpl

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()

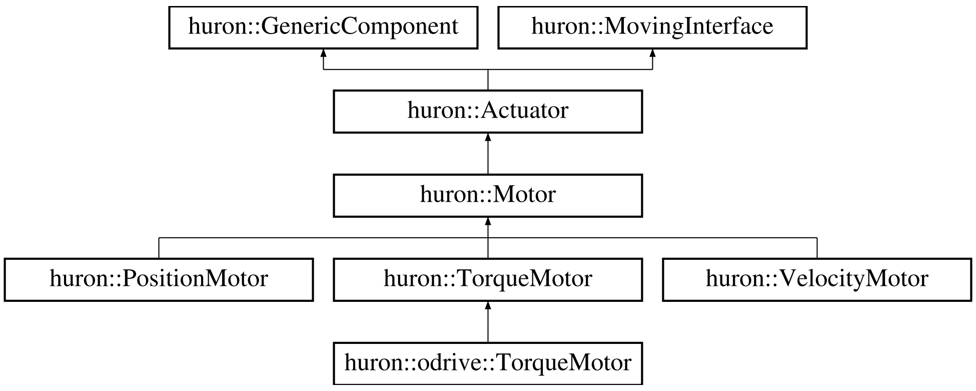
```
const Eigen::VectorXd & huron::multibody::internal::ModelImplInterface::GetTorques ( ) const
[virtual]
```

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

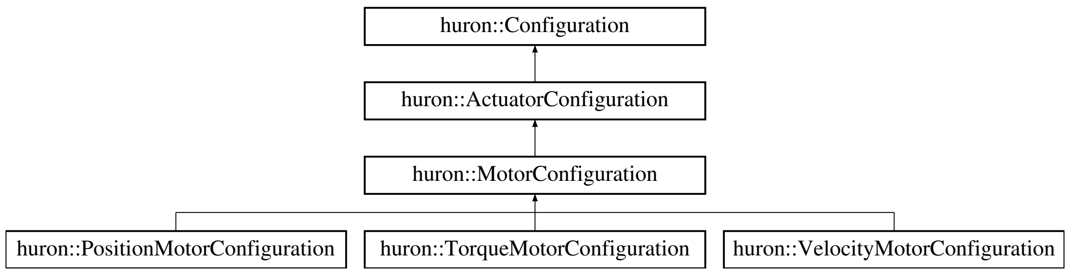
Additional Inherited Members

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:


huron::Configuration
huron::ActuatorConfiguration
huron::MotorConfiguration
huron::PositionMotorConfiguration
huron::TorqueMotorConfiguration
huron::VelocityMotorConfiguration

Public Member Functions

- **MotorConfiguration** (ConfigMap config_map, std::set< std::string > valid_keys)
- **MotorConfiguration** (ConfigMap config_map)

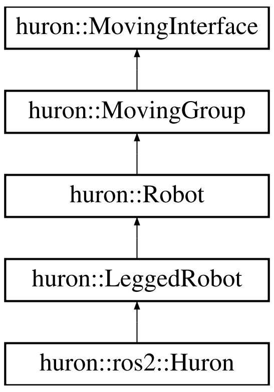
Additional Inherited Members

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

value	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.

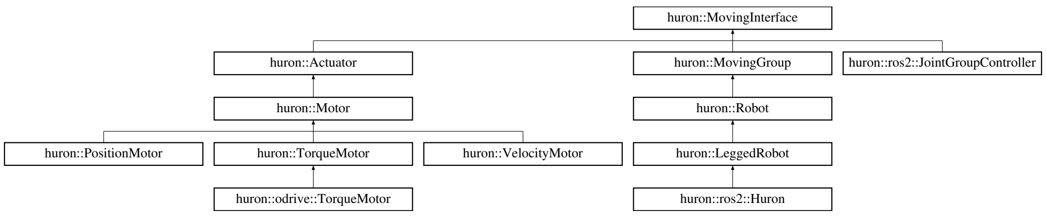
The documentation for this class was generated from the following files:

- /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

value	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

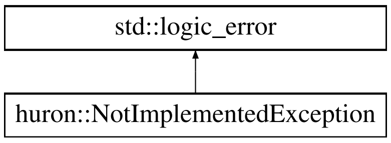
- std::variant< uint16_t, uint32_t > **id**
- uint32_t **mask**

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 ()

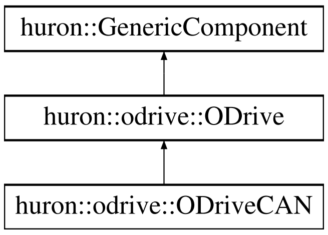
The documentation for this class was generated from the following file:

- /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()

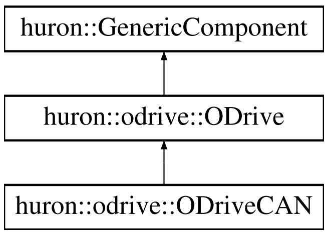
```
void huron::odrive::ODrive::Initialize ( )  [override], [virtual]
```

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]
```

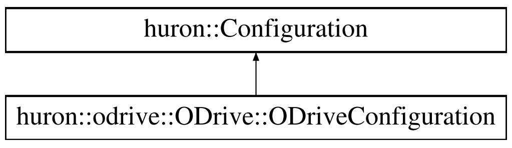
Implements huron::GenericComponent.

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]

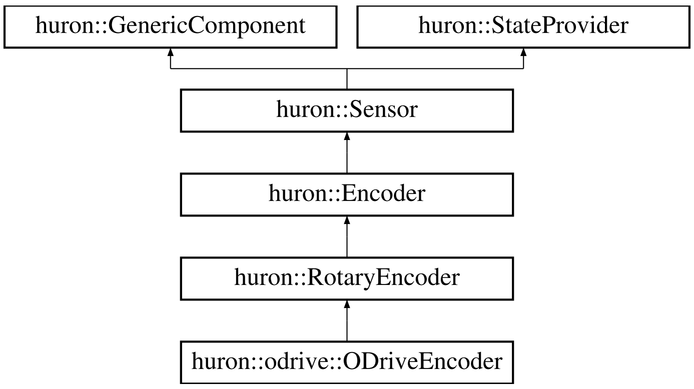
```
huron::odrive::ODrive::ODriveConfiguration::ODriveConfiguration ( )  [inline]
```

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 RequestUpdate↩ State(). 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()

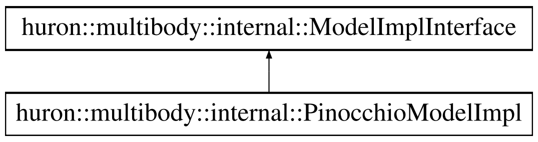
`void huron::odrive::ODriveEncoder::Terminate ( ) [override], [virtual]`

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:


huron::multibody::internal::ModelImplInterface
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::ModelImplInterface.

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::ModelImplInterface.

8.47.1.3 EvalCenterOfMassPosition()

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

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::ModelImplInterface.

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::ModelImplInterface.

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::ModelImplInterface.

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::ModelImplInterface.

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::MatrixXd & 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::ModelImplInterface.

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::ModelImplInterface.

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::ModelImplInterface.

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::ModelImplInterface.

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::ModelImplInterface.

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::ModelImplInterface.

8.47.1.22 GetJointIndex()

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

Implements huron::multibody::internal::ModelImplInterface.

8.47.1.23 GetJointNames()

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

Reimplemented from huron::multibody::internal::ModelImplInterface.

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.

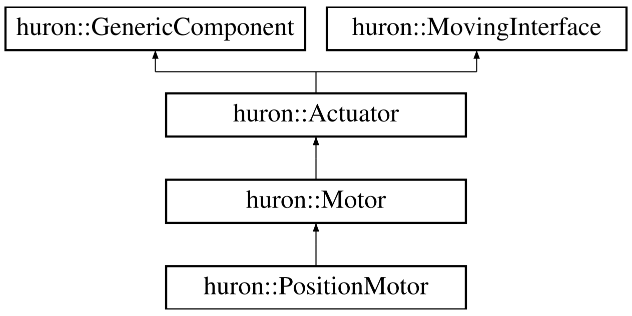
8.47.1.35 num_velocities()

```
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

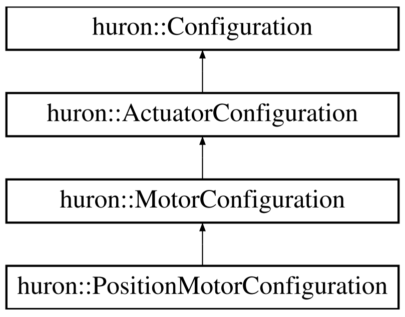
Additional Inherited Members

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:


huron::Configuration
huron::ActuatorConfiguration
huron::MotorConfiguration
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

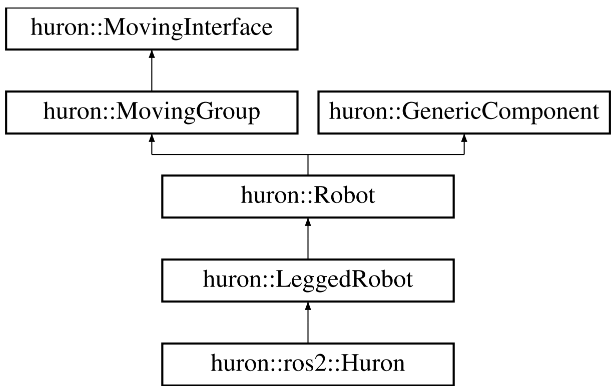
- Eigen::MatrixXd **GetTorque** (const Eigen::Vector2d &cop, const Eigen::VectorXd &position, const Eigen::↩ VectorXd &velocity)

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 ( )
```

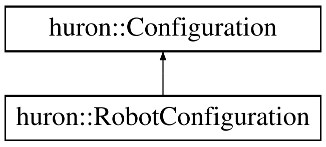
Calls RequestStateUpdate() on all the registered state providers for joints.

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

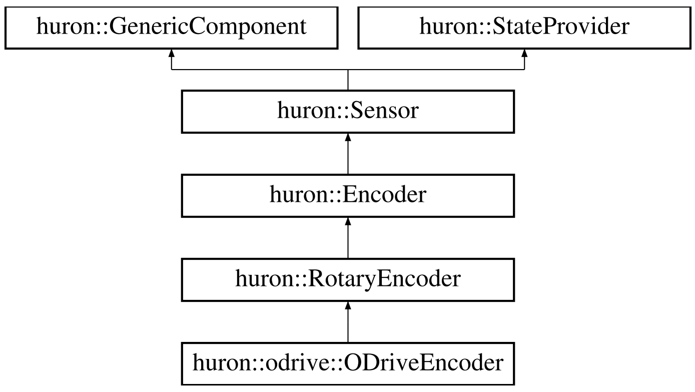
The documentation for this class was generated from the following file:

- /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 RequestUpdate↩ State(). 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.

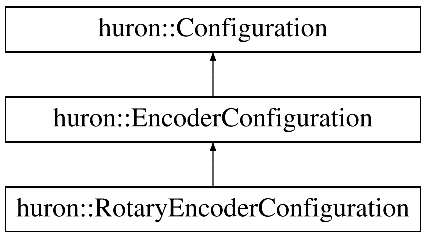
8.53.2.11 Reset()

```
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:


huron::Configuration
huron::EncoderConfiguration
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()

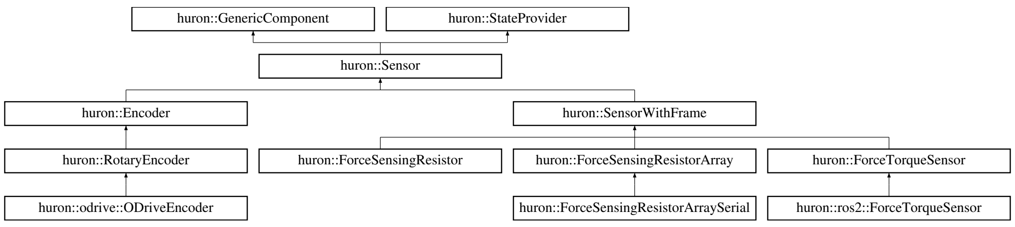
```
huron::RotaryEncoderConfiguration::RotaryEncoderConfiguration (
            ConfigMap config_map,
            std::set< std::string > valid_keys )  [inline]
```

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:


huron::GenericComponent
huron::StateProvider
huron::Sensor
huron::Encoder
huron::SensorWithFrame
huron::RotaryEncoder
huron::ForceSensingResistor
huron::ForceSensingResistorArray
huron::ForceTorqueSensor
huron::odrive::ODriveEncoder
huron::ForceSensingResistorArraySerial
huron::ros2::ForceTorqueSensor

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()

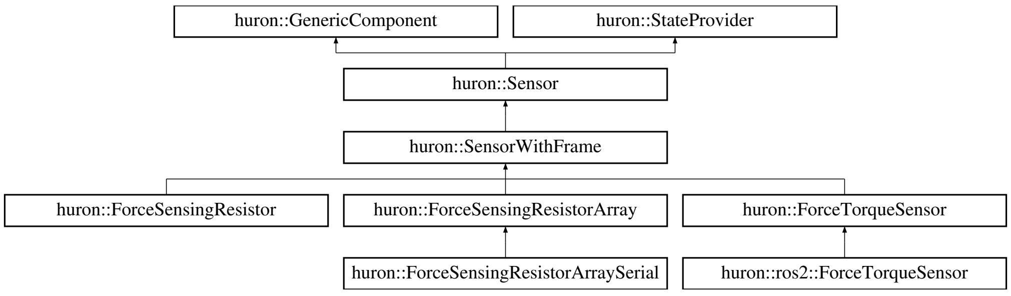
```
Eigen::VectorXd huron::Sensor::GetValue ( ) const  [virtual]
```

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:


huron::GenericComponent
huron::StateProvider
huron::Sensor
huron::SensorWithFrame
huron::ForceSensingResistor
huron::ForceSensingResistorArray
huron::ForceTorqueSensor
huron::ForceSensingResistorArraySerial
huron::ros2::ForceTorqueSensor

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.

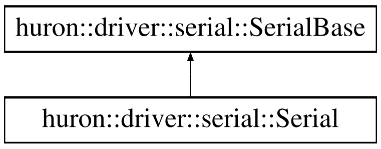
Additional Inherited Members

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:


huron::driver::serial::SerialBase
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]

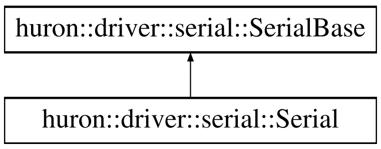
```
size_t huron::driver::serial::Serial::Write (
            const uint8_t * data,
            size_t nbytes )  [override], [virtual]
```

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:


huron::driver::serial::SerialBase
huron::driver::serial::Serial

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

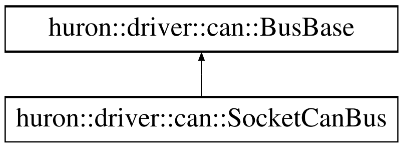
- std::string **port_**
- uint32_t **baudrate_**
- Parity **parity_**
- StopBits **stopbits_**
- FlowControl **flowcontrol_**

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

handle	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()

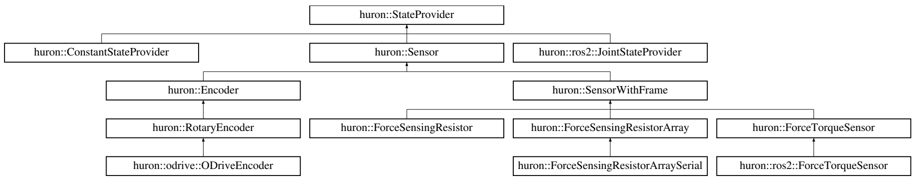
```
bool huron::driver::can::SocketCanBus::unsubscribe (
             CanSubscription * handle )  [final], [virtual]
```

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:


huron::StateProvider
huron::ConstantStateProvider
huron::Sensor
huron::ros2::JointStateProvider
huron::Encoder
huron::SensorWithFrame
huron::RotaryEncoder
huron::ForceSensingResistor
huron::ForceSensingResistorArray
huron::ForceTorqueSensor
huron::odrive::ODriveEncoder
huron::ForceSensingResistorArraySerial
huron::ros2::ForceTorqueSensor

Public Member Functions

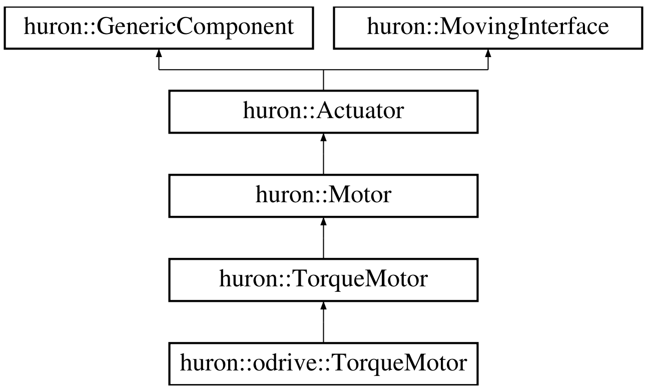
- **StateProvider** (const Eigen::Vector2i &dim)
- **StateProvider** (int rows, int cols)
- **StateProvider** (const StateProvider &)=delete
- StateProvider & **operator=** (const StateProvider &)=delete
- virtual void **RequestStateUpdate** ()=0
- virtual void **GetNewState** (Eigen::Ref< Eigen::MatrixXd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

value	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]`

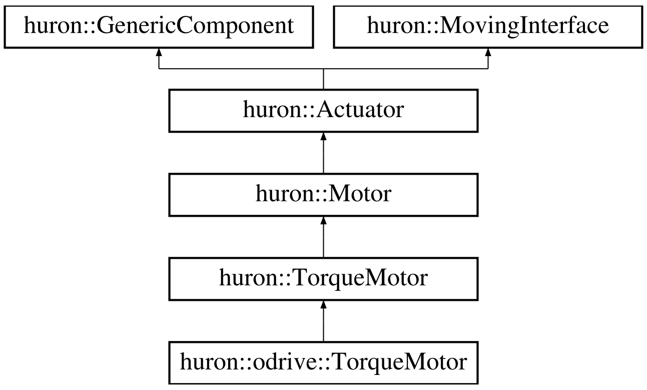
Implements huron::GenericComponent.

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

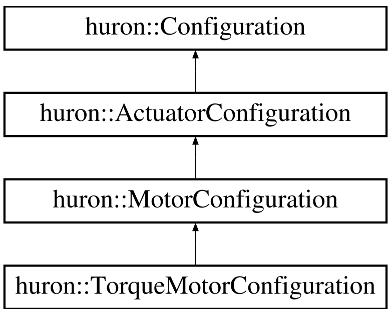
Additional Inherited Members

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:


huron::Configuration
huron::ActuatorConfiguration
huron::MotorConfiguration
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]
```

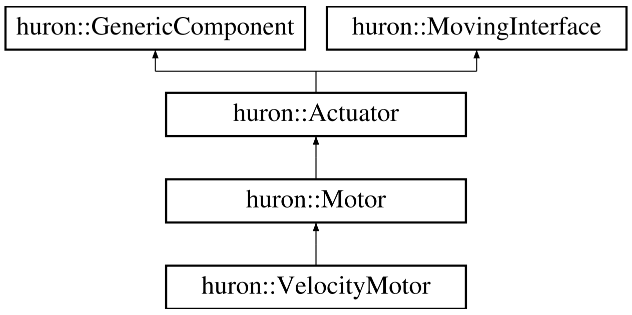
Supports further inheritance.

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

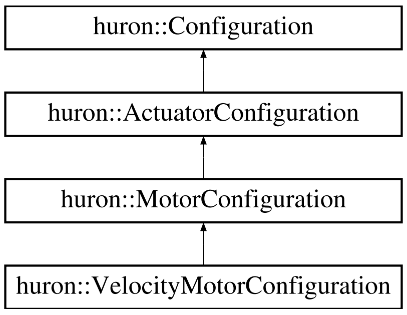
Additional Inherited Members

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:


huron::Configuration
huron::ActuatorConfiguration
huron::MotorConfiguration
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()

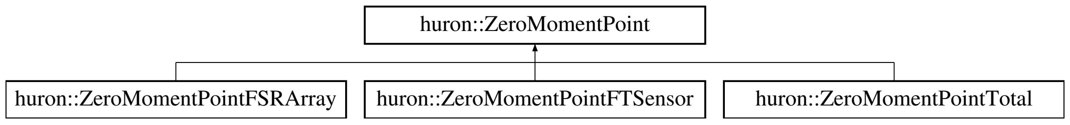
```
huron::VelocityMotorConfiguration::VelocityMotorConfiguration (
            ConfigMap config_map,
            std::set< std::string > valid_keys )  [inline]
```

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:


huron::ZeroMomentPoint
huron::ZeroMomentPointFSRArray
huron::ZeroMomentPointFTSensor
huron::ZeroMomentPointTotal

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

zmp	The zero moment point in the ZMP frame.
fz	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

zmp	The zero moment point in the ZMP frame.

Returns

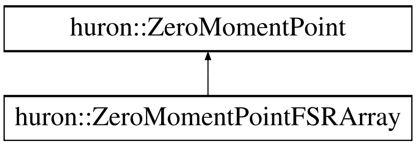
The zero moment point in the world frame.

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

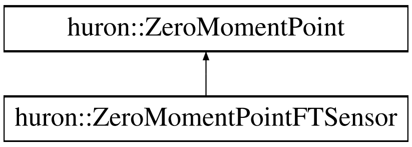
zmp	The zero moment point in the ZMP frame.
fz	The normal force.

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

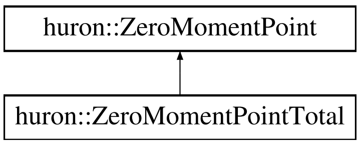
zmp	The zero moment point in the ZMP frame.
fz	The normal force.

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

zmp	The zero moment point in the ZMP frame.
fz	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

```
#pragma once

#include <memory>
#include <utility>

namespace huron {

template <typename ClassWithProtectedCtor>
class enable_protected_make_shared {
 protected:
  template <typename... Args>
  static std::shared_ptr<ClassWithProtectedCtor> make_shared(Args &&... args) {
    class make_shared_enabler : public ClassWithProtectedCtor {
     public:
      // Ensures that the constructor is not public.
      static_assert(!std::is_constructible_v<ClassWithProtectedCtor, Args...>);
      explicit make_shared_enabler(Args &&... args)
        : ClassWithProtectedCtor(std::forward<Args>(args)...) {}
    };
    return std::make_shared<make_shared_enabler>(std::forward<Args>(args)...);
  }
};

}  // namespace huron
```

9.2 enable_protected_make_unique.h

```
#pragma once

#include <memory>
#include <utility>

namespace huron {

template <typename ClassWithProtectedCtor>
class enable_protected_make_unique {
 protected:
  template <typename... Args>
  static std::unique_ptr<ClassWithProtectedCtor> make_unique(Args &&... args) {
    class make_unique_enabler : public ClassWithProtectedCtor {
     public:
      // Ensures that the constructor is not public.
      static_assert(!std::is_constructible_v<ClassWithProtectedCtor, Args...>);
      explicit make_unique_enabler(Args &&... args)
        : ClassWithProtectedCtor(std::forward<Args>(args)...) {}
    };
    return std::make_unique<make_unique_enabler>(std::forward<Args>(args)...);
  }
};

}  // namespace huron
```

9.3 types.h

```
#pragma once

#include <eigen3/Eigen/Core>
```

```

namespace huron {

typedef Eigen::Matrix< double, 6, 1 > Vector6d;
typedef Eigen::Matrix< double, 6, 6 > Matrix6d;
typedef Eigen::Matrix< double, 6, Eigen::Dynamic > Matrix6Xd;

}  // namespace huron
```

9.4 push_recovery.h

```
#pragma once

#include <eigen3/Eigen/Dense>

#include <iostream>
#include <complex>
#include <vector>

class PushRecoveryControl {
 private:
  static const inline std::complex<double> i{0.0, 1.0};
  // EOM of 3 DOF model
  // Mass in kg, length in meter
  static constexpr double alpha = 0.7;
  static constexpr double m1 = 5.9117,
        m2 = 4.2554,
        m3 = 10.19329;

  static constexpr double lc1 = 0.15149,
        lc2 = 0.24517,
        lc3 = 0.1585;

  static constexpr double l1 = 0.3715,
        l2 = 0.49478,
        l3 = 0.32662;

  static constexpr double g = 9.81;
  static constexpr double I1 = 0.0222,
        I2 = 0.01009,
        I3 = 0.0219;

  // Desired position, velocity and acceleration of location of the com

  double theta1_d = 0,
        theta2_d = 0,
        theta3_d = 0;

  double theta1_dot_d = 0,
        theta2_dot_d = 0,
        theta3_dot_d = 0;

  double theta1_dddot = 0,
        theta2_dddot = 0,
        theta3_dddot = 0;

  double x_com_d = 0,
        x_com_ddot = 0,
        x_com_dddot = 0;

  // Values
  double theta1, theta2, theta3 = 0;
  double theta1_dot, theta2_dot, theta3_dot = 0;
  double X_COM, X_dot_COM = 0;
  Eigen::MatrixXd ModelCalculation();
  Eigen::MatrixXd CalculateCOM();
  template <typename T>
  int sign (const T &val) { return (val > 0) - (val < 0); }
  Eigen::MatrixXd SMCController(const Eigen::Vector2d& cop,
                              const Eigen::MatrixXd& J_X_COM,
                              const Eigen::MatrixXd& J_X_COM_dot);
  Eigen::MatrixXd SMCPostureCorrection();
  double constrainAngle(double x);

 public:
  Eigen::MatrixXd GetTorque(const Eigen::Vector2d& cop,
                          const Eigen::VectorXd& position,
                          const Eigen::VectorXd& velocity);
};
```

9.5 can_helpers.h

```
/*
 * Original source from: https://github.com/odriverobotics/ODrive/tree/master
 * */
#pragma once

#include <stdint.h>
#include <algorithm>
#include <cstring>
#include <iterator>

struct can_Message_t {
  uint32_t id = 0x000;  // 11-bit max is 0x7ff, 29-bit max is 0x1FFFFFFF
  bool isExt = false;
  bool rtr = false;
  uint8_t len = 8;
  uint8_t buf[8] = {0, 0, 0, 0, 0, 0, 0, 0};
};

struct can_Signal_t {
  const uint8_t startBit;
  const uint8_t length;
  const bool isIntel;
  const float factor;
  const float offset;
};

struct can_Cyclic_t {
  uint32_t cycleTime_ms;
  uint32_t lastTime_ms;
};

template <typename T>
constexpr T can_getSignal(can_Message_t msg, const uint8_t startBit,
              const uint8_t length, const bool isIntel) {
  uint64_t tempVal = 0;
  uint64_t mask = length < 64 ? (1ULL « length) - 1ULL : -1ULL;

  if (isIntel) {
    std::memcpy(&tempVal, msg.buf, sizeof(tempVal));
    tempVal = (tempVal » startBit) & mask;
  } else {
    std::reverse(std::begin(msg.buf), std::end(msg.buf));
    std::memcpy(&tempVal, msg.buf, sizeof(tempVal));
    tempVal = (tempVal » (64 - startBit - length)) & mask;
  }

  T retVal;
  std::memcpy(&retVal, &tempVal, sizeof(T));
  return retVal;
}

template <typename T>
constexpr void can_setSignal(can_Message_t& msg, const T& val,
                 const uint8_t startBit, const uint8_t length,
                 const bool isIntel) {
  uint64_t valAsBits = 0;
  std::memcpy(&valAsBits, &val, sizeof(val));

  uint64_t mask = length < 64 ? (1ULL « length) - 1ULL : -1ULL;

  if (isIntel) {
    uint64_t data = 0;
    std::memcpy(&data, msg.buf, sizeof(data));

    data &= ~(mask « startBit);
    data |= valAsBits « startBit;

    std::memcpy(msg.buf, &data, sizeof(data));
  } else {
    uint64_t data = 0;
    std::reverse(std::begin(msg.buf), std::end(msg.buf));
    std::memcpy(&data, msg.buf, sizeof(data));

    data &= ~(mask « (64 - startBit - length));
    data |= valAsBits « (64 - startBit - length);

    std::memcpy(msg.buf, &data, sizeof(data));
    std::reverse(std::begin(msg.buf), std::end(msg.buf));
  }
}

template<typename T>
void can_setSignal(can_Message_t& msg, const T& val, const uint8_t startBit,
           const uint8_t length, const bool isIntel, const float factor,
           const float offset) {
```

```
  T scaledVal = static_cast<T>((val - offset) / factor);
  can_setSignal<T>(msg, scaledVal, startBit, length, isIntel);
}

template<typename T>
float can_getSignal(can_Message_t msg, const uint8_t startBit,
            const uint8_t length, const bool isIntel,
            const float factor, const float offset) {
  T retVal = can_getSignal<T>(msg, startBit, length, isIntel);
  return (retVal * factor) + offset;
}

template <typename T>
float can_getSignal(can_Message_t msg, const can_Signal_t& signal) {
  return can_getSignal<T>(msg, signal.startBit, signal.length, signal.isIntel,
              signal.factor, signal.offset);
}

template <typename T>
void can_setSignal(can_Message_t& msg, const T& val,
          const can_Signal_t& signal) {
  can_setSignal(msg, val, signal.startBit, signal.length, signal.isIntel,
        signal.factor, signal.offset);
}
```

9.6 canbus.h

```
/*
 * Original source from: https://github.com/odriverobotics/ODrive/tree/master
 * */
#pragma once

#include <variant>
#include "can_helpers.h"

namespace huron {
namespace driver {
namespace can {

struct MsgIdFilterSpecs {
  std::variant<uint16_t, uint32_t> id;
  uint32_t mask;
};

class BusBase {
 public:
  typedef void(*on_can_message_cb_t)(void* ctx, const can_Message_t& message);
  struct CanSubscription {};

  BusBase() = default;
  BusBase(const BusBase&) = delete;
  BusBase& operator=(const BusBase&) = delete;
  virtual ~BusBase() = default;

  virtual bool send_message(const can_Message_t& message) = 0;

  virtual bool recv_message(can_Message_t& message,
                uint32_t timeout = UINT32_MAX) = 0;

  virtual bool subscribe(const MsgIdFilterSpecs& filter,
              on_can_message_cb_t callback, void* ctx,
              CanSubscription** handle) = 0;

  virtual bool unsubscribe(CanSubscription* handle) = 0;
};

}  // namespace can
}  // namespace driver
}  // namespace huron
```

9.7 socket_can_bus.h

```
#pragma once

#include <string>
#include <sockcanpp/CanDriver.hpp>

#include "canbus.h"

#define CAN_CLK_HZ (16000000)
#define CAN_CLK_MHZ (16)

```

```
namespace huron {
namespace driver {
namespace can {

// Anonymous enum for defining the most common CAN baud rates
enum {
  CAN_BAUD_125K = 125000,
  CAN_BAUD_250K = 250000,
  CAN_BAUD_500K = 500000,
  CAN_BAUD_1000K = 1000000,
  CAN_BAUD_1M = 1000000
};

class SocketCanBus : public BusBase {
 public:
  static constexpr sockcanpp::milliseconds kRecvTimeout{1000};
  // struct Config_t {
  //    uint32_t baud_rate = CAN_BAUD_250K;
  //    Protocol protocol = PROTOCOL_SIMPLE;
  //
  //    Bus* parent = nullptr; // set in apply_config()
  //    void set_baud_rate(uint32_t value) { parent->set_baud_rate(value); }
  // };

  SocketCanBus(std::string can_if, uint32_t axis_id)
    : can_if_(can_if), axis_id_(axis_id), recv_timeout_(kRecvTimeout) {}
  SocketCanBus(const SocketCanBus&) = delete;
  SocketCanBus& operator=(const SocketCanBus&) = delete;
  ~SocketCanBus() = default;

  std::string can_if_;
  uint32_t axis_id_;
  sockcanpp::milliseconds recv_timeout_;
  sockcanpp::CanDriver can_driver_{can_if_, CAN_RAW};

  static const uint8_t kCanFifoNone = 0xff;

  bool send_message(const can_Message_t& message) final;
  bool recv_message(can_Message_t& message,
            uint32_t timeout = UINT32_MAX) final;
  bool subscribe(const MsgIdFilterSpecs& filter,
         on_can_message_cb_t callback, void* ctx,
         CanSubscription** handle) final;
  bool unsubscribe(CanSubscription* handle) final;
};

}  // namespace can
}  // namespace driver
}  // namespace huron
```

9.8 config.h

```
#pragma once

#include "odrive_config.h"
```

9.9 odrive_config.h

```
#pragma once

#define AXIS_COUNT (2)

#define ODRIVE_VELOCITY_LIMIT (15.0)
#define ODRIVE_CURRENT_LIMIT (70.0)
```

9.10 serial.h

```
#pragma once

#include <cstddef>
#include <cstdint>
#include <vector>
#include <string>

namespace huron {
namespace driver {
namespace serial {

enum class Parity {
  None = 0,
  Odd = 1,
  Even = 2,
  Mark = 3,
  Space = 4
};

enum class StopBits {
  One = 1,
  Two = 2,
  OnePointFive
};

enum class FlowControl {
  None = 0,
  Software,
  Hardware
};

class SerialBase {
 public:
  SerialBase(std::string port,
             uint32_t baudrate,
             Parity parity,
             StopBits stopbits,
             FlowControl flowcontrol) :
    port_(port),
    baudrate_(baudrate),
    parity_(parity),
    stopbits_(stopbits),
    flowcontrol_(flowcontrol) {}
  SerialBase(const SerialBase&) = delete;
  SerialBase& operator=(const SerialBase&) = delete;
  virtual ~SerialBase() = default;

  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:
  std::string port_;
  uint32_t baudrate_;
  Parity parity_;
  StopBits stopbits_;
  FlowControl flowcontrol_;
};

}  // namespace serial
}  // namespace driver
}  // namespace huron
```

9.11 wjwwood_serial.h

```cpp
#pragma once

#include <serial/serial.h>

#include <memory>
#include <vector>
#include <string>

#include "serial.h"

namespace huron {
namespace driver {
namespace serial {

class Serial : public SerialBase{
 public:
  Serial(std::string port,
         uint32_t baudrate,
         Parity parity,
         StopBits stopbits,
         FlowControl flowcontrol);
  Serial(const Serial&) = delete;
  Serial& operator=(const Serial&) = delete;
  virtual ~Serial() = default;

  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;

 private:
  std::unique_ptr<::serial::Serial> wjwwood_serial_;

  static ::serial::flowcontrol_t ConvertFlowControl(FlowControl flowcontrol) {
    return static_cast<::serial::flowcontrol_t>(flowcontrol);
  }
  static ::serial::parity_t ConvertParity(Parity parity) {
    return static_cast<::serial::parity_t>(parity);
  }
  static ::serial::stopbits_t ConvertStopBits(StopBits stopbits) {
    return static_cast<::serial::stopbits_t>(stopbits);
  }
};

}  // namespace serial
}  // namespace driver
}  // namespace huron
```

9.12 rotation.h

```
#pragma once

#include <eigen3/Eigen/Core>
#include <cmath>

namespace huron {
namespace math {

Eigen::Vector3d ZyxToRpy(
  const Eigen::Ref<const Eigen::Vector3d>& zyx);

Eigen::Vector3d RpyToZyx(
  const Eigen::Ref<const Eigen::Vector3d>& zyx);

Eigen::Matrix3d ZyxToRotationMatrix(
  const Eigen::Ref<const Eigen::Vector3d>& zyx);

Eigen::Vector3d RotationMatrixToZyx(
  const Eigen::Ref<const Eigen::Matrix3d>& rotation_matrix);

Eigen::Matrix3d RpyToRotationMatrix(
  const Eigen::Ref<const Eigen::Vector3d>& rpy);

Eigen::Vector3d RotationMatrixToRpy(
  const Eigen::Ref<const Eigen::Matrix3d>& rotation_matrix);


}  // namespace math
}  // namespace huron
```

9.13 com_frame.h

```
#pragma once

#include <string>
#include <memory>

#include "huron/multibody/frame.h"

namespace huron {
namespace multibody {

class ComFrame : public Frame {
 public:
  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;
  ~ComFrame() override = default;

  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;

 private:
  FrameIndex parent_frame_index_;

  Eigen::Affine3d ParentToThisTransform() const;
};

}  // namespace multibody
}  // namespace huron
```

9.14 frame.h

```
#pragma once

#include <eigen3/Eigen/Dense>

#include <memory>
#include <string>

#include "huron/enable_protected_make_shared.h"

```

```
namespace huron {
namespace multibody {

class Model;

using FrameIndex = size_t;

enum class FrameType {
  kLogical,
  kFixed,
  kJoint,
  kSensor,
  kPhysical,
};

class Frame : public enable_protected_make_shared<Frame> {
 public:
  friend class Model;

  Frame(const Frame&) = delete;
  Frame& operator=(const Frame&) = delete;
  virtual ~Frame() = default;

  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 { return name_; }
  FrameIndex index() const { return index_; }
  FrameType type() const { return type_; }
  bool is_user_defined() const { return is_user_defined_; }

 protected:
  Frame(FrameIndex index,
        const std::string& name,
        FrameType type,
        bool is_user_defined,
        std::weak_ptr<const Model> model);

  const FrameIndex index_;
  const std::string name_;
  const FrameType type_;
  bool is_user_defined_;
  const std::weak_ptr<const Model> model_;
};

}  // namespace multibody
}  // namespace huron
```

9.15 joint_common.h

```
#pragma once

#include <eigen3/Eigen/Dense>

#include <stddef.h>
#include <string>
#include <cassert>
#include <limits>

#include "frame.h"

namespace huron {
namespace multibody {

using JointIndex = size_t;

enum class JointType {
  kUnknown = 0,
  kFixed,
  kRevolute,
  kPrismatic,
  kPlanar,
  kSpherical,
  kFreeFlyer
};

struct JointDescription {
 public:
  // TODO(dtbpkmte): Properly implement JointIndex/FrameIndex and casts.
  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)
      : id_((JointIndex) id),
        name_(name),
        parent_frame_id_((FrameIndex) parent_frame_id),
        child_frame_id_((FrameIndex) child_frame_id),
        num_positions_(num_positions),
        num_velocities_(num_velocities),
        type_(type),
        min_position_(min_position),
        max_position_(max_position),
        min_velocity_(min_velocity),
        max_velocity_(max_velocity),
        min_acceleration_(min_acceleration),
        max_acceleration_(max_acceleration),
        min_torque_(min_torque),
        max_torque_(max_torque),
        friction_(friction),
        damping_(damping) {
    assert(min_position.size() == num_positions_);
    assert(max_position.size() == num_positions_);
    assert((max_position.array() >= min_position.array()).all());

    assert(min_velocity.size() == num_velocities_);
    assert(max_velocity.size() == num_velocities_);
    assert((max_velocity.array() >= min_velocity.array()).all());

    assert(min_acceleration.size() == num_velocities_);
    assert(max_acceleration.size() == num_velocities_);
    assert((max_acceleration.array() >= min_acceleration.array()).all());

    assert(min_torque.size() == num_velocities_);
    assert(max_torque.size() == num_velocities_);
    assert((max_torque.array() >= min_torque.array()).all());

    assert(friction.size() == num_velocities_);
    assert(damping.size() == num_velocities_);
  }

  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(id, name,
                         parent_frame_id, child_frame_id,
                         num_positions, num_velocities,
                         type,
                         min_position, max_position,
                          min_velocity, max_velocity,
                          min_acceleration, max_acceleration,
                          min_torque, max_torque,
                          Eigen::VectorXd::Zero(num_velocities),
                          Eigen::VectorXd::Zero(num_velocities)) {}

   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)
       : JointDescription(
           id, name,
           parent_frame_id, child_frame_id,
           num_positions, num_velocities,
           type,
           Eigen::VectorXd::Constant(num_positions,
             -std::numeric_limits<double>::infinity()),
           Eigen::VectorXd::Constant(num_positions,
             std::numeric_limits<double>::infinity()),
           Eigen::VectorXd::Constant(num_velocities,
             -std::numeric_limits<double>::infinity()),
           Eigen::VectorXd::Constant(num_velocities,
             std::numeric_limits<double>::infinity()),
           Eigen::VectorXd::Constant(num_velocities,
             -std::numeric_limits<double>::infinity()),
           Eigen::VectorXd::Constant(num_velocities,
             std::numeric_limits<double>::infinity()),
           Eigen::VectorXd::Constant(num_velocities,
             -std::numeric_limits<double>::infinity()),
           Eigen::VectorXd::Constant(num_velocities,
             std::numeric_limits<double>::infinity()),
           Eigen::VectorXd::Zero(num_velocities),
           Eigen::VectorXd::Zero(num_velocities)) {}

   JointIndex id() const { return id_; }
   const std::string& name() const { return name_; }
   FrameIndex parent_frame_id() const { return parent_frame_id_; }
   FrameIndex child_frame_id() const { return child_frame_id_; }
   size_t num_positions() const { return num_positions_; }
   size_t num_velocities() const { return num_velocities_; }
   JointType type() const { return type_; }
   const Eigen::VectorXd& min_position() const { return min_position_; }
   const Eigen::VectorXd& max_position() const { return max_position_; }
   const Eigen::VectorXd& min_velocity() const { return min_velocity_; }
   const Eigen::VectorXd& max_velocity() const { return max_velocity_; }
   const Eigen::VectorXd& min_acceleration() const { return min_acceleration_; }
   const Eigen::VectorXd& max_acceleration() const { return max_acceleration_; }
   const Eigen::VectorXd& min_torque() const { return min_torque_; }
   const Eigen::VectorXd& max_torque() const { return max_torque_; }
   const Eigen::VectorXd& friction() const { return friction_; }
   const Eigen::VectorXd& damping() const { return damping_; }

  private:
   JointIndex id_;
   std::string name_;
   FrameIndex parent_frame_id_;
   FrameIndex child_frame_id_;
   size_t num_positions_;
   size_t num_velocities_;
   JointType type_;
   Eigen::VectorXd min_position_;
   Eigen::VectorXd max_position_;
   Eigen::VectorXd min_velocity_;
   Eigen::VectorXd max_velocity_;
   Eigen::VectorXd min_acceleration_;
   Eigen::VectorXd max_acceleration_;
   Eigen::VectorXd min_torque_;
   Eigen::VectorXd max_torque_;
   Eigen::VectorXd friction_;
   Eigen::VectorXd damping_;
 };

 std::ostream& operator«(std::ostream &os, const JointDescription &jd);

 }  // namespace multibody
 }  // namespace huron
```

9.16 logical_frame.h

```
#pragma once

#include <string>
#include <memory>

#include "huron/multibody/frame.h"

namespace huron {
namespace multibody {

class LogicalFrame : public Frame, enable_protected_make_shared<LogicalFrame> {
 public:
  friend class Model;

  LogicalFrame(const LogicalFrame&) = delete;
  LogicalFrame& operator=(const LogicalFrame&) = delete;
  ~LogicalFrame() override = default;

  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:
  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);

 private:
  FrameIndex parent_frame_index_;
  const std::function<Eigen::Affine3d(const Eigen::Affine3d&)>
    transform_function_;
};

}  // namespace multibody
}  // namespace huron
```

9.17 model.h

```
#pragma once

#include <utility>
#include <memory>
#include <vector>
#include <string>
#include <unordered_map>

#include "huron/multibody/model_impl_types.h"
#include "huron/multibody/model_impl_interface.h"
#include "huron/multibody/frame.h"

namespace huron {
namespace multibody {

class Model : public std::enable_shared_from_this<Model> {
  using ModelImplInterface = internal::ModelImplInterface;

 public:
  Model();
  Model(const Model&) = delete;
  Model& operator=(const Model&) = delete;
  ~Model() = default;

  void AddModelImpl(ModelImplType type, bool set_as_default = false);

  template<typename ...Args>
  void AddJoint(JointIndex index,
                Args&&... args) {
    assert(!is_constructed_);
    assert(!is_finalized_);
    if (joints_[index] != nullptr) {
      // TODO(dtbpkmte): provide index information in the error message.
      throw std::runtime_error("Joint already exists at this index.");
    }
    joints_[index] = std::make_shared<Joint>(std::forward<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);

  template<typename FrameImpl, typename ...Args>
  std::weak_ptr<const Frame> AddFrame(const std::string& name, Args&&... args) {
    static_assert(
      std::is_base_of_v<Frame, FrameImpl>,
      "Invalid frame type.");
    static_assert(
      std::is_base_of_v<enable_protected_make_shared<FrameImpl>, FrameImpl>,
      "Frame-derived class must also derive from "
      "enable_protected_make_shared.");
    assert(is_constructed_);
    assert(!is_finalized_);

    DoAddFrame<FrameImpl>(name, true, std::forward<Args>(args)...);
    return frames_.back();
  }

  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) {
     default_impl_index_ = index;
   }
   size_t GetDefaultModelImpl() const {
     return default_impl_index_;
   }


   // Kinematics and Dynamics wrapper functions
   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;

   const Eigen::VectorXd& GetTorques() const;

   const Eigen::MatrixXd& GetMassMatrix() const;

   const Eigen::MatrixXd& GetCoriolisMatrix() const;

   const Eigen::VectorXd& GetNonlinearEffects() const;

   const Eigen::VectorXd& GetGravity() const;

   const huron::Vector6d& GetSpatialMomentum() const;

   huron::Vector6d GetCentroidalMomentum() const;

   const huron::Matrix6Xd& GetCentroidalMatrix() const;

   void ComputeAll();

   void ForwardKinematics();

   bool is_finalized() const {
     return is_finalized_;
   }

   size_t num_positions() const {
     return num_positions_;
   }

   size_t num_velocities() const {
     return num_velocities_;
   }

   size_t num_joints() const {
     return joints_.size();
   }

   size_t num_frames() const {
     return frames_.size();
   }

  protected:
   ModelImplInterface const * GetModelImpl(size_t index) const;

   // template<typename FrameImpl, typename ...Args>
   // void DoAddFrame(const std::string& name, bool is_user_defined,
   //                 Args&&... args);
   template<typename FrameImpl, typename ...Args>
   void DoAddFrame(const std::string& name, bool is_user_defined,
                           Args&&... args) {
     // Check if the frame name already exists
     if (frame_name_to_index_.find(name) != frame_name_to_index_.end()) {
       throw std::runtime_error("Frame name already exists.");
     }
     frames_.push_back(FrameImpl::make_shared(
                         frames_.size(),  // frame index
                         name,  // frame name
                         is_user_defined,
                         weak_from_this(),  // model
                         std::forward<Args>(args)...));
     frame_name_to_index_[name] = frames_.size() - 1;
   }

   void DoAddFrameFromModelDescription(FrameIndex idx,
                                       const std::string& name,
                                       FrameType type);

   size_t default_impl_index_ = 0;
   std::vector<std::unique_ptr<ModelImplInterface» impls_;
   std::vector<std::shared_ptr<Joint» joints_;
   Eigen::VectorXd states_;
   size_t num_positions_ = 0;
   size_t num_velocities_ = 0;

   std::vector<std::shared_ptr<Frame» frames_;
   std::unordered_map<std::string, FrameIndex> frame_name_to_index_;

   bool is_constructed_ = false;
   bool is_finalized_ = false;
 };

 }  // namespace multibody
 }  // namespace huron
```

9.18 model_composite.h

```
#pragma once

#include <vector>
#include <memory>

#include "model.h"

namespace huron {
namespace multibody {

class ModelComposite final : public class Model {
 public:
  ModelComposite();

  void RegisterModel(std::unique_ptr<Model> model);

 private:
  std::vector<std::unique_ptr<Model» models_;
};

}  // namespace multibody
}  // namespace huron
```

9.19 model_impl_factory.h

```
#pragma once

#include <memory>

#include "model_impl_types.h"
#include "model_impl_interface.h"
#include "pinocchio_model_impl.h"

namespace huron {
namespace multibody {
namespace internal {

class ModelImplFactory final {
  friend class multibody::Model;
 public:
  ModelImplFactory() = delete;
  ModelImplFactory(const ModelImplFactory&) = delete;
  ModelImplFactory& operator=(const ModelImplFactory&) = delete;
  ~ModelImplFactory() = default;
 private:
  static std::unique_ptr<internal::ModelImplInterface>
  Create(ModelImplType type) {
    switch (type) {
      case ModelImplType::kPinocchio:
        return std::make_unique<internal::PinocchioModelImpl>();
      default:
        throw std::runtime_error("ModelImplType not implemented.");
    }
  }
};

}  // namespace internal
}  // namespace multibody
}  // namespace huron
```

9.20 model_impl_interface.h

```
#pragma once

#include <vector>
#include <memory>
#include <string>

#include "huron/types.h"
#include "huron/control_interfaces/joint.h"
#include "joint_common.h"

namespace huron {
namespace multibody {
namespace internal {

class ModelImplInterface {
 public:
  ModelImplInterface() = default;
  ModelImplInterface(const ModelImplInterface&) = delete;
  ModelImplInterface& operator=(const ModelImplInterface&) = delete;
  virtual ~ModelImplInterface() = default;

  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;

  virtual const Eigen::VectorXd& GetTorques() const;

  virtual const Eigen::MatrixXd& GetMassMatrix() const;

  virtual const Eigen::MatrixXd& GetCoriolisMatrix() const;

  virtual const Eigen::VectorXd& GetNonlinearEffects() const;

  virtual const Eigen::VectorXd& GetGravity() const;

  virtual const huron::Vector6d& GetSpatialMomentum() const;

  virtual huron::Vector6d GetCentroidalMomentum() const;

  virtual const huron::Matrix6Xd& GetCentroidalMatrix() const;

  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;
};

}  // namespace internal
}  // namespace multibody
}  // namespace huron
```

9.21 model_impl_types.h

```
#pragma once

namespace huron {
namespace multibody {

enum class ModelImplType {
  kPinocchio,
};

}  // namespace multibody
}  // namespace huron
```

9.22 pinocchio_model_impl.h

```
#pragma once

#include <experimental/propagate_const>
#include <vector>
#include <string>
#include <memory>

#include "huron/multibody/model_impl_interface.h"
#include "huron/multibody/joint_common.h"
#include "huron/exceptions/not_implemented_exception.h"

namespace huron {
namespace multibody {
namespace internal {

class PinocchioModelImpl : public ModelImplInterface {
 public:
  PinocchioModelImpl();
  PinocchioModelImpl(const PinocchioModelImpl&) = delete;
  PinocchioModelImpl& operator=(const PinocchioModelImpl&) = delete;
  ~PinocchioModelImpl() override;

  static bool IsAvailable() { return true; }

  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;
  const Eigen::VectorXd& GetTorques() const override;
  const Eigen::MatrixXd& GetMassMatrix() const override;
  const Eigen::MatrixXd& GetCoriolisMatrix() const override;
  const Eigen::VectorXd& GetNonlinearEffects() const override;
  const Eigen::VectorXd& GetGravity() const override;
  const huron::Vector6d& GetSpatialMomentum() const override;
  huron::Vector6d GetCentroidalMomentum() const override;
  const huron::Matrix6Xd& GetCentroidalMatrix() const override;

  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 { return is_built_; }
  size_t num_positions() const override { return num_positions_; }
  size_t num_velocities() const override { return num_velocities_; }
  size_t num_joints() const override { return num_joints_; }
  size_t num_frames() const override { return num_frames_; }

 private:
  struct Impl;
  std::experimental::propagate_const<std::unique_ptr<Impl» impl_;

  bool is_built_ = false;
  size_t num_positions_ = 0;
  size_t num_velocities_ = 0;
  size_t num_joints_ = 0;
  size_t num_frames_ = 0;
};

}  // namespace internal
}  // namespace multibody
}  // namespace huron
```

9.23 force_torque_sensor.h

```
#pragma once

#include <memory>
#include <utility>

#include "huron/sensors/force_torque_sensor.h"

namespace huron {
namespace ros2 {

class HuronNode;

class ForceTorqueSensor : public huron::ForceTorqueSensor {
  friend class HuronNode;
 public:
  ForceTorqueSensor(bool reverse_wrench_direction,
                    std::weak_ptr<const multibody::Frame> frame);
  ForceTorqueSensor(const ForceTorqueSensor&) = delete;
  ForceTorqueSensor& operator=(const ForceTorqueSensor&) = delete;
  ~ForceTorqueSensor() override = default;

  void Initialize() override;
  void SetUp() override;
  void Terminate() override;

 protected:
  Vector6d DoGetWrenchRaw() override;

 private:
  size_t index_;
  std::weak_ptr<const HuronNode> node_;

  void SetNode(std::weak_ptr<HuronNode> node) {
    node_ = std::move(node);
  }

  void SetIndex(size_t index) {
    index_ = index;
  }
};

}  // namespace ros2
}  // namespace huron
```

9.24 force_torque_sensor.h

```
#pragma once

#include <eigen3/Eigen/Core>

#include <memory>

#include "huron/control_interfaces/sensor_with_frame.h"
#include "huron/types.h"

namespace huron {

class ForceTorqueSensor : public SensorWithFrame {
 public:
  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;
  ~ForceTorqueSensor() override = default;

  void RequestStateUpdate() override;

  void GetNewState(Eigen::Ref<Eigen::MatrixXd> new_state) const override;
  Eigen::VectorXd GetValue() const override;

 protected:
  virtual Vector6d DoGetWrenchRaw() = 0;

  bool reverse_wrench_direction_;

 private:
  huron::Vector6d wrench_;
};

}  // namespace huron
```

9.25 huron.h

```
#pragma once

#include "huron_node.h"

#include <string>
#include <vector>
#include <memory>

#include "huron/control_interfaces/legged_robot.h"

namespace huron {
namespace ros2 {

class Huron : public huron::LeggedRobot {
 public:
  Huron(std::shared_ptr<HuronNode> node,
        std::unique_ptr<huron::RobotConfiguration> config);
  explicit Huron(std::shared_ptr<HuronNode> node);

  Huron(const Huron&) = delete;
  Huron& operator=(const Huron&) = delete;
  ~Huron() override = default;

  // GenericComponent interface
  void Initialize() override;
  void SetUp() override;
  void Terminate() override;

  // ROS-specific
  void Loop();

 private:
  std::shared_ptr<HuronNode> node_;
};

}  // namespace ros2
}  // namespace huron
```

9.26 huron_node.h

```
#pragma once

#include <eigen3/Eigen/Core>

#include <memory>
#include <vector>
#include <string>

#include <rclcpp/rclcpp.hpp>
#include <sensor_msgs/msg/joint_state.hpp>
#include <std_msgs/msg/float64_multi_array.hpp>
#include <geometry_msgs/msg/wrench_stamped.hpp>
#include <nav_msgs/msg/odometry.hpp>

#include "huron/types.h"


namespace huron {
namespace ros2 {

class ForceTorqueSensor;
class JointGroupController;
class JointStateProvider;

class HuronNode : public rclcpp::Node {
 public:
  HuronNode();
  ~HuronNode() override = default;

  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 {
    return wrenches_[idx];
  }

  void AddJointStateProvider(
    std::shared_ptr<JointStateProvider> jsp,
    const std::string& topic,
    size_t nq, size_t nv,
    bool is_odom = false);
  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();

  Eigen::VectorXd GetJointState(size_t id_q, size_t dim_q,
                                size_t id_v, size_t dim_v) const {
    Eigen::VectorXd state(dim_q + dim_v);
    state.segment(0, dim_q) = joint_state_.segment(id_q, dim_q);
    state.segment(dim_q, dim_v) = joint_state_.segment(nq_ + id_v,
                                                        dim_v);
    return state;
  }

 private:
  bool finalized_ = false;
  size_t nq_ = 0;
  size_t nv_ = 0;

  rclcpp::Subscription<nav_msgs::msg::Odometry>::SharedPtr odom_sub_;
  rclcpp::Subscription<sensor_msgs::msg::JointState>::SharedPtr
    joint_state_sub_;
  std::vector<rclcpp::Publisher<std_msgs::msg::Float64MultiArray>::SharedPtr>
    float64_multi_array_pubs_;
  std::vector<rclcpp::Subscription<geometry_msgs::msg::WrenchStamped>
    ::SharedPtr> wrench_stamped_subs_;

  Eigen::VectorXd joint_state_;
  std::vector<huron::Vector6d> wrenches_;
};

}  // namespace ros2
}  // namespace huron
```

9.27 joint_group_controller.h

```
#pragma once

#include <vector>
#include <memory>
#include <utility>

#include "huron/control_interfaces/moving_interface.h"

namespace huron {
namespace ros2 {

class HuronNode;

class JointGroupController : public huron::MovingInterface {
  friend class HuronNode;
 public:
  explicit JointGroupController(size_t dim);
  JointGroupController(const JointGroupController&) = delete;
  JointGroupController& operator=(const JointGroupController&) = delete;
  ~JointGroupController() override = default;

  bool Move(const std::vector<double>& values) override;
  bool Move(const Eigen::VectorXd& values) override;
  bool Stop() override;

 private:
  std::weak_ptr<HuronNode> node_;
  size_t dim_;
  size_t pub_idx_;

  void SetNode(std::weak_ptr<HuronNode> node) {
    node_ = std::move(node);
  }

  void SetPubIdx(size_t pub_idx) {
    pub_idx_ = pub_idx;
  }
};

}  // namespace ros2
}  // namespace huron
```

9.28 joint_state_provider.h

```
#pragma once

#include <memory>
#include <utility>

#include "huron/control_interfaces/state_provider.h"

namespace huron {
namespace ros2 {

class HuronNode;

class JointStateProvider : public huron::StateProvider {
  friend class HuronNode;
 public:
  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;

 private:
  std::weak_ptr<HuronNode> node_;
  size_t nq_;
  size_t nv_;
  size_t id_q_;
  size_t id_v_;

  void SetNode(std::weak_ptr<HuronNode> node) {
    node_ = std::move(node);
  }
};

}  // namespace ros2
}  // namespace huron
```

9.29 actuator.h

```
#pragma once

#include <vector>
#include <set>
#include <string>
#include <utility>
#include <memory>

#include "huron/control_interfaces/moving_interface.h"
#include "huron/control_interfaces/generic_component.h"

namespace huron {

class ActuatorConfiguration : public Configuration {
 private:
  static const inline std::set<std::string> kActuatorValidKeys{};

 public:
  ActuatorConfiguration(ConfigMap config_map,
                        std::set<std::string> valid_keys)
      : Configuration(config_map, [&valid_keys]() {
                          std::set<std::string> tmp(kActuatorValidKeys);
                          tmp.merge(valid_keys);
                          return tmp;
                        }()) {}

  explicit ActuatorConfiguration(ConfigMap config_map)
      : ActuatorConfiguration(config_map, {}) {}

  ActuatorConfiguration()
      : ActuatorConfiguration({}, {}) {}
};

class Actuator : public GenericComponent, public MovingInterface {
 public:
  Actuator(size_t dim, std::unique_ptr<ActuatorConfiguration> config)
    : GenericComponent(std::move(config)), MovingInterface(dim) {}
  explicit Actuator(size_t dim)
    : Actuator(dim, std::make_unique<ActuatorConfiguration>()) {}
  Actuator(const Actuator&) = delete;
  Actuator& operator=(const Actuator&) = delete;
  ~Actuator() override = default;
};

}  // namespace huron
```

9.30 configuration.h

```
#pragma once

#include <string>
#include <unordered_map>
#include <set>
#include <any>

#include "huron/exceptions/not_implemented_exception.h"

namespace huron {
```

```

typedef std::unordered_map<std::string, std::any> ConfigMap;

class Configuration {
 protected:
  const std::set<std::string> valid_keys_;
  ConfigMap config_map_;

  bool ValidateKey(std::string config_key) {
    return valid_keys_.count(config_key);
  }
  ConfigMap ValidateMap(ConfigMap config_map);
  virtual std::any GetFromComponent(std::string config_key) {
    throw NotImplementedException(__func__);
  }

 public:
  Configuration(ConfigMap config_map, std::set<std::string> valid_keys);
  explicit Configuration(ConfigMap config_map);
  Configuration(const Configuration&) = delete;
  Configuration& operator=(const Configuration&) = delete;
  virtual ~Configuration() = default;

  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);
};

}  // namespace huron
```

9.31 constant_state_provider.h

```
#pragma once

#include "state_provider.h"

namespace huron {

class ConstantStateProvider : public StateProvider {
 public:
  explicit ConstantStateProvider(const Eigen::MatrixXd& state)
     : StateProvider(state.rows(), state.cols()),
       state_(state) {}
  ConstantStateProvider(const ConstantStateProvider&) = delete;
  ConstantStateProvider& operator=(const ConstantStateProvider&) = delete;
  ~ConstantStateProvider() override = default;

  void RequestStateUpdate() override {}
  void GetNewState(Eigen::Ref<Eigen::MatrixXd> new_state) const override {
    new_state = state_;
  }

  void SetState(const Eigen::MatrixXd& state) {
    assert(state.rows() == dim()[0] && state.cols() == dim()[1]);
    state_ = state;
  }

 private:
  Eigen::MatrixXd state_;
};

}  // namespace huron
```

9.32 encoder.h

```
#pragma once

#include <set>
#include <string>
#include <utility>
#include <memory>

#include "huron/control_interfaces/sensor.h"

namespace huron {

class EncoderConfiguration : public Configuration {
 private:
  static const inline std::set<std::string> kEncoderValidKeys{};

 public:
  EncoderConfiguration(ConfigMap config_map,
                          std::set<std::string> valid_keys)
      : Configuration(config_map, [&valid_keys]() {
                           std::set<std::string> tmp(kEncoderValidKeys);
                           tmp.merge(valid_keys);
                           return tmp;
                         }()) {}

  explicit EncoderConfiguration(ConfigMap config_map)
      : EncoderConfiguration(config_map, {}) {}

  EncoderConfiguration()
      : EncoderConfiguration({}, {}) {}
};

class Encoder : public Sensor {
 public:
  Encoder(double gear_ratio, std::unique_ptr<EncoderConfiguration> config)
    : Sensor(2, 1, std::move(config)), gear_ratio_(gear_ratio) {}
  explicit Encoder(double gear_ratio)
    : Encoder(gear_ratio, std::make_unique<EncoderConfiguration>()) {}
  explicit Encoder(std::unique_ptr<EncoderConfiguration> config)
    : Encoder(1.0, std::move(config)) {}
  Encoder() : Encoder(1.0) {}
  Encoder(const Encoder&) = delete;
  Encoder& operator=(const Encoder&) = delete;
  virtual ~Encoder() = default;

  void GetNewState(Eigen::Ref<Eigen::MatrixXd> new_state) const override {
    new_state = Eigen::Vector2d(GetPosition(), GetVelocity());
  }

  virtual double GetPosition() const = 0;
  virtual double GetVelocity() const = 0;

  virtual void Reset() = 0;

 protected:
  double gear_ratio_;
};

}  // namespace huron
```

9.33 generic_component.h

```
#pragma once

#include <memory>
#include <unordered_map>
#include <string>
#include <utility>

#include "huron/control_interfaces/configuration.h"

namespace huron {

class GenericComponent {
 protected:
  std::unique_ptr<Configuration> config_;

  virtual void ConfigureKey(std::string config_key, std::any config_value) {}

  virtual void ConfigureMap(const ConfigMap& config_map) {
    for (auto& pair : config_map) {
      ConfigureKey(pair.first, pair.second);
    }
  }

 public:
  explicit GenericComponent(std::unique_ptr<Configuration> config)
    : config_(std::move(config)) {}
  GenericComponent()
    : GenericComponent(std::make_unique<Configuration>(ConfigMap())) {}
  GenericComponent(const GenericComponent&) = delete;
  GenericComponent& operator=(const GenericComponent&) = delete;
  virtual ~GenericComponent() = default;

  void Configure(std::string config_key, std::any config_value) {
    if (config_->Set(config_key, config_value)) {
      return ConfigureKey(config_key, config_value);
    }
  }
  void Configure(ConfigMap config) {
    if (config_->Set(config)) {
      return ConfigureMap(config);
    }
  }
  void Configure(std::unique_ptr<Configuration> config_ptr) {
    config_ = std::move(config_ptr);
  }

  virtual void Initialize() = 0;
  virtual void SetUp() = 0;
  virtual void Terminate() = 0;
};

}  // namespace huron
```

9.34 joint.h

```
#pragma once

#include <memory>
#include <vector>
#include <set>
#include <string>

#include "huron/multibody/joint_common.h"
#include "huron/control_interfaces/configuration.h"
#include "huron/control_interfaces/state_provider.h"

namespace huron {

class Joint {
  using JointDescription = huron::multibody::JointDescription;
  using JointType = huron::multibody::JointType;

 public:
  explicit Joint(std::unique_ptr<JointDescription> joint_desc,
                 std::shared_ptr<StateProvider> state_provider = nullptr);
  Joint(const Joint&) = delete;
  Joint& operator=(const Joint&) = delete;
  virtual ~Joint() = default;

  void SetIndices(size_t id_q, size_t id_v) {
    id_q_ = id_q;
    id_v_ = id_v;
  }

  void SetStateProvider(std::shared_ptr<StateProvider> state_provider);

  void UpdateState();

  JointType GetJointType() const {
    return jd_->type();
  }

  const Eigen::VectorXd& GetPositions() const {
    return positions_;
  }

  const Eigen::VectorXd& GetVelocities() const {
    return velocities_;
  }

  const JointDescription* const Info() const {
    return jd_.get();
  }

  bool IsFullyConfigured() const {
    return jd_->type() == JointType::kUnknown || state_provider_ != nullptr;
  }

  size_t id_q() const {
    return id_q_;
  }

  size_t id_v() const {
    return id_v_;
  }

  size_t num_positions() const {
    return jd_->num_positions();
  }

  size_t nq() const {
    return num_positions();
  }

  size_t num_velocities() const {
    return jd_->num_velocities();
  }

  size_t nv() const {
    return num_velocities();
  }

 protected:
  std::unique_ptr<JointDescription> jd_;
  Eigen::VectorXd positions_;
  Eigen::VectorXd velocities_;
  size_t id_q_;
  size_t id_v_;

  std::shared_ptr<StateProvider> state_provider_;
};

}  // namespace huron
```

9.35 legged_robot.h

```
#pragma once

#include <eigen3/Eigen/Core>

#include <memory>

#include "robot.h"
#include "huron/locomotion/zero_moment_point.h"

namespace huron {

class LeggedRobot : public Robot {
 public:
  explicit LeggedRobot(std::unique_ptr<RobotConfiguration> config);
  LeggedRobot();
  LeggedRobot(const LeggedRobot&) = delete;
  LeggedRobot& operator=(const LeggedRobot&) = delete;
  ~LeggedRobot() override = default;

  void InitializeZmp(std::shared_ptr<ZeroMomentPoint> zmp);
  Eigen::Vector2d EvalZeroMomentPoint();

 private:
  std::shared_ptr<ZeroMomentPoint> zmp_;
};

}  // namespace huron
```

9.36 limb.h

```
#pragma once

#include <vector>

#include "joint.h"
#include "moving_group_component.h"

namespace huron {

class Limb : public MovingGroupComponent {
 public:
  void Init(std::vector<Joint> joints);
  void AddJoint(Joint& joint);

 private:
  std::vector<Joint> joints_;
};

}  // namespace huron
```

9.37 motor.h

```
#pragma once

#include <vector>
#include <set>
#include <string>
#include <utility>
#include <memory>
```

```
#include "huron/control_interfaces/actuator.h"

namespace huron {

class MotorConfiguration : public ActuatorConfiguration {
 private:
  static const inline std::set<std::string> kMotorValidKeys{};

 public:
  MotorConfiguration(ConfigMap config_map,
                     std::set<std::string> valid_keys)
      : ActuatorConfiguration(config_map, [&valid_keys]() {
                        std::set<std::string> tmp(kMotorValidKeys);
                        tmp.merge(valid_keys);
                        return tmp;
                      }()) {}

  explicit MotorConfiguration(ConfigMap config_map)
      : MotorConfiguration(config_map, {}) {}

  MotorConfiguration()
      : MotorConfiguration({}, {}) {}
};

class Motor : public Actuator {
 public:
  explicit Motor(std::unique_ptr<MotorConfiguration> config,
                 double gear_ratio = 1.0)
    : Actuator(1, std::move(config)) {}
  explicit Motor(double gear_ratio)
    : Motor(std::make_unique<MotorConfiguration>(), gear_ratio) {}
  Motor() : Motor(1.0) {}
  Motor(const Motor&) = delete;
  Motor& operator=(const Motor&) = delete;
  ~Motor() override = default;

  virtual bool Move(double value) = 0;

 private:
  double gear_ratio_;
};

}  // namespace huron
```

9.38 moving_group.h

```
#pragma once

#include <eigen3/Eigen/Dense>

#include <memory>
#include <utility>
#include <vector>

#include "moving_interface.h"

namespace huron {

class MovingGroup : public MovingInterface {
 public:
  MovingGroup();
  MovingGroup(const MovingGroup&) = delete;
  MovingGroup& operator=(const MovingGroup&) = delete;
  ~MovingGroup() override = default;

  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:
  std::vector<std::shared_ptr<MovingInterface>> moving_components_;
  std::vector<size_t> moving_interface_dims_;
};

}  // namespace huron
```

9.39 moving_interface.h

```
#pragma once

#include <eigen3/Eigen/Dense>

#include <vector>
#include <memory>
#include <utility>

namespace huron {

class MovingInterface {
 public:
  explicit MovingInterface(size_t dim) : dim_(dim) {}
  MovingInterface(const MovingInterface&) = delete;
  MovingInterface& operator=(const MovingInterface&) = delete;
  virtual ~MovingInterface() = default;

  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 { return dim_; }

 protected:
  size_t dim_;
};

}  // namespace huron
```

9.40 position_motor.h

```
#pragma once

#include <set>
#include <string>
#include <utility>
#include <memory>

#include "huron/control_interfaces/motor.h"

namespace huron {

class PositionMotorConfiguration : public MotorConfiguration {
 public:
  PositionMotorConfiguration(ConfigMap config_map,
                             std::set<std::string> valid_keys)
    : MotorConfiguration(config_map,
                         [&valid_keys]() {
                           std::set<std::string> tmp(kPositionMotorValidKeys);
                           tmp.merge(valid_keys);
                           return tmp;
                         }()) {}

  PositionMotorConfiguration()
    : PositionMotorConfiguration({}, {}) {}

 private:
  static const inline std::set<std::string> kPositionMotorValidKeys{};
};

class PositionMotor : public Motor {
 public:
  explicit PositionMotor(std::unique_ptr<PositionMotorConfiguration> config,
                         double gear_ratio)
    : Motor(std::move(config), gear_ratio) {}
  explicit PositionMotor(double gear_ratio)
    : Motor(gear_ratio) {}
  PositionMotor() : Motor() {}
  PositionMotor(const PositionMotor&) = delete;
  PositionMotor& operator=(const PositionMotor&) = delete;
  virtual ~PositionMotor() = default;
};

}  // namespace huron
```

9.41 robot.h

```
#pragma once
```

```

#include <vector>
#include <set>
#include <string>
#include <utility>
#include <memory>

#include "huron/control_interfaces/configuration.h"
#include "huron/control_interfaces/generic_component.h"
#include "huron/control_interfaces/moving_group.h"
#include "huron/control_interfaces/joint.h"
#include "huron/multibody/model.h"
#include "huron/multibody/joint_common.h"

namespace huron {

class RobotConfiguration : public Configuration {
 private:
  static const inline std::set<std::string> kRobotValidKeys{};

 public:
  RobotConfiguration(ConfigMap config_map,
                     std::set<std::string> valid_keys)
      : Configuration(config_map, [&valid_keys]() {
                        std::set<std::string> tmp(kRobotValidKeys);
                        tmp.merge(valid_keys);
                        return tmp;
                      }()) {}

  explicit RobotConfiguration(ConfigMap config_map)
      : RobotConfiguration(config_map, {}) {}

  RobotConfiguration()
      : RobotConfiguration({}, {}) {}
};

class Robot : public MovingGroup, public GenericComponent {
  using Model = multibody::Model;

 public:
  explicit Robot(std::unique_ptr<RobotConfiguration> config);
  Robot();
  Robot(const Robot&) = delete;
  Robot& operator=(const Robot&) = delete;
  ~Robot() override = default;

  Model* const GetModel() { return model_.get(); }

  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:
  Robot(std::unique_ptr<RobotConfiguration> config,
        std::shared_ptr<Model> model);
  explicit Robot(std::shared_ptr<Model> model);

  std::shared_ptr<Model> model_;
  std::vector<std::shared_ptr<StateProvider>> non_joint_state_providers_;
  std::vector<std::shared_ptr<StateProvider>> joint_state_providers_;
};

}  // namespace huron
```

9.42 rotary_encoder.h

```
#pragma once

#include <cmath>
#include <set>
#include <string>
#include <utility>
#include <memory>

#include "encoder.h"

namespace huron {

```

```
class RotaryEncoderConfiguration : public EncoderConfiguration {
 public:
  RotaryEncoderConfiguration(ConfigMap config_map,
                             std::set<std::string> valid_keys)
      : EncoderConfiguration(config_map,
                             [&valid_keys]() {
                               std::set<std::string> tmp(kRotEncValidKeys);
                               tmp.merge(valid_keys);
                               return tmp;
                             }()) {}

  explicit RotaryEncoderConfiguration(double cpr)
      : RotaryEncoderConfiguration(
          ConfigMap({{"cpr", cpr}}), {}) {}

 private:
  static const inline std::set<std::string> kRotEncValidKeys{"cpr"};
};

class RotaryEncoder : public Encoder {
 public:
  RotaryEncoder(double gear_ratio,
                std::unique_ptr<RotaryEncoderConfiguration> config)
    : Encoder(gear_ratio, std::move(config)) {
    cpr_ = std::any_cast<double>(config_.get()->Get("cpr"));
  }
  RotaryEncoder(double gear_ratio, double cpr)
      : RotaryEncoder(gear_ratio,
                      std::make_unique<RotaryEncoderConfiguration>(cpr)) {}
  explicit RotaryEncoder(double cpr)
      : RotaryEncoder(1.0, cpr) {}
  RotaryEncoder(const RotaryEncoder&) = delete;
  RotaryEncoder& operator=(const RotaryEncoder&) = delete;
  ~RotaryEncoder() override = default;

  void RequestStateUpdate() final {
    prev_count_ = count_;
    prev_velocity_ = velocity_;
    DoUpdateState();
  }

  double GetCount() const {
    return count_;
  }

  double GetVelocityCount() const {
    return velocity_;
  }

  double GetPrevCount() const {
    return prev_count_;
  }

  double GetCPR() const {
    return cpr_;
  }

  double GetPosition() const override {
    return count_ / cpr_ * 2.0 * M_PI / gear_ratio_;
  }

  double GetAngleDegree() const {
    return count_ / cpr_ * 360.0 / gear_ratio_;
  }

  double GetVelocity() const override {
    return velocity_ / cpr_ * 2 * M_PI / gear_ratio_;
  }

  double GetVelocityDegree() const {
    return velocity_ / cpr_ * 360.0 / gear_ratio_;
  }

  void Reset() override {
    count_ = 0.0;
    prev_count_ = 0.0;
  }

 protected:
  virtual void DoUpdateState() = 0;

  double velocity_ = 0.0;
  double prev_velocity_ = 0.0;
  double count_ = 0.0;
  double prev_count_ = 0.0;
  double cpr_;
};
```

```

}  // namespace huron
```

9.43 sensor.h

```
#pragma once

#include <eigen3/Eigen/Core>

#include <memory>

#include "huron/control_interfaces/generic_component.h"
#include "huron/control_interfaces/state_provider.h"

namespace huron {

class Sensor : public GenericComponent, public StateProvider {
 public:
  Sensor(const Eigen::Vector2i& dim,
         std::unique_ptr<Configuration> config);
  explicit 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 ~Sensor() = default;

  virtual Eigen::VectorXd GetValue() const;
  virtual Eigen::VectorXd ReloadAndGetValue();
};

}  // namespace huron
```

9.44 sensor_with_frame.h

```
#pragma once

#include <eigen3/Eigen/Core>

#include <memory>

#include "huron/control_interfaces/sensor.h"
#include "huron/multibody/frame.h"

namespace huron {

class SensorWithFrame : public Sensor {
  using Frame = multibody::Frame;

 public:
  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;
  ~SensorWithFrame() override = default;

  std::weak_ptr<const Frame> GetSensorFrame() const {
    return frame_;
  }

 private:
  std::weak_ptr<const Frame> frame_;
};

}  // namespace huron
```

9.45 state_provider.h

```
#pragma once

```

```
#include <eigen3/Eigen/Dense>

namespace huron {

class StateProvider {
 public:
  explicit StateProvider(const Eigen::Vector2i& dim)
    : dim_(dim) {}
  StateProvider(int rows, int cols)
    : dim_(rows, cols) {}
  StateProvider(const StateProvider&) = delete;
  StateProvider& operator=(const StateProvider&) = delete;
  virtual ~StateProvider() = default;

  virtual void RequestStateUpdate() = 0;
  virtual void GetNewState(Eigen::Ref<Eigen::MatrixXd> new_state) const = 0;

  const Eigen::Vector2i& dim() const { return dim_; }

 private:
  const Eigen::Vector2i dim_;
};

}  // namespace huron
```

9.46 torque_motor.h

```
#pragma once

#include <set>
#include <string>
#include <utility>
#include <memory>

#include "huron/control_interfaces/motor.h"

namespace huron {

class TorqueMotorConfiguration : public MotorConfiguration {
 public:
  TorqueMotorConfiguration(ConfigMap config_map,
                           std::set<std::string> valid_keys)
    : MotorConfiguration(config_map,
                         [&valid_keys]() {
                           std::set<std::string> tmp(kTorqueMotorValidKeys);
                           tmp.merge(valid_keys);
                           return tmp;
                         }()) {}

  TorqueMotorConfiguration()
    : TorqueMotorConfiguration({}, {}) {}

 private:
  static const inline std::set<std::string> kTorqueMotorValidKeys{};
};

class TorqueMotor : public Motor {
 public:
  TorqueMotor(std::unique_ptr<TorqueMotorConfiguration> config,
              double gear_ratio)
    : Motor(std::move(config), gear_ratio) {}
  explicit TorqueMotor(double gear_ratio)
    : Motor(gear_ratio) {}
  TorqueMotor() : Motor() {}
  TorqueMotor(const TorqueMotor&) = delete;
  TorqueMotor& operator=(const TorqueMotor&) = delete;
  ~TorqueMotor() override = default;
};

}  // namespace huron
```

9.47 velocity_motor.h

```
#pragma once

#include <set>
#include <string>
#include <utility>
#include <memory>

#include "huron/control_interfaces/motor.h"

```

```
namespace huron {

class VelocityMotorConfiguration : public MotorConfiguration {
 public:
  VelocityMotorConfiguration(ConfigMap config_map,
                             std::set<std::string> valid_keys)
    : MotorConfiguration(config_map,
                         [&valid_keys]() {
                           std::set<std::string> tmp(kVelocityMotorValidKeys);
                           tmp.merge(valid_keys);
                           return tmp;
                         }()) {}

  VelocityMotorConfiguration()
    : VelocityMotorConfiguration({}, {}) {}

 private:
  static const inline std::set<std::string> kVelocityMotorValidKeys{};
};

class VelocityMotor : public Motor {
 public:
  VelocityMotor(std::unique_ptr<VelocityMotorConfiguration> config,
                double gear_ratio)
    : Motor(std::move(config), gear_ratio) {}
  explicit VelocityMotor(double gear_ratio)
    : Motor(gear_ratio) {}
  VelocityMotor() : Motor() {}
  VelocityMotor(const VelocityMotor&) = delete;
  VelocityMotor& operator=(const VelocityMotor&) = delete;
  ~VelocityMotor() override = default;
};

}  // namespace huron
```

9.48 invalid_configuration_exception.h

```

#pragma once

#include <stdexcept>
#include <string>

namespace huron {

class InvalidConfigurationException : public std::logic_error {
 private:
  std::string _text;

  InvalidConfigurationException(const char* message, const char* function)
      : std::logic_error("Invalid Configuration provided.") {
    _text = message;
    _text += " : ";
    _text += function;
  }

 public:
  InvalidConfigurationException()
      : InvalidConfigurationException(
            "Invalid Configuration provided.", __FUNCTION__) {}

  explicit InvalidConfigurationException(const char* message)
      : InvalidConfigurationException(message, __FUNCTION__) {}

  virtual const char *what() const throw() {
      return _text.c_str();
  }
};

}  // namespace huron
```

9.49 not_implemented_exception.h

```

#pragma once

#include <stdexcept>
#include <string>

namespace huron {

class NotImplementedException : public std::logic_error {
 private:
  std::string _text;

  NotImplementedException(const char* message, const char* function)
      : std::logic_error("Not Implemented") {
    _text = message;
    _text += " : ";
    _text += function;
  }

 public:
  NotImplementedException()
      : NotImplementedException("Not Implemented", __FUNCTION__) {}

  explicit NotImplementedException(const char* function)
      : NotImplementedException("Not Implemented", function) {}

  virtual const char *what() const throw() {
      return _text.c_str();
  }
};

}  // namespace huron
```

9.50 zero_moment_point.h

```
#pragma once

#include <eigen3/Eigen/Dense>

#include <memory>

#include "huron/sensors/force_torque_sensor.h"
#include "huron/sensors/force_sensing_resistor_array.h"
#include "huron/multibody/logical_frame.h"

namespace huron {

class ZeroMomentPoint {
 public:
  ZeroMomentPoint(std::weak_ptr<const multibody::Frame> zmp_frame,
                  double normal_force_threshold);
  ZeroMomentPoint(const ZeroMomentPoint&) = delete;
  ZeroMomentPoint& operator=(const ZeroMomentPoint&) = delete;
  virtual ~ZeroMomentPoint() = default;

  virtual Eigen::Vector2d Eval(double& fz) = 0;
  Eigen::Vector2d Eval() {
    double fz;
    return Eval(fz);
  }

  Eigen::Affine3d ZmpToWorld(const Eigen::Vector2d& zmp) const;

 protected:
  std::weak_ptr<const multibody::Frame> zmp_frame_;
  double normal_force_threshold_;
};

}  // namespace huron
```

9.51 zero_moment_point_fsr_array.h

```
#pragma once

#include <memory>

#include "huron/locomotion/zero_moment_point.h"

namespace huron {

class ZeroMomentPointFSRArray : public ZeroMomentPoint {
 public:
  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;
  ~ZeroMomentPointFSRArray() override = default;

  Eigen::Vector2d Eval(double& fz) override;

 private:
  std::shared_ptr<ForceSensingResistorArray> fsr_array_;
};

}  // namespace huron
```

9.52 zero_moment_point_ft_sensor.h

```
#pragma once

#include <memory>
#include <vector>

#include "huron/locomotion/zero_moment_point.h"

namespace huron {

class ZeroMomentPointFTSensor : public ZeroMomentPoint {
 public:
  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;
  ~ZeroMomentPointFTSensor() override = default;

  Eigen::Vector2d Eval(double& fz) override;

 private:
  std::vector<std::shared_ptr<ForceTorqueSensor» ft_sensors_;
};

}  // namespace huron
```

9.53 zero_moment_point_total.h

```
#pragma once

#include <memory>
#include <vector>

#include "huron/locomotion/zero_moment_point.h"

namespace huron {

class ZeroMomentPointTotal : public ZeroMomentPoint {
 public:
  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;
  ~ZeroMomentPointTotal() override = default;

  Eigen::Vector2d Eval(double& fz) override;

 private:
  std::vector<std::shared_ptr<ZeroMomentPoint» zmp_vector_;
};

}  // namespace huron
```

9.54 odrive.h

```
#pragma once

#include <cstdint>
#include <set>
#include <string>
#include <utility>
#include <memory>

#include "odrive_enums.h"
#include "huron/control_interfaces/generic_component.h"

namespace huron {
namespace odrive {
```

```
class ODrive : public huron::GenericComponent {
 protected:
  static const uint32_t kGetTimeout = 100;  // ms

  uint32_t get_timeout_;
  bool is_calibrated_ = false;

 public:
  class ODriveConfiguration : public huron::Configuration {
   public:
    ODriveConfiguration(ConfigMap config_map,
                          std::set<std::string> valid_keys)
      : huron::Configuration(config_map,
                             [&valid_keys]() {
                               std::set<std::string> tmp(kODriveKeys);
                               tmp.merge(valid_keys);
                               return tmp;
                             }()) {}

    explicit ODriveConfiguration(ConfigMap config_map)
      : ODriveConfiguration(config_map, std::set<std::string>()) {}

    ODriveConfiguration()
      : ODriveConfiguration(ConfigMap(), std::set<std::string>()) {}

   private:
    static const inline std::set<std::string> kODriveKeys{
      "velocity_limit",
      "current_limit",
      "traj_vel_limit",
      "traj_accel_limit",
      "traj_decel_limit",
      "traj_inertia"};
  };

  ODrive(std::unique_ptr<ODriveConfiguration> config, uint32_t get_timeout)
      : huron::GenericComponent(std::move(config)),
        get_timeout_(get_timeout) {}
  explicit ODrive(uint32_t get_timeout = kGetTimeout)
      : ODrive(std::make_unique<ODriveConfiguration>(),
               get_timeout) {}
  ODrive(const ODrive&) = delete;
  ODrive& operator=(const ODrive&) = delete;
  ~ODrive() override = default;

  void Initialize() override;

  bool Calibrate();

  void ConfigureKey(std::string config_key, std::any config_value) override;

  // Get functions (msg.rtr bit must be set)
  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;
  // msg.rtr bit must NOT be set
  virtual bool GetAdcVoltage(float& adc_voltage) = 0;

  // Set functions
  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;

  // Other functions
  virtual bool Nmt() = 0;
  virtual bool Estop() = 0;
  virtual bool ClearErrors() = 0;
  virtual bool StartAnticogging() = 0;
};

}  // namespace odrive
}  // namespace huron
```

9.55 odrive_can.h

```
#pragma once

#include <memory>
#include <string>

#include "odrive.h"
#include "huron/driver/can/canbus.h"

namespace huron {
namespace odrive {

class ODriveCAN : public ODrive {
 private:
  static const uint32_t kRecvTimeout = 100;  // ms

 public:
  enum {
    MSG_CO_NMT_CTRL = 0x000,  // CANOpen NMT Message REC
    MSG_ODRIVE_HEARTBEAT,
    MSG_ODRIVE_ESTOP,
    MSG_GET_MOTOR_ERROR,  // Errors
    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,  // CANOpen NMT Heartbeat  SEND
  };
  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;
  virtual ~ODriveCAN() = default;

  // GenericComponent interface
  void SetUp() override;
  void Terminate() override;

  // Get functions (msg.rtr bit must be set)
  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;
  // msg.rtr bit must NOT be set
  bool GetAdcVoltage(float& adc_voltage) override;

  // Set functions
  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;

  // Other functions
  bool Nmt() override;
  bool Estop() override;
  bool ClearErrors() override;
  bool StartAnticogging() override;

  static constexpr uint8_t NUM_NODE_ID_BITS = 6;
  static constexpr uint8_t NUM_CMD_ID_BITS = 11 - NUM_NODE_ID_BITS;

  // Utility functions
  static constexpr uint32_t GetNodeId(uint32_t msgID) {
    return (msgID » NUM_CMD_ID_BITS);  // Upper 6 or more bits
  }

  static constexpr uint8_t GetCmdId(uint32_t msgID) {
    return (msgID & 0x01F);  // Bottom 5 bits
  }

 private:
  huron::driver::can::BusBase* canbus_;
  uint32_t can_id_;
  uint32_t axis_id_;
  bool is_ext_ = false;
};

}  // namespace odrive
}  // namespace huron
```

9.56 odrive_enums.h

```
/*
 * Original source from: https://github.com/odriverobotics/ODrive/tree/master
 * */
#pragma once

/* TODO: This file is dangerous because the enums could potentially change between API versions. Should
    transmit as part of the JSON.
** To regenerate this file, nagivate to the top level of the ODrive repository and run:
** python Firmware/interface_generator_stub.py --definitions Firmware/odrive-interface.yaml --template
    tools/arduino_enums_template.j2 --output Arduino/ODriveArduino/ODriveEnums.h
*/

// ODrive.GpioMode
enum GpioMode {
  GPIO_MODE_DIGITAL                       = 0,
  GPIO_MODE_DIGITAL_PULL_UP               = 1,
  GPIO_MODE_DIGITAL_PULL_DOWN             = 2,
  GPIO_MODE_ANALOG_IN                     = 3,
  GPIO_MODE_UART_A                        = 4,
  GPIO_MODE_UART_B                        = 5,
  GPIO_MODE_UART_C                        = 6,
  GPIO_MODE_CAN_A                         = 7,
  GPIO_MODE_I2C_A                         = 8,
  GPIO_MODE_SPI_A                         = 9,
  GPIO_MODE_PWM                           = 10,
  GPIO_MODE_ENC0                          = 11,
  GPIO_MODE_ENC1                          = 12,
  GPIO_MODE_ENC2                          = 13,
  GPIO_MODE_MECH_BRAKE                    = 14,
  GPIO_MODE_STATUS                        = 15,
};

// ODrive.StreamProtocolType
enum StreamProtocolType {
  STREAM_PROTOCOL_TYPE_FIBRE              = 0,
  STREAM_PROTOCOL_TYPE_ASCII              = 1,
  STREAM_PROTOCOL_TYPE_STDOUT             = 2,
  STREAM_PROTOCOL_TYPE_ASCII_AND_STDOUT   = 3,
};

```

```
// ODrive.Can.Protocol
enum Protocol {
  PROTOCOL_SIMPLE                          = 0x00000001,
};

// ODrive.Axis.AxisState
enum AxisState {
  AXIS_STATE_UNDEFINED                     = 0,
  AXIS_STATE_IDLE                          = 1,
  AXIS_STATE_STARTUP_SEQUENCE              = 2,
  AXIS_STATE_FULL_CALIBRATION_SEQUENCE     = 3,
  AXIS_STATE_MOTOR_CALIBRATION             = 4,
  AXIS_STATE_ENCODER_INDEX_SEARCH          = 6,
  AXIS_STATE_ENCODER_OFFSET_CALIBRATION    = 7,
  AXIS_STATE_CLOSED_LOOP_CONTROL           = 8,
  AXIS_STATE_LOCKIN_SPIN                   = 9,
  AXIS_STATE_ENCODER_DIR_FIND              = 10,
  AXIS_STATE_HOMING                        = 11,
  AXIS_STATE_ENCODER_HALL_POLARITY_CALIBRATION = 12,
  AXIS_STATE_ENCODER_HALL_PHASE_CALIBRATION = 13,
};

// ODrive.Encoder.Mode
enum EncoderMode {
  ENCODER_MODE_INCREMENTAL                 = 0,
  ENCODER_MODE_HALL                        = 1,
  ENCODER_MODE_SINCOS                      = 2,
  ENCODER_MODE_SPI_ABS_CUI                 = 256,
  ENCODER_MODE_SPI_ABS_AMS                 = 257,
  ENCODER_MODE_SPI_ABS_AEAT                = 258,
  ENCODER_MODE_SPI_ABS_RLS                 = 259,
  ENCODER_MODE_SPI_ABS_MA732               = 260,
};

// ODrive.Controller.ControlMode
enum ControlMode {
  CONTROL_MODE_VOLTAGE_CONTROL             = 0,
  CONTROL_MODE_TORQUE_CONTROL              = 1,
  CONTROL_MODE_VELOCITY_CONTROL            = 2,
  CONTROL_MODE_POSITION_CONTROL            = 3,
};

// ODrive.Controller.InputMode
enum InputMode {
  INPUT_MODE_INACTIVE                      = 0,
  INPUT_MODE_PASSTHROUGH                   = 1,
  INPUT_MODE_VEL_RAMP                      = 2,
  INPUT_MODE_POS_FILTER                    = 3,
  INPUT_MODE_MIX_CHANNELS                  = 4,
  INPUT_MODE_TRAP_TRAJ                     = 5,
  INPUT_MODE_TORQUE_RAMP                   = 6,
  INPUT_MODE_MIRROR                        = 7,
  INPUT_MODE_TUNING                        = 8,
};

// ODrive.Motor.MotorType
enum MotorType {
  MOTOR_TYPE_HIGH_CURRENT                  = 0,
  MOTOR_TYPE_GIMBAL                        = 2,
  MOTOR_TYPE_ACIM                          = 3,
};

// ODrive.Error
enum ODriveError {
  ODRIVE_ERROR_NONE                        = 0x00000000,
  ODRIVE_ERROR_CONTROL_ITERATION_MISSED    = 0x00000001,
  ODRIVE_ERROR_DC_BUS_UNDER_VOLTAGE        = 0x00000002,
  ODRIVE_ERROR_DC_BUS_OVER_VOLTAGE         = 0x00000004,
  ODRIVE_ERROR_DC_BUS_OVER_REGEN_CURRENT   = 0x00000008,
  ODRIVE_ERROR_DC_BUS_OVER_CURRENT         = 0x00000010,
  ODRIVE_ERROR_BRAKE_DEADTIME_VIOLATION    = 0x00000020,
  ODRIVE_ERROR_BRAKE_DUTY_CYCLE_NAN        = 0x00000040,
  ODRIVE_ERROR_INVALID_BRAKE_RESISTANCE    = 0x00000080,
};

// ODrive.Can.Error
enum CanError {
  CAN_ERROR_NONE                           = 0x00000000,
  CAN_ERROR_DUPLICATE_CAN_IDS              = 0x00000001,
};

// ODrive.Axis.Error
enum AxisError {
  AXIS_ERROR_NONE                          = 0x00000000,
  AXIS_ERROR_INVALID_STATE                 = 0x00000001,
  AXIS_ERROR_MOTOR_FAILED                  = 0x00000040,
  AXIS_ERROR_SENSORLESS_ESTIMATOR_FAILED   = 0x00000080,
  AXIS_ERROR_ENCODER_FAILED                = 0x00000100,
  AXIS_ERROR_CONTROLLER_FAILED             = 0x00000200,
  AXIS_ERROR_WATCHDOG_TIMER_EXPIRED        = 0x00000800,
  AXIS_ERROR_MIN_ENDSTOP_PRESSED           = 0x00001000,
  AXIS_ERROR_MAX_ENDSTOP_PRESSED           = 0x00002000,
  AXIS_ERROR_ESTOP_REQUESTED               = 0x00004000,
  AXIS_ERROR_HOMING_WITHOUT_ENDSTOP        = 0x00020000,
  AXIS_ERROR_OVER_TEMP                     = 0x00040000,
  AXIS_ERROR_UNKNOWN_POSITION              = 0x00080000,
};

// ODrive.Motor.Error
enum MotorError {
  MOTOR_ERROR_NONE                         = 0x00000000,
  MOTOR_ERROR_PHASE_RESISTANCE_OUT_OF_RANGE = 0x00000001,
  MOTOR_ERROR_PHASE_INDUCTANCE_OUT_OF_RANGE = 0x00000002,
  MOTOR_ERROR_DRV_FAULT                    = 0x00000008,
  MOTOR_ERROR_CONTROL_DEADLINE_MISSED      = 0x00000010,
  MOTOR_ERROR_MODULATION_MAGNITUDE         = 0x00000080,
  MOTOR_ERROR_CURRENT_SENSE_SATURATION     = 0x00000400,
  MOTOR_ERROR_CURRENT_LIMIT_VIOLATION      = 0x00001000,
  MOTOR_ERROR_MODULATION_IS_NAN            = 0x00010000,
  MOTOR_ERROR_MOTOR_THERMISTOR_OVER_TEMP   = 0x00020000,
  MOTOR_ERROR_FET_THERMISTOR_OVER_TEMP     = 0x00040000,
  MOTOR_ERROR_TIMER_UPDATE_MISSED          = 0x00080000,
  MOTOR_ERROR_CURRENT_MEASUREMENT_UNAVAILABLE = 0x00100000,
  MOTOR_ERROR_CONTROLLER_FAILED            = 0x00200000,
  MOTOR_ERROR_I_BUS_OUT_OF_RANGE           = 0x00400000,
  MOTOR_ERROR_BRAKE_RESISTOR_DISARMED      = 0x00800000,
  MOTOR_ERROR_SYSTEM_LEVEL                 = 0x01000000,
  MOTOR_ERROR_BAD_TIMING                   = 0x02000000,
  MOTOR_ERROR_UNKNOWN_PHASE_ESTIMATE       = 0x04000000,
  MOTOR_ERROR_UNKNOWN_PHASE_VEL            = 0x08000000,
  MOTOR_ERROR_UNKNOWN_TORQUE               = 0x10000000,
  MOTOR_ERROR_UNKNOWN_CURRENT_COMMAND      = 0x20000000,
  MOTOR_ERROR_UNKNOWN_CURRENT_MEASUREMENT  = 0x40000000,
  MOTOR_ERROR_UNKNOWN_VBUS_VOLTAGE         = 0x80000000,
  MOTOR_ERROR_UNKNOWN_VOLTAGE_COMMAND      = 0x100000000,
  MOTOR_ERROR_UNKNOWN_GAINS                = 0x200000000,
  MOTOR_ERROR_CONTROLLER_INITIALIZING      = 0x400000000,
  MOTOR_ERROR_UNBALANCED_PHASES            = 0x800000000,
};

// ODrive.Controller.Error
enum ControllerError {
  CONTROLLER_ERROR_NONE                    = 0x00000000,
  CONTROLLER_ERROR_OVERSPEED               = 0x00000001,
  CONTROLLER_ERROR_INVALID_INPUT_MODE      = 0x00000002,
  CONTROLLER_ERROR_UNSTABLE_GAIN           = 0x00000004,
  CONTROLLER_ERROR_INVALID_MIRROR_AXIS     = 0x00000008,
  CONTROLLER_ERROR_INVALID_LOAD_ENCODER    = 0x00000010,
  CONTROLLER_ERROR_INVALID_ESTIMATE        = 0x00000020,
  CONTROLLER_ERROR_INVALID_CIRCULAR_RANGE  = 0x00000040,
  CONTROLLER_ERROR_SPINOUT_DETECTED        = 0x00000080,
};

// ODrive.Encoder.Error
enum EncoderError {
  ENCODER_ERROR_NONE                       = 0x00000000,
  ENCODER_ERROR_UNSTABLE_GAIN              = 0x00000001,
  ENCODER_ERROR_CPR_POLEPAIRS_MISMATCH     = 0x00000002,
  ENCODER_ERROR_NO_RESPONSE                = 0x00000004,
  ENCODER_ERROR_UNSUPPORTED_ENCODER_MODE   = 0x00000008,
  ENCODER_ERROR_ILLEGAL_HALL_STATE         = 0x00000010,
  ENCODER_ERROR_INDEX_NOT_FOUND_YET        = 0x00000020,
  ENCODER_ERROR_ABS_SPI_TIMEOUT            = 0x00000040,
  ENCODER_ERROR_ABS_SPI_COM_FAIL           = 0x00000080,
  ENCODER_ERROR_ABS_SPI_NOT_READY          = 0x00000100,
  ENCODER_ERROR_HALL_NOT_CALIBRATED_YET    = 0x00000200,
};

// ODrive.SensorlessEstimator.Error
enum SensorlessEstimatorError {
  SENSORLESS_ESTIMATOR_ERROR_NONE          = 0x00000000,
  SENSORLESS_ESTIMATOR_ERROR_UNSTABLE_GAIN = 0x00000001,
  SENSORLESS_ESTIMATOR_ERROR_UNKNOWN_CURRENT_MEASUREMENT = 0x00000002,
};

```

9.57 odrive_rotary_encoder.h

```
#pragma once

#include <memory>
#include "huron/control_interfaces/rotary_encoder.h"
```

```
#include "huron/odrive/odrive.h"

namespace huron {
namespace odrive {

class ODriveEncoder final : public huron::RotaryEncoder {
 public:
  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;
  ~ODriveEncoder() override = default;

  void Initialize() override;
  void SetUp() override;
  void Terminate() override;

 protected:
  void DoUpdateState() override;

 private:
  std::shared_ptr<ODrive> odrive_;
};

}  // namespace odrive
}  // namespace huron
```

9.58 odrive_torque_motor.h

```
#pragma once

#include <memory>
#include <vector>

#include "huron/control_interfaces/torque_motor.h"
#include "huron/odrive/odrive_can.h"

namespace huron {
namespace odrive {

class TorqueMotor : public huron::TorqueMotor {
 public:
  TorqueMotor(
    std::unique_ptr<TorqueMotorConfiguration> config,
    std::shared_ptr<ODrive> odrive,
    double gear_ratio);
  TorqueMotor(
    std::shared_ptr<ODrive> odrive,
    double gear_ratio);
  explicit TorqueMotor(std::shared_ptr<ODrive> odrive);
  TorqueMotor(const TorqueMotor&) = delete;
  TorqueMotor& operator=(const TorqueMotor&) = delete;
  ~TorqueMotor() = default;

  // GenericComponent methods
  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;

 private:
  std::shared_ptr<ODrive> odrive_;
};

}  // namespace odrive
}  // namespace huron
```

9.59 force_sensing_resistor.h

```
#pragma once

#include <eigen3/Eigen/Core>
#include <memory>
#include "huron/control_interfaces/sensor_with_frame.h"

```

```
namespace huron {

class ForceSensingResistor : public SensorWithFrame {
 public:
  explicit 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;
  ~ForceSensingResistor() override = default;
};

}  // namespace huron
```

9.60 force_sensing_resistor_array.h

```
#pragma once

#include <eigen3/Eigen/Core>

#include <string>
#include <memory>
#include <vector>

#include "huron/control_interfaces/sensor_with_frame.h"
#include "huron/sensors/force_sensing_resistor.h"

namespace huron {

class ForceSensingResistorArray : public SensorWithFrame {
 public:
  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;
  ~ForceSensingResistorArray() override = default;

  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 { return fsr_array_.size(); }

 protected:
  std::string name_;
  Eigen::VectorXd values_;
  std::vector<std::shared_ptr<ForceSensingResistor» fsr_array_;
};

}  // namespace huron
```

9.61 force_sensing_resistor_array_serial.h

```
#pragma once

#include <memory>
#include <vector>
#include <string>

#include "huron/sensors/force_sensing_resistor_array.h"
#include "huron/driver/serial/serial.h"

namespace huron {

class ForceSensingResistorArraySerial : public ForceSensingResistorArray {
 public:
  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;

  ~ForceSensingResistorArraySerial() override = default;

  void RequestStateUpdate() override;

  Eigen::VectorXd GetValue() const override;
  Eigen::VectorXd ReloadAndGetValue() override;

  void Initialize() override;
  void SetUp() override;
  void Terminate() override;

 private:
  static inline const std::string delimiter = ",";
  std::shared_ptr<driver::serial::SerialBase> serial_;
};

}  // namespace huron
```

9.62 string.h

```
#pragma once

#include <string>
#include <vector>

namespace huron {
namespace utils {

std::vector<std::string> split(const std::string& str,
                               const std::string& delimiter);
}  // namespace utils
}  // namespace huron
```

9.63 time.h

```
#pragma once

#include <chrono>  //NOLINT

template <
    class result_t   = std::chrono::milliseconds,
    class clock_t    = std::chrono::steady_clock,
    class duration_t = std::chrono::milliseconds
>
auto since(std::chrono::time_point<clock_t, duration_t> const& start) {
    return std::chrono::duration_cast<result_t>(clock_t::now() - start);
}

```

Index