Behaviour Tree PiCar-V

Generated by Doxygen 1.9.5

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 behaviour_tree Namespace Reference

Classes

- class BehaviourTreeHandler
- class CarContext

5.2 car Namespace Reference

Namespaces

- namespace configuration
- namespace plugin
- namespace system

5.3 car::configuration Namespace Reference

Classes

- struct Configuration

5.4 car::plugin Namespace Reference

Classes

- class Plugin
- class PluginManager

5.5 car::system Namespace Reference

Namespaces

- namespace device
- namespace logging
- namespace messaging
- namespace movement

Classes

- class CarSystem

5.6 car::system::device Namespace Reference

Namespaces

- namespace lidar

Classes

- class CameraDevice
- class DeviceManager

5.7 car::system::device::lidar Namespace Reference

Classes

- class LidarDevice
- class LidarDummy
- class LidarScanner

5.8 car::system::logging Namespace Reference

Classes

- class VectorSink

Typedefs

- using vector_sink_mt = VectorSink< std::mutex >

5.8.1 Typedef Documentation

5.8.1.1 vector_sink_mt

```
using car::system::logging::vector_sink_mt = typedef VectorSink<std::mutex>
```

5.9 car::system::messaging Namespace Reference

Classes

- class MessagingSystem

5.10 car::system::movement Namespace Reference

Namespaces

- namespace controller

Classes

- class MovementSystem

5.11 car::system::movement::controller Namespace Reference

Classes

- class AbstractMovementController
- class DummyMovementController

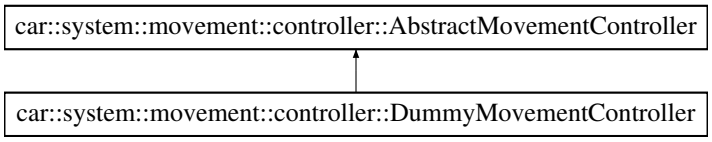
Chapter 6

Class Documentation

6.1 car::system::movement::controller::AbstractMovementController Class Reference

`#include <AbstractMovementController.h>`

Inheritance diagram for car::system::movement::controller::AbstractMovementController:



Public Member Functions

- virtual void initialize ()=0
- virtual void stop ()=0
- virtual void terminate ()=0
- virtual void setRearWheelsSpeed (const int speed)=0
- virtual void setRearLeftWheelSpeed (const int speed)=0
- virtual void setRearRightWheelSpeed (const int speed)=0
- virtual void setFrontWheelsAngle (const float angle)=0
- virtual void setCameraServo1Angle (const float angle)=0
- virtual void setCameraServo2Angle (const float angle)=0
- virtual void setRearWheelsDirectionToForward ()=0
- virtual void setRearLeftWheelDirectionToForward ()=0
- virtual void setRearRightWheelDirectionToForward ()=0
- virtual void setRearWheelsDirectionToBackward ()=0
- virtual void setRearLeftWheelDirectionToBackward ()=0
- virtual void setRearRightWheelDirectionToBackward ()=0

6.1.1 Member Function Documentation

6.1.1.1 initialize()

```
virtual void car::system::movement::controller::AbstractMovementController::initialize ( )
[pure virtual]
```

Implemented in car::system::movement::controller::DummyMovementController.

6.1.1.2 setCameraServo1Angle()

```
virtual void car::system::movement::controller::AbstractMovementController::setCameraServo1↩
Angle (
            const float angle )  [pure virtual]
```

Implemented in car::system::movement::controller::DummyMovementController.

6.1.1.3 setCameraServo2Angle()

```
virtual void car::system::movement::controller::AbstractMovementController::setCameraServo2↩
Angle (
            const float angle )  [pure virtual]
```

Implemented in car::system::movement::controller::DummyMovementController.

6.1.1.4 setFrontWheelsAngle()

```
virtual void car::system::movement::controller::AbstractMovementController::setFrontWheels↩
Angle (
            const float angle )  [pure virtual]
```

Implemented in car::system::movement::controller::DummyMovementController.

6.1.1.5 setRearLeftWheelDirectionToBackward()

```
virtual void car::system::movement::controller::AbstractMovementController::setRearLeftWheel↩
DirectionToBackward ( )  [pure virtual]
```

Implemented in car::system::movement::controller::DummyMovementController.

6.1.1.6 setRearLeftWheelDirectionToForward()

```
virtual void car::system::movement::controller::AbstractMovementController::setRearLeftWheel↩
DirectionToForward ( )  [pure virtual]
```

Implemented in car::system::movement::controller::DummyMovementController.

6.1.1.7 setRearLeftWheelSpeed()

```
virtual void car::system::movement::controller::AbstractMovementController::setRearLeftWheel↩
Speed (
            const int speed )  [pure virtual]
```

Implemented in car::system::movement::controller::DummyMovementController.

6.1.1.8 setRearRightWheelDirectionToBackward()

```
virtual void car::system::movement::controller::AbstractMovementController::setRearRight↩
WheelDirectionToBackward ( )  [pure virtual]
```

Implemented in car::system::movement::controller::DummyMovementController.

6.1.1.9 setRearRightWheelDirectionToForward()

```
virtual void car::system::movement::controller::AbstractMovementController::setRearRight↩
WheelDirectionToForward ( )  [pure virtual]
```

Implemented in car::system::movement::controller::DummyMovementController.

6.1.1.10 setRearRightWheelSpeed()

```
virtual void car::system::movement::controller::AbstractMovementController::setRearRight↩
WheelSpeed (
            const int speed )  [pure virtual]
```

Implemented in car::system::movement::controller::DummyMovementController.

6.1.1.11 setRearWheelsDirectionToBackward()

```
virtual void car::system::movement::controller::AbstractMovementController::setRearWheels↩
DirectionToBackward ( )  [pure virtual]
```

Implemented in car::system::movement::controller::DummyMovementController.

6.1.1.12 setRearWheelsDirectionToForward()

```
virtual void car::system::movement::controller::AbstractMovementController::setRearWheels↩
DirectionToForward ( )  [pure virtual]
```

Implemented in car::system::movement::controller::DummyMovementController.

6.1.1.13 setRearWheelsSpeed()

```
virtual void car::system::movement::controller::AbstractMovementController::setRearWheelsSpeed
(
            const int speed )  [pure virtual]
```

Implemented in car::system::movement::controller::DummyMovementController.

6.1.1.14 stop()

```
virtual void car::system::movement::controller::AbstractMovementController::stop ( )  [pure
virtual]
```

Implemented in car::system::movement::controller::DummyMovementController.

6.1.1.15 terminate()

```
virtual void car::system::movement::controller::AbstractMovementController::terminate ( )
[pure virtual]
```

Implemented in car::system::movement::controller::DummyMovementController.

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

- include/car/system/movement/controller/AbstractMovementController.h

6.2 BackWheels Class Reference

Public Member Functions

- BackWheels (const int &bus_number=1)
- void forward ()
- void backward ()
- void stop ()
- int getSpeed () const
- void setSpeed (const int &speed)
- void ready ()
- void calibration ()
- void caliLeft ()
- void caliRight ()
- void caliOK ()

Public Attributes

- PCA9685 pca9685

Private Attributes

- std::unique_ptr< TB6612 > left_wheel
- std::unique_ptr< TB6612 > right_wheel
- int forward_A
- int forward_B
- int cali_forward_A
- int cali_forward_B
- int speed

6.2.1 Constructor & Destructor Documentation

6.2.1.1 BackWheels()

```
BackWheels::BackWheels (
            const int & bus_number = 1 )  [inline]
```

6.2.2 Member Function Documentation

6.2.2.1 backward()

```
void BackWheels::backward ( )  [inline]
```

6.2.2.2 calibration()

```
void BackWheels::calibration ( )  [inline]
```

6.2.2.3 caliLeft()

```
void BackWheels::caliLeft ( )  [inline]
```

6.2.2.4 caliOK()

```
void BackWheels::caliOK ( )  [inline]
```

6.2.2.5 caliRight()

```
void BackWheels::caliRight ( )  [inline]
```

6.2.2.6 forward()

```
void BackWheels::forward ( )  [inline]
```

6.2.2.7 getSpeed()

```
int BackWheels::getSpeed ( ) const  [inline]
```

6.2.2.8 ready()

```
void BackWheels::ready ( )  [inline]
```

6.2.2.9 setSpeed()

```
void BackWheels::setSpeed (
            const int & speed )  [inline]
```

6.2.2.10 stop()

```
void BackWheels::stop ( )  [inline]
```

6.2.3 Member Data Documentation

6.2.3.1 cali_forward_A

```
int BackWheels::cali_forward_A  [private]
```

6.2.3.2 cali_forward_B

```
int BackWheels::cali_forward_B  [private]
```

6.2.3.3 forward_A

```
int BackWheels::forward_A  [private]
```

6.2.3.4 forward_B

```
int BackWheels::forward_B  [private]
```

6.2.3.5 left_wheel

```
std::unique_ptr<TB6612> BackWheels::left_wheel  [private]
```

6.2.3.6 pca9685

```
PCA9685 BackWheels::pca9685
```

6.2.3.7 right_wheel

```
std::unique_ptr<TB6612> BackWheels::right_wheel  [private]
```

6.2.3.8 speed

```
int BackWheels::speed  [private]
```

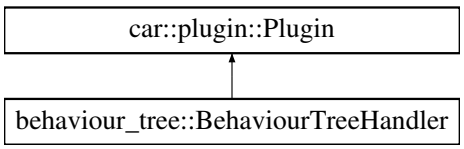
The documentation for this class was generated from the following file:

- tests/tb6612/test_rear_wheels.cpp

6.3 behaviour_tree::BehaviourTreeHandler Class Reference

```
#include <BehaviourTreeHandler.hpp>
```

Inheritance diagram for behaviour_tree::BehaviourTreeHandler:



Public Member Functions

- void initialize (std::shared_ptr< car::system::CarSystem > car_system) final override
- void handleCommand (const std::string message, const rapidjson::Document &message_json)
- void setBehaviourTree (const rapidjson::Document &message_json)
- void startBehaviourTree ()
- void stopBehaviourTree ()
- void update () final override
- void stop () final override
- std::string getName () final override
- void _setBehaviourTree (std::shared_ptr< BehaviourTree > behaviour_tree)

Private Attributes

- std::shared_ptr< car::system::CarSystem > car_system
- std::shared_ptr< BehaviourTree > behaviour_tree
- std::shared_ptr< Context > context
- int tick_count = 0
- std::chrono::time_point< std::chrono::steady_clock > last_connected

6.3.1 Member Function Documentation

6.3.1.1 _setBehaviourTree()

```
void behaviour_tree::BehaviourTreeHandler::_setBehaviourTree (
            std::shared_ptr< BehaviourTree > behaviour_tree )  [inline]
```

6.3.1.2 getName()

```
std::string behaviour_tree::BehaviourTreeHandler::getName ( )  [inline], [final], [override],
[virtual]
```

Implements car::plugin::Plugin.

6.3.1.3 handleCommand()

```
void behaviour_tree::BehaviourTreeHandler::handleCommand (
            const std::string message,
            const rapidjson::Document & message_json )  [inline]
```

6.3.1.4 initialize()

```
void behaviour_tree::BehaviourTreeHandler::initialize (
            std::shared_ptr< car::system::CarSystem > car_system )  [inline], [final], [override],
[virtual]
```

Implements car::plugin::Plugin.

6.3.1.5 setBehaviourTree()

```
void behaviour_tree::BehaviourTreeHandler::setBehaviourTree (
            const rapidjson::Document & message_json )  [inline]
```

6.3.1.6 startBehaviourTree()

```
void behaviour_tree::BehaviourTreeHandler::startBehaviourTree ( )  [inline]
```

6.3.1.7 stop()

```
void behaviour_tree::BehaviourTreeHandler::stop ( )  [inline], [final], [override], [virtual]
```

Implements car::plugin::Plugin.

6.3.1.8 stopBehaviourTree()

```
void behaviour_tree::BehaviourTreeHandler::stopBehaviourTree ( )  [inline]
```

6.3.1.9 update()

```
void behaviour_tree::BehaviourTreeHandler::update ( )  [inline], [final], [override], [virtual]
```

Implements car::plugin::Plugin.

6.3.2 Member Data Documentation

6.3.2.1 behaviour_tree

```
std::shared_ptr<BehaviourTree> behaviour_tree::BehaviourTreeHandler::behaviour_tree  [private]
```

6.3.2.2 car_system

```
std::shared_ptr<car::system::CarSystem> behaviour_tree::BehaviourTreeHandler::car_system
[private]
```

6.3.2.3 context

```
std::shared_ptr<Context> behaviour_tree::BehaviourTreeHandler::context  [private]
```

6.3.2.4 last_connected

```
std::chrono::time_point<std::chrono::steady_clock> behaviour_tree::BehaviourTreeHandler↩
::last_connected  [private]
```

6.3.2.5 tick_count

```
int behaviour_tree::BehaviourTreeHandler::tick_count = 0  [private]
```

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

- include/behaviour_tree/BehaviourTreeHandler.hpp

6.4 car::system::device::CameraDevice Class Reference

```
#include <CameraDevice.h>
```

Public Member Functions

- CameraDevice (std::shared_ptr< configuration::Configuration > configuration)
- CameraDevice (const CameraDevice &)=delete
- CameraDevice & operator= (const CameraDevice &)=delete
- CameraDevice (CameraDevice &&)=delete
- CameraDevice & operator= (CameraDevice &&)=delete
- ~CameraDevice ()=default
- std::string getFrameBuffer () const

Static Public Member Functions

- static tl::expected< std::unique_ptr< CameraDevice >, std::string > create (std::shared_ptr< configuration::Configuration > configuration)

Protected Member Functions

- void start ()
- void update ()
- void stop ()
- void disconnect ()
- void terminate ()

Private Attributes

- std::shared_ptr< configuration::Configuration > configuration
- std::unique_ptr< cv::VideoCapture > camera_
- bool connected_ = false
- std::string frame_buffer_
- std::mutex camera_mutex_
- std::chrono::steady_clock::time_point last

Friends

- class DeviceManager

6.4.1 Constructor & Destructor Documentation

6.4.1.1 CameraDevice() [1/3]

```
car::system::device::CameraDevice::CameraDevice (
            std::shared_ptr< configuration::Configuration > configuration )  [inline]
```

6.4.1.2 CameraDevice() [2/3]

```
car::system::device::CameraDevice::CameraDevice (
            const CameraDevice &  )  [delete]
```

6.4.1.3 CameraDevice() [3/3]

```
car::system::device::CameraDevice::CameraDevice (
            CameraDevice &&  )  [delete]
```

6.4.1.4 ~CameraDevice()

```
car::system::device::CameraDevice::~CameraDevice ( )  [default]
```

6.4.2 Member Function Documentation

6.4.2.1 create()

```
tl::expected< std::unique_ptr< CameraDevice >, std::string > car::system::device::Camera↩
Device::create (
            std::shared_ptr< configuration::Configuration > configuration )  [static]
```

6.4.2.2 disconnect()

```
void car::system::device::CameraDevice::disconnect ( )  [protected]
```

6.4.2.3 getFrameBuffer()

```
std::string car::system::device::CameraDevice::getFrameBuffer ( ) const
```

6.4.2.4 operator=() [1/2]

```
CameraDevice & car::system::device::CameraDevice::operator= (
            CameraDevice &&  )  [delete]
```

6.4.2.5 operator=() [2/2]

```
CameraDevice & car::system::device::CameraDevice::operator= (
            const CameraDevice &  )  [delete]
```

6.4.2.6 start()

```
void car::system::device::CameraDevice::start ( )  [protected]
```

6.4.2.7 stop()

```
void car::system::device::CameraDevice::stop ( )  [protected]
```

6.4.2.8 terminate()

```
void car::system::device::CameraDevice::terminate ( )  [protected]
```

6.4.2.9 update()

```
void car::system::device::CameraDevice::update ( )  [protected]
```

6.4.3 Friends And Related Function Documentation

6.4.3.1 DeviceManager

```
friend class DeviceManager  [friend]
```

6.4.4 Member Data Documentation

6.4.4.1 camera_

```
std::unique_ptr<cv::VideoCapture> car::system::device::CameraDevice::camera_  [private]
```

6.4.4.2 camera_mutex_

```
std::mutex car::system::device::CameraDevice::camera_mutex_  [private]
```

6.4.4.3 configuration

```
std::shared_ptr<configuration::Configuration> car::system::device::CameraDevice::configuration
[private]
```

6.4.4.4 connected_

```
bool car::system::device::CameraDevice::connected_ = false  [private]
```

6.4.4.5 frame_buffer_

```
std::string car::system::device::CameraDevice::frame_buffer_  [private]
```

6.4.4.6 last

```
std::chrono::steady_clock::time_point car::system::device::CameraDevice::last  [private]
```

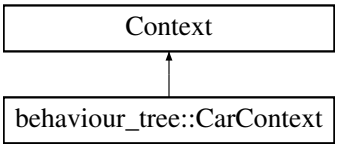
The documentation for this class was generated from the following files:

- include/car/system/device/CameraDevice.h
- src/car/system/device/CameraDevice.cpp

6.5 behaviour_tree::CarContext Class Reference

```
#include <CarContext.hpp>
```

Inheritance diagram for behaviour_tree::CarContext:



Public Member Functions

- CarContext (std::shared_ptr< BehaviourTree > behaviour_tree, std::shared_ptr< car::system::CarSystem > car_system)
- std::shared_ptr< car::system::CarSystem > getCarSystem () const
- void _ () override

Private Attributes

- std::shared_ptr< car::system::CarSystem > car_system

6.5.1 Constructor & Destructor Documentation

6.5.1.1 CarContext()

```
behaviour_tree::CarContext::CarContext (
            std::shared_ptr< BehaviourTree > behaviour_tree,
            std::shared_ptr< car::system::CarSystem > car_system )  [inline]
```

6.5.2 Member Function Documentation

6.5.2.1 _()

```
void behaviour_tree::CarContext::_ ( )  [inline], [override]
```

6.5.2.2 getCarSystem()

```
std::shared_ptr< car::system::CarSystem > behaviour_tree::CarContext::getCarSystem ( ) const
[inline]
```

6.5.3 Member Data Documentation

6.5.3.1 car_system

```
std::shared_ptr<car::system::CarSystem> behaviour_tree::CarContext::car_system  [private]
```

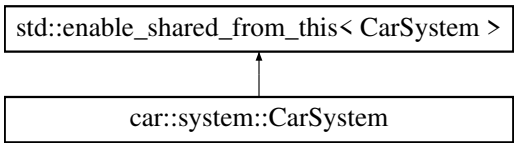
The documentation for this class was generated from the following file:

- include/behaviour_tree/CarContext.hpp

6.6 car::system::CarSystem Class Reference

```
#include <CarSystem.h>
```

Inheritance diagram for car::system::CarSystem:



Public Member Functions

- CarSystem (std::shared_ptr< Configuration > configuration, std::unique_ptr< DeviceManager > device_↩ manager, std::unique_ptr< MessagingSystem > messaging_system, std::unique_ptr< MovementSystem > movement_system, std::unique_ptr< PluginManager > plugin_manager)
- void initialize ()
- void reload ()
- void start ()
- void stop ()
- tl::expected< nullptr_t, std::string > tryConnect ()
- void disconnect ()
- void terminate ()

  Only devices should be terminated here since deconstructor does not work when the program is terminated by the user.
- void update ()
- const std::shared_ptr< Configuration > getConfiguration () const
- void setConfiguration (std::shared_ptr< Configuration > configuration)
- DeviceManager * getDeviceManager () const
- MessagingSystem * getMessagingSystem () const
- MovementSystem * getMovementSystem () const
- template<typename T >
  const std::shared_ptr< T > getPlugin () const

Private Member Functions

- void sendData ()

Private Attributes

- std::shared_ptr< Configuration > configuration_
- const std::unique_ptr< DeviceManager > device_manager_
- const std::unique_ptr< MessagingSystem > messaging_system_
- const std::unique_ptr< MovementSystem > movement_system_
- const std::unique_ptr< PluginManager > plugin_manager_
- bool initialized = false
- bool started = false

6.6.1 Constructor & Destructor Documentation

6.6.1.1 CarSystem()

```
car::system::CarSystem::CarSystem (
            std::shared_ptr< Configuration > configuration,
            std::unique_ptr< DeviceManager > device_manager,
            std::unique_ptr< MessagingSystem > messaging_system,
            std::unique_ptr< MovementSystem > movement_system,
            std::unique_ptr< PluginManager > plugin_manager )
```

6.6.2 Member Function Documentation

6.6.2.1 disconnect()

```
void car::system::CarSystem::disconnect ( )
```

6.6.2.2 getConfiguration()

```
const std::shared_ptr< Configuration > car::system::CarSystem::getConfiguration ( ) const
[inline]
```

6.6.2.3 getDeviceManager()

```
DeviceManager * car::system::CarSystem::getDeviceManager ( ) const  [inline]
```

6.6.2.4 getMessagingSystem()

```
MessagingSystem * car::system::CarSystem::getMessagingSystem ( ) const  [inline]
```

6.6.2.5 getMovementSystem()

```
MovementSystem * car::system::CarSystem::getMovementSystem ( ) const  [inline]
```

6.6.2.6 getPlugin()

```
template<typename T >
const std::shared_ptr< T > car::system::CarSystem::getPlugin ( ) const  [inline]
```

6.6.2.7 initialize()

```
void car::system::CarSystem::initialize ( )
```

6.6.2.8 reload()

```
void car::system::CarSystem::reload ( )
```

6.6.2.9 sendData()

```
void car::system::CarSystem::sendData ( )  [private]
```

6.6.2.10 setConfiguration()

```
void car::system::CarSystem::setConfiguration (
            std::shared_ptr< Configuration > configuration )
```

6.6.2.11 start()

```
void car::system::CarSystem::start ( )
```

6.6.2.12 stop()

```
void car::system::CarSystem::stop ( )
```

6.6.2.13 terminate()

```
void car::system::CarSystem::terminate ( )
```

Only devices should be terminated here since deconstructor does not work when the program is terminated by the user.

6.6.2.14 tryConnect()

```
tl::expected< nullptr_t, std::string > car::system::CarSystem::tryConnect ( )
```

6.6.2.15 update()

```
void car::system::CarSystem::update ( )
```

6.6.3 Member Data Documentation

6.6.3.1 configuration_

```
std::shared_ptr<Configuration> car::system::CarSystem::configuration_  [private]
```

6.6.3.2 device_manager_

```
const std::unique_ptr<DeviceManager> car::system::CarSystem::device_manager_  [private]
```

6.6.3.3 initialized

```
bool car::system::CarSystem::initialized = false  [private]
```

6.6.3.4 messaging_system_

```
const std::unique_ptr<MessagingSystem> car::system::CarSystem::messaging_system_  [private]
```

6.6.3.5 movement_system_

```
const std::unique_ptr<MovementSystem> car::system::CarSystem::movement_system_  [private]
```

6.6.3.6 plugin_manager_

```
const std::unique_ptr<PluginManager> car::system::CarSystem::plugin_manager_  [private]
```

6.6.3.7 started

```
bool car::system::CarSystem::started = false  [private]
```

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

- include/car/system/CarSystem.h
- src/car/system/CarSystem.cpp

6.7 car::configuration::Configuration Struct Reference

```
#include <Configuration.h>
```

Public Member Functions

- void setCameraFps (const int camera_fps)
- const int getCameraFpsInterval ()

Public Attributes

- std::string host = "127.0.0.1:3000"
- int camera_index = 0
- bool use_camera = true
- std::string lidar_port = ""
- bool use_lidar = true
- std::chrono::milliseconds behaviour_tree_update_ms_interval = std::chrono::milliseconds(100)

Private Attributes

- int camera_fps = 60
- int camera_fps_interval = 1000

6.7.1 Member Function Documentation

6.7.1.1 getCameraFpsInterval()

```
const int car::configuration::Configuration::getCameraFpsInterval ( )  [inline]
```

6.7.1.2 setCameraFps()

```
void car::configuration::Configuration::setCameraFps (
            const int camera_fps )  [inline]
```

6.7.2 Member Data Documentation

6.7.2.1 behaviour_tree_update_ms_interval

```
std::chrono::milliseconds car::configuration::Configuration::behaviour_tree_update_ms_interval
= std::chrono::milliseconds(100)
```

6.7.2.2 camera_fps

```
int car::configuration::Configuration::camera_fps = 60  [private]
```

6.7.2.3 camera_fps_interval

```
int car::configuration::Configuration::camera_fps_interval = 1000  [private]
```

6.7.2.4 camera_index

```
int car::configuration::Configuration::camera_index = 0
```

6.7.2.5 host

```
std::string car::configuration::Configuration::host = "127.0.0.1:3000"
```

6.7.2.6 lidar_port

```
std::string car::configuration::Configuration::lidar_port = ""
```

6.7.2.7 use_camera

```
bool car::configuration::Configuration::use_camera = true
```

6.7.2.8 use_lidar

```
bool car::configuration::Configuration::use_lidar = true
```

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

- include/car/configuration/Configuration.h

6.8 car::system::device::DeviceManager Class Reference

```
#include <DeviceManager.h>
```

Public Member Functions

- DeviceManager (std::unique_ptr< CameraDevice > camera_device, std::unique_ptr< lidar::LidarDevice > lidar_device)
- CameraDevice * getCameraDevice ()
- lidar::LidarDevice * getLidarDevice ()
- const bool isRunning () const
- void initialize (std::shared_ptr< system::CarSystem > car_system)
- void start ()
- void update ()
- void stop ()
- void terminate ()

Static Public Member Functions

- static tl::expected< std::unique_ptr< DeviceManager >, std::string > create (std::shared_ptr< Configuration > configuration)

Private Attributes

- std::shared_ptr< car::system::CarSystem > car_system
- bool is_initialized_ = false
- bool is_running_ = false
- std::unique_ptr< lidar::LidarDevice > lidar_device_
- std::unique_ptr< CameraDevice > camera_device_

6.8.1 Constructor & Destructor Documentation

6.8.1.1 DeviceManager()

```
car::system::device::DeviceManager::DeviceManager (
            std::unique_ptr< CameraDevice > camera_device,
            std::unique_ptr< lidar::LidarDevice > lidar_device ) [inline]
```

6.8.2 Member Function Documentation

6.8.2.1 create()

```
tl::expected< std::unique_ptr< DeviceManager >, std::string > car::system::device::Device↩
Manager::create (
            std::shared_ptr< Configuration > configuration )  [static]
```

6.8.2.2 getCameraDevice()

```
CameraDevice * car::system::device::DeviceManager::getCameraDevice ( )  [inline]
```

6.8.2.3 getLidarDevice()

```
lidar::LidarDevice * car::system::device::DeviceManager::getLidarDevice ( )  [inline]
```

6.8.2.4 initialize()

```
void car::system::device::DeviceManager::initialize (
            std::shared_ptr< system::CarSystem > car_system )
```

6.8.2.5 isRunning()

```
const bool car::system::device::DeviceManager::isRunning ( ) const  [inline]
```

6.8.2.6 start()

```
void car::system::device::DeviceManager::start ( )
```

6.8.2.7 stop()

```
void car::system::device::DeviceManager::stop ( )
```

6.8.2.8 terminate()

```
void car::system::device::DeviceManager::terminate ( )
```

6.8.2.9 update()

```
void car::system::device::DeviceManager::update ( )
```

6.8.3 Member Data Documentation

6.8.3.1 camera_device_

```
std::unique_ptr<CameraDevice> car::system::device::DeviceManager::camera_device_  [private]
```

6.8.3.2 car_system

```
std::shared_ptr<car::system::CarSystem> car::system::device::DeviceManager::car_system  [private]
```

6.8.3.3 is_initialized_

```
bool car::system::device::DeviceManager::is_initialized_ = false  [private]
```

6.8.3.4 is_running_

```
bool car::system::device::DeviceManager::is_running_ = false  [private]
```

6.8.3.5 lidar_device_

```
std::unique_ptr<lidar::LidarDevice> car::system::device::DeviceManager::lidar_device_  [private]
```

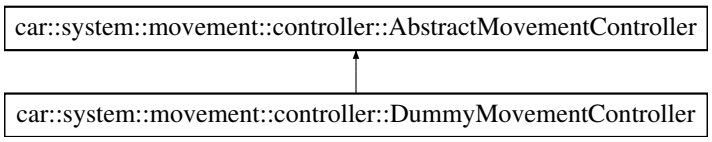
The documentation for this class was generated from the following files:

- include/car/system/device/DeviceManager.h
- src/car/system/device/DeviceManager.cpp

6.9 car::system::movement::controller::DummyMovementController Class Reference

```
#include <DummyMovementController.h>
```

Inheritance diagram for car::system::movement::controller::DummyMovementController:



Public Member Functions

- void initialize () final override
- void stop () final override
- void terminate () final override
- void setRearWheelsSpeed (const int speed) final override
- void setRearLeftWheelSpeed (const int speed) final override
- void setRearRightWheelSpeed (const int speed) final override
- void setFrontWheelsAngle (const float angle) final override
- void setCameraServo1Angle (const float angle) final override
- void setCameraServo2Angle (const float angle) final override
- void setRearWheelsDirectionToForward () final override
- void setRearLeftWheelDirectionToForward () final override
- void setRearRightWheelDirectionToForward () final override
- void setRearWheelsDirectionToBackward () final override
- void setRearLeftWheelDirectionToBackward () final override
- void setRearRightWheelDirectionToBackward () final override

6.9.1 Member Function Documentation

6.9.1.1 initialize()

```
void car::system::movement::controller::DummyMovementController::initialize ( )  [inline],
[final], [override], [virtual]
```

Implements car::system::movement::controller::AbstractMovementController.

6.9.1.2 setCameraServo1Angle()

```
void car::system::movement::controller::DummyMovementController::setCameraServo1Angle (
            const float angle )  [final], [override], [virtual]
```

Implements car::system::movement::controller::AbstractMovementController.

6.9.1.3 setCameraServo2Angle()

```
void car::system::movement::controller::DummyMovementController::setCameraServo2Angle (
            const float angle )  [final], [override], [virtual]
```

Implements car::system::movement::controller::AbstractMovementController.

6.9.1.4 setFrontWheelsAngle()

```
void car::system::movement::controller::DummyMovementController::setFrontWheelsAngle (
            const float angle )  [final], [override], [virtual]
```

Implements car::system::movement::controller::AbstractMovementController.

6.9.1.5 setRearLeftWheelDirectionToBackward()

```
void car::system::movement::controller::DummyMovementController::setRearLeftWheelDirectionTo↩
Backward ( )  [final], [override], [virtual]
```

Implements car::system::movement::controller::AbstractMovementController.

6.9.1.6 setRearLeftWheelDirectionToForward()

```
void car::system::movement::controller::DummyMovementController::setRearLeftWheelDirectionTo↩
Forward ( )  [final], [override], [virtual]
```

Implements car::system::movement::controller::AbstractMovementController.

6.9.1.7 setRearLeftWheelSpeed()

```
void car::system::movement::controller::DummyMovementController::setRearLeftWheelSpeed (
            const int speed )  [final], [override], [virtual]
```

Implements car::system::movement::controller::AbstractMovementController.

6.9.1.8 setRearRightWheelDirectionToBackward()

```
void car::system::movement::controller::DummyMovementController::setRearRightWheelDirection↩
ToBackward ( )  [final], [override], [virtual]
```

Implements car::system::movement::controller::AbstractMovementController.

6.9.1.9 setRearRightWheelDirectionToForward()

```
void car::system::movement::controller::DummyMovementController::setRearRightWheelDirection↩
ToForward ( )  [final], [override], [virtual]
```

Implements car::system::movement::controller::AbstractMovementController.

6.9.1.10 setRearRightWheelSpeed()

```
void car::system::movement::controller::DummyMovementController::setRearRightWheelSpeed (
            const int speed )  [final], [override], [virtual]
```

Implements car::system::movement::controller::AbstractMovementController.

6.9.1.11 setRearWheelsDirectionToBackward()

```
void car::system::movement::controller::DummyMovementController::setRearWheelsDirectionTo↩
Backward ( )  [final], [override], [virtual]
```

Implements car::system::movement::controller::AbstractMovementController.

6.9.1.12 setRearWheelsDirectionToForward()

```
void car::system::movement::controller::DummyMovementController::setRearWheelsDirectionTo↩
Forward ( )  [final], [override], [virtual]
```

Implements car::system::movement::controller::AbstractMovementController.

6.9.1.13 setRearWheelsSpeed()

```
void car::system::movement::controller::DummyMovementController::setRearWheelsSpeed (
            const int speed ) [final], [override], [virtual]
```

Implements car::system::movement::controller::AbstractMovementController.

6.9.1.14 stop()

```
void car::system::movement::controller::DummyMovementController::stop ( ) [final], [override],
[virtual]
```

Implements car::system::movement::controller::AbstractMovementController.

6.9.1.15 terminate()

```
void car::system::movement::controller::DummyMovementController::terminate ( ) [inline],
[final], [override], [virtual]
```

Implements car::system::movement::controller::AbstractMovementController.

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

- include/car/system/movement/controller/DummyMovementController.h
- src/car/system/movement/controller/DummyMovementController.cpp

6.10 car::system::messaging::MessagingSystem::FirstMessageStruct Struct Reference

```
#include <MessagingSystem.h>
```

Public Attributes

- std::string error_message
- std::string uuid
- std::condition_variable condition

6.10.1 Member Data Documentation

6.10.1.1 condition

```
std::condition_variable car::system::messaging::MessagingSystem::FirstMessageStruct::condition
```

6.10.1.2 error_message

```
std::string car::system::messaging::MessagingSystem::FirstMessageStruct::error_message
```

6.10.1.3 uuid

```
std::string car::system::messaging::MessagingSystem::FirstMessageStruct::uuid
```

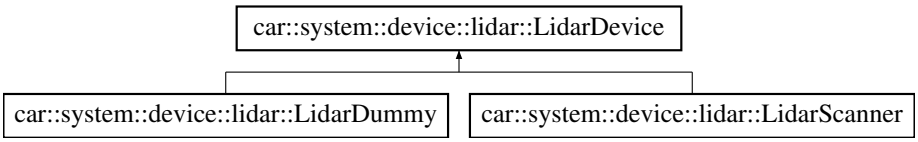
The documentation for this struct was generated from the following file:

- include/car/system/messaging/MessagingSystem.h

6.11 car::system::device::lidar::LidarDevice Class Reference

```
#include <LidarDevice.h>
```

Inheritance diagram for car::system::device::lidar::LidarDevice:



Public Member Functions

- std::vector< Measure > getScanData () const
- virtual void start ()=0
- virtual void update ()=0
- virtual void stop ()=0
- virtual void initialize ()=0
- virtual void terminate ()=0
- virtual void disconnect ()=0

Protected Member Functions

- void setScanData (const std::vector< Measure > &scan_data)

Protected Attributes

- std::vector< Measure > scan_data_

Friends

- class DeviceManager

6.11.1 Member Function Documentation

6.11.1.1 disconnect()

```
virtual void car::system::device::lidar::LidarDevice::disconnect ( )  [pure virtual]
```

Implemented in car::system::device::lidar::LidarDummy, and car::system::device::lidar::LidarScanner.

6.11.1.2 getScanData()

```
std::vector< Measure > car::system::device::lidar::LidarDevice::getScanData ( ) const  [inline]
```

6.11.1.3 initialize()

```
virtual void car::system::device::lidar::LidarDevice::initialize ( )  [pure virtual]
```

Implemented in car::system::device::lidar::LidarDummy, and car::system::device::lidar::LidarScanner.

6.11.1.4 setScanData()

```
void car::system::device::lidar::LidarDevice::setScanData (
            const std::vector< Measure > & scan_data )  [inline], [protected]
```

6.11.1.5 start()

```
virtual void car::system::device::lidar::LidarDevice::start ( )  [pure virtual]
```

Implemented in car::system::device::lidar::LidarDummy, and car::system::device::lidar::LidarScanner.

6.11.1.6 stop()

```
virtual void car::system::device::lidar::LidarDevice::stop ( )  [pure virtual]
```

Implemented in car::system::device::lidar::LidarDummy, and car::system::device::lidar::LidarScanner.

6.11.1.7 terminate()

```
virtual void car::system::device::lidar::LidarDevice::terminate ( )  [pure virtual]
```

Implemented in car::system::device::lidar::LidarDummy, and car::system::device::lidar::LidarScanner.

6.11.1.8 update()

```
virtual void car::system::device::lidar::LidarDevice::update ( )  [pure virtual]
```

Implemented in car::system::device::lidar::LidarDummy, and car::system::device::lidar::LidarScanner.

6.11.2 Friends And Related Function Documentation

6.11.2.1 DeviceManager

```
friend class DeviceManager  [friend]
```

6.11.3 Member Data Documentation

6.11.3.1 scan_data_

```
std::vector<Measure> car::system::device::lidar::LidarDevice::scan_data_  [protected]
```

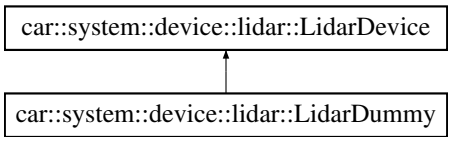
The documentation for this class was generated from the following file:

- include/car/system/device/lidar/LidarDevice.h

6.12 car::system::device::lidar::LidarDummy Class Reference

`#include <LidarDummy.h>`

Inheritance diagram for car::system::device::lidar::LidarDummy:



Public Member Functions

- LidarDummy ()
- void start () final override
- void update () final override
- void stop () final override
- void initialize () final override
- void terminate () final override
- void disconnect () final override

Additional Inherited Members

6.12.1 Constructor & Destructor Documentation

6.12.1.1 LidarDummy()

`car::system::device::lidar::LidarDummy::LidarDummy ( ) [inline]`

6.12.2 Member Function Documentation

6.12.2.1 disconnect()

`void car::system::device::lidar::LidarDummy::disconnect ( ) [inline], [final], [override], [virtual]`

Implements car::system::device::lidar::LidarDevice.

6.12.2.2 initialize()

```
void car::system::device::lidar::LidarDummy::initialize ( )  [inline], [final], [override],
[virtual]
```

Implements car::system::device::lidar::LidarDevice.

6.12.2.3 start()

```
void car::system::device::lidar::LidarDummy::start ( )  [inline], [final], [override], [virtual]
```

Implements car::system::device::lidar::LidarDevice.

6.12.2.4 stop()

```
void car::system::device::lidar::LidarDummy::stop ( )  [inline], [final], [override], [virtual]
```

Implements car::system::device::lidar::LidarDevice.

6.12.2.5 terminate()

```
void car::system::device::lidar::LidarDummy::terminate ( )  [inline], [final], [override],
[virtual]
```

Implements car::system::device::lidar::LidarDevice.

6.12.2.6 update()

```
void car::system::device::lidar::LidarDummy::update ( )  [inline], [final], [override], [virtual]
```

Implements car::system::device::lidar::LidarDevice.

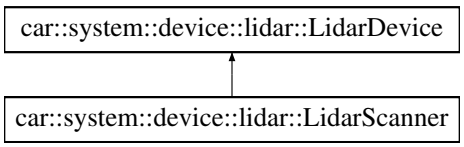
The documentation for this class was generated from the following file:

- include/car/system/device/lidar/LidarDummy.h

6.13 car::system::device::lidar::LidarScanner Class Reference

`#include <LidarScanner.h>`

Inheritance diagram for car::system::device::lidar::LidarScanner:



Public Member Functions

- LidarScanner (std::shared_ptr< configuration::Configuration > configuration, std::unique_ptr< RPLidar > lidar)
- void start () final override
- void update () final override
- void stop () final override
- void initialize () final override
- void disconnect () final override
- void terminate () final override

Static Public Member Functions

- static tl::expected< std::unique_ptr< LidarScanner >, std::string > create (std::shared_ptr< configuration::Configuration > configuration) noexcept

Private Attributes

- std::atomic_bool running = false
- std::shared_ptr< configuration::Configuration > configuration_
- std::vector< Measure > scan_data_
- std::unique_ptr< RPLidar > lidar_
- std::variant< std::function< std::vector< Measure >()>, nullptr_t > scan_generator_ = nullptr
- std::mutex scan_data_mutex_

Additional Inherited Members

6.13.1 Constructor & Destructor Documentation

6.13.1.1 LidarScanner()

```
car::system::device::lidar::LidarScanner::LidarScanner (
            std::shared_ptr< configuration::Configuration > configuration,
            std::unique_ptr< RPLidar > lidar )  [inline]
```

6.13.2 Member Function Documentation

6.13.2.1 create()

```
static tl::expected< std::unique_ptr< LidarScanner >, std::string > car::system::device↩
::lidar::LidarScanner::create (
            std::shared_ptr< configuration::Configuration > configuration )  [inline], [static],
[noexcept]
```

6.13.2.2 disconnect()

```
void car::system::device::lidar::LidarScanner::disconnect ( )  [inline], [final], [override],
[virtual]
```

Implements car::system::device::lidar::LidarDevice.

6.13.2.3 initialize()

```
void car::system::device::lidar::LidarScanner::initialize ( )  [inline], [final], [override],
[virtual]
```

Implements car::system::device::lidar::LidarDevice.

6.13.2.4 start()

```
void car::system::device::lidar::LidarScanner::start ( )  [inline], [final], [override], [virtual]
```

Implements car::system::device::lidar::LidarDevice.

6.13.2.5 stop()

```
void car::system::device::lidar::LidarScanner::stop ( )  [inline], [final], [override], [virtual]
```

Implements car::system::device::lidar::LidarDevice.

6.13.2.6 terminate()

```
void car::system::device::lidar::LidarScanner::terminate ( )  [inline], [final], [override],
[virtual]
```

Implements car::system::device::lidar::LidarDevice.

6.13.2.7 update()

```
void car::system::device::lidar::LidarScanner::update ( )  [inline], [final], [override],
[virtual]
```

Implements car::system::device::lidar::LidarDevice.

6.13.3 Member Data Documentation

6.13.3.1 configuration_

```
std::shared_ptr<configuration::Configuration> car::system::device::lidar::LidarScanner::configuration↩
_  [private]
```

6.13.3.2 lidar_

```
std::unique_ptr<RPLidar> car::system::device::lidar::LidarScanner::lidar_  [private]
```

6.13.3.3 running

```
std::atomic_bool car::system::device::lidar::LidarScanner::running = false  [private]
```

6.13.3.4 scan_data_

```
std::vector<Measure> car::system::device::lidar::LidarScanner::scan_data_  [private]
```

6.13.3.5 scan_data_mutex_

```
std::mutex car::system::device::lidar::LidarScanner::scan_data_mutex_  [private]
```

6.13.3.6 scan_generator_

```
std::variant<std::function<std::vector<Measure>()>, nullptr_t> car::system::device::lidar↩
::LidarScanner::scan_generator_ = nullptr  [private]
```

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

- include/car/system/device/lidar/LidarScanner.h

6.14 car::system::messaging::MessagingSystem Class Reference

```
#include <MessagingSystem.h>
```

Classes

- struct FirstMessageStruct

Public Member Functions

- MessagingSystem ()
- void initialize (std::shared_ptr< configuration::Configuration > configuration)

  Initializes the use of Websockets and initializes the Signals.
- void initializeWebSocket ()

  Creates a new Websocket object for use.
- const tl::expected< nullptr_t, std::string > tryConnect ()

  Attempts to connect to the Websocket server and retrieves the first message from the Websocket (Should be UUID)
- void stop ()
- void terminate ()
- void setConfiguration (std::shared_ptr< configuration::Configuration > configuration)
- nod::signal< void(const std::string, const rapidjson::Document &)> & getCommandSignal ()
- nod::signal< void(const std::string, const rapidjson::Document &)> & getSelectionSignal ()
- nod::signal< void(const std::string)> & getMessageSignal ()
- nod::signal< void(const std::string)> & getDisconnectSignal ()
- void onMessageCallback (const ix::WebSocketMessagePtr &msg) const
- void onDisconnect (const std::string)
- const std::string getUUID () const
- void handleMessage (const std::string &message) const

  Sends out signals depending on the type of message.
- void sendMessage (const std::string &message)
- void onFirstMessage (const ix::WebSocketMessagePtr &msg, FirstMessageStruct &first_message_struct)

  Actually retrieves the First Message from the Websocket to put into FirstMessageStruct.
- const bool isConnected () const

Public Attributes

- nod::signal< void(std::string)> on_disconnect_signal_
- nod::signal< void(const std::string)> message_signal_
- nod::signal< void(const std::string, const rapidjson::Document &)> command_signal_
- nod::signal< void(const std::string, const rapidjson::Document &)> selection_signal_

Private Member Functions

- tl::expected< std::string, std::string > getFirstMessage ()

  Waits and retrieves the first message when connecting to a websocket.

Private Attributes

- std::shared_ptr< configuration::Configuration > configuration_
- std::unique_ptr< ix::WebSocket > websocket_
- std::string websocket_url_
- std::string uuid_
- bool connected_ = false

6.14.1 Constructor & Destructor Documentation

6.14.1.1 MessagingSystem()

```
car::system::messaging::MessagingSystem::MessagingSystem ( )
```

6.14.2 Member Function Documentation

6.14.2.1 getCommandSignal()

```
nod::signal< void(const std::string, const rapidjson::Document &)> & car::system::messaging↩
::MessagingSystem::getCommandSignal ( )  [inline]
```

6.14.2.2 getDisconnectSignal()

```
nod::signal< void(const std::string)> & car::system::messaging::MessagingSystem::getDisconnect↩
Signal ( )  [inline]
```

6.14.2.3 getFirstMessage()

```
tl::expected< std::string, std::string > car::system::messaging::MessagingSystem::getFirst↩
Message ( )  [private]
```

Waits and retrieves the first message when connecting to a websocket.

Returns

tl::expected<std::string, std::string>

6.14.2.4 getMessageSignal()

```
nod::signal< void(const std::string)> & car::system::messaging::MessagingSystem::getMessage↩
Signal ( )  [inline]
```

6.14.2.5 getSelectionSignal()

```
nod::signal< void(const std::string, const rapidjson::Document &)> & car::system::messaging↩
::MessagingSystem::getSelectionSignal ( )  [inline]
```

6.14.2.6 getUUID()

```
const std::string car::system::messaging::MessagingSystem::getUUID ( ) const  [inline]
```

6.14.2.7 handleMessage()

```
void car::system::messaging::MessagingSystem::handleMessage (
            const std::string & message ) const
```

Sends out signals depending on the type of message.

Parameters

message	

6.14.2.8 initialize()

```
void car::system::messaging::MessagingSystem::initialize (
            std::shared_ptr< configuration::Configuration > configuration )
```

Initializes the use of Websockets and initializes the Signals.

Parameters

configuration	

6.14.2.9 initializeWebSocket()

```
void car::system::messaging::MessagingSystem::initializeWebSocket ( )
```

Creates a new Websocket object for use.

6.14.2.10 isConnected()

```
const bool car::system::messaging::MessagingSystem::isConnected ( ) const  [inline]
```

6.14.2.11 onDisconnect()

```
void car::system::messaging::MessagingSystem::onDisconnect (
            const std::string message )
```

6.14.2.12 onFirstMessage()

```
void car::system::messaging::MessagingSystem::onFirstMessage (
            const ix::WebSocketMessagePtr & msg,
            FirstMessageStruct & first_message_struct )
```

Actually retrieves the First Message from the Websocket to put into FirstMessageStruct.

Parameters

msg	
first_message_struct	

6.14.2.13 onMessageCallback()

```
void car::system::messaging::MessagingSystem::onMessageCallback (
            const ix::WebSocketMessagePtr & msg ) const
```

6.14.2.14 sendMessage()

```
void car::system::messaging::MessagingSystem::sendMessage (
            const std::string & message )
```

6.14.2.15 setConfiguration()

```
void car::system::messaging::MessagingSystem::setConfiguration (
            std::shared_ptr< configuration::Configuration > configuration )
```

6.14.2.16 stop()

```
void car::system::messaging::MessagingSystem::stop ( )
```

6.14.2.17 terminate()

```
void car::system::messaging::MessagingSystem::terminate ( )
```

6.14.2.18 tryConnect()

```
const tl::expected< nullptr_t, std::string > car::system::messaging::MessagingSystem::try↩
Connect ( )
```

Attempts to connect to the Websocket server and retrieves the first message from the Websocket (Should be UUID)

Returns

const tl::expected<nullptr_t, std::string>

6.14.3 Member Data Documentation

6.14.3.1 command_signal_

```
nod::signal<void(const std::string, const rapidjson::Document&)> car::system::messaging::↩
MessagingSystem::command_signal_
```

6.14.3.2 configuration_

```
std::shared_ptr<configuration::Configuration> car::system::messaging::MessagingSystem::configuration↩
_  [private]
```

6.14.3.3 connected_

```
bool car::system::messaging::MessagingSystem::connected_ = false  [private]
```

6.14.3.4 message_signal_

```
nod::signal<void(const std::string)> car::system::messaging::MessagingSystem::message_signal↩
_
```

6.14.3.5 on_disconnect_signal_

```
nod::signal<void(std::string)> car::system::messaging::MessagingSystem::on_disconnect_signal↩
_
```

6.14.3.6 selection_signal_

```
nod::signal<void(const std::string, const rapidjson::Document&)> car::system::messaging::↩
MessagingSystem::selection_signal_
```

6.14.3.7 uuid_

```
std::string car::system::messaging::MessagingSystem::uuid_  [private]
```

6.14.3.8 websocket_

```
std::unique_ptr<ix::WebSocket> car::system::messaging::MessagingSystem::websocket_  [private]
```

6.14.3.9 websocket_url_

```
std::string car::system::messaging::MessagingSystem::websocket_url_  [private]
```

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

- include/car/system/messaging/MessagingSystem.h
- src/car/system/messaging/MessagingSystem.cpp

6.15 car::system::movement::MovementSystem Class Reference

```
#include <MovementSystem.h>
```

Public Member Functions

- MovementSystem (std::unique_ptr< AbstractMovementController > movement_controller)
- void initialize ()
- void start ()
- void stop ()
- void terminate ()
- void setRearWheelsSpeed (const int speed) const
- void setRearLeftWheelSpeed (const int speed) const
- void setRearRightWheelSpeed (const int speed) const
- void setFrontWheelsAngle (const float angle) const
- void setCameraServo1Angle (const float angle) const
- void setCameraServo2Angle (const float angle) const
- void setRearWheelsDirectionToForward () const
- void setRearLeftWheelDirectionToForward () const
- void setRearRightWheelDirectionToForward () const
- void setRearWheelsDirectionToBackward () const
- void setRearLeftWheelDirectionToBackward () const
- void setRearRightWheelDirectionToBackward () const
- ~MovementSystem ()

Private Attributes

- std::unique_ptr< AbstractMovementController > movement_controller

6.15.1 Constructor & Destructor Documentation

6.15.1.1 MovementSystem()

```
car::system::movement::MovementSystem::MovementSystem (
            std::unique_ptr< AbstractMovementController > movement_controller )  [inline]
```

6.15.1.2 ~MovementSystem()

```
car::system::movement::MovementSystem::~MovementSystem ( )  [inline]
```

6.15.2 Member Function Documentation

6.15.2.1 initialize()

```
void car::system::movement::MovementSystem::initialize ( )  [inline]
```

6.15.2.2 setCameraServo1Angle()

```
void car::system::movement::MovementSystem::setCameraServo1Angle (
            const float angle ) const  [inline]
```

6.15.2.3 setCameraServo2Angle()

```
void car::system::movement::MovementSystem::setCameraServo2Angle (
            const float angle ) const  [inline]
```

6.15.2.4 setFrontWheelsAngle()

```
void car::system::movement::MovementSystem::setFrontWheelsAngle (
            const float angle ) const  [inline]
```

6.15.2.5 setRearLeftWheelDirectionToBackward()

```
void car::system::movement::MovementSystem::setRearLeftWheelDirectionToBackward ( ) const
[inline]
```

6.15.2.6 setRearLeftWheelDirectionToForward()

```
void car::system::movement::MovementSystem::setRearLeftWheelDirectionToForward ( ) const  [inline]
```

6.15.2.7 setRearLeftWheelSpeed()

```
void car::system::movement::MovementSystem::setRearLeftWheelSpeed (
            const int speed ) const  [inline]
```

6.15.2.8 setRearRightWheelDirectionToBackward()

```
void car::system::movement::MovementSystem::setRearRightWheelDirectionToBackward ( ) const
[inline]
```

6.15.2.9 setRearRightWheelDirectionToForward()

```
void car::system::movement::MovementSystem::setRearRightWheelDirectionToForward ( ) const
[inline]
```

6.15.2.10 setRearRightWheelSpeed()

```
void car::system::movement::MovementSystem::setRearRightWheelSpeed (
            const int speed ) const  [inline]
```

6.15.2.11 setRearWheelsDirectionToBackward()

```
void car::system::movement::MovementSystem::setRearWheelsDirectionToBackward ( ) const  [inline]
```

6.15.2.12 setRearWheelsDirectionToForward()

```
void car::system::movement::MovementSystem::setRearWheelsDirectionToForward ( ) const  [inline]
```

6.15.2.13 setRearWheelsSpeed()

```
void car::system::movement::MovementSystem::setRearWheelsSpeed (
            const int speed ) const  [inline]
```

6.15.2.14 start()

```
void car::system::movement::MovementSystem::start ( )  [inline]
```

6.15.2.15 stop()

```
void car::system::movement::MovementSystem::stop ( )  [inline]
```

6.15.2.16 terminate()

```
void car::system::movement::MovementSystem::terminate ( )  [inline]
```

6.15.3 Member Data Documentation

6.15.3.1 movement_controller

```
std::unique_ptr<AbstractMovementController> car::system::movement::MovementSystem::movement_↩
controller  [private]
```

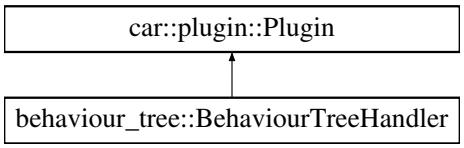
The documentation for this class was generated from the following file:

- include/car/system/movement/MovementSystem.h

6.16 car::plugin::Plugin Class Reference

`#include <Plugin.h>`

Inheritance diagram for car::plugin::Plugin:



Public Member Functions

- virtual void initialize (std::shared_ptr< car::system::CarSystem > car_system)=0
- virtual void update ()=0
- virtual void stop ()=0
- virtual std::string getName ()=0

6.16.1 Member Function Documentation

6.16.1.1 getName()

```
virtual std::string car::plugin::Plugin::getName ( )  [pure virtual]
```

Implemented in behaviour_tree::BehaviourTreeHandler.

6.16.1.2 initialize()

```
virtual void car::plugin::Plugin::initialize (
            std::shared_ptr< car::system::CarSystem > car_system )  [pure virtual]
```

Implemented in behaviour_tree::BehaviourTreeHandler.

6.16.1.3 stop()

```
virtual void car::plugin::Plugin::stop ( )  [pure virtual]
```

Implemented in behaviour_tree::BehaviourTreeHandler.

6.16.1.4 update()

```
virtual void car::plugin::Plugin::update ( )  [pure virtual]
```

Implemented in behaviour_tree::BehaviourTreeHandler.

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

- include/car/plugin/Plugin.h

6.17 car::plugin::PluginManager Class Reference

```
#include <PluginManager.h>
```

Public Member Functions

- void initialize (std::shared_ptr< system::CarSystem > car_system)
- void update ()
- void stop ()
- void terminate ()
- void addPlugin (std::shared_ptr< Plugin > plugin)
- template<typename T >
  std::shared_ptr< T > getPlugin ()

Private Attributes

- std::vector< std::shared_ptr< Plugin > > plugins

6.17.1 Member Function Documentation

6.17.1.1 addPlugin()

```
void car::plugin::PluginManager::addPlugin (
            std::shared_ptr< Plugin > plugin )  [inline]
```

6.17.1.2 getPlugin()

```
template<typename T >
std::shared_ptr< T > car::plugin::PluginManager::getPlugin ( )  [inline]
```

6.17.1.3 initialize()

```
void car::plugin::PluginManager::initialize (
            std::shared_ptr< system::CarSystem > car_system )  [inline]
```

6.17.1.4 stop()

```
void car::plugin::PluginManager::stop ( )  [inline]
```

6.17.1.5 terminate()

```
void car::plugin::PluginManager::terminate ( )  [inline]
```

6.17.1.6 update()

```
void car::plugin::PluginManager::update ( )  [inline]
```

6.17.2 Member Data Documentation

6.17.2.1 plugins

```
std::vector<std::shared_ptr<Plugin> > car::plugin::PluginManager::plugins  [private]
```

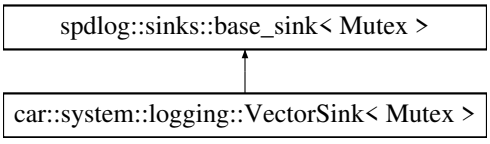
The documentation for this class was generated from the following file:

- include/car/plugin/PluginManager.h

6.18 car::system::logging::VectorSink< Mutex > Class Template Reference

```
#include <VectorSink.h>
```

Inheritance diagram for car::system::logging::VectorSink< Mutex >:



Public Member Functions

- VectorSink (int max_lines)
- void sink_it_ (const spdlog::details::log_msg &msg) override
- void flush_ () override
- const std::vector< std::string > & get_log_messages () const

Private Attributes

- const int max_lines
- std::vector< std::string > log_messages

6.18.1 Constructor & Destructor Documentation

6.18.1.1 VectorSink()

```
template<typename Mutex >
car::system::logging::VectorSink< Mutex >::VectorSink (
            int max_lines )  [inline]
```

6.18.2 Member Function Documentation

6.18.2.1 flush_()

```
template<typename Mutex >
void car::system::logging::VectorSink< Mutex >::flush_ ( )  [inline], [override]
```

6.18.2.2 get_log_messages()

```
template<typename Mutex >
const std::vector< std::string > & car::system::logging::VectorSink< Mutex >::get_log_↩
messages ( ) const  [inline]
```

6.18.2.3 sink_it_()

```
template<typename Mutex >
void car::system::logging::VectorSink< Mutex >::sink_it_ (
            const spdlog::details::log_msg & msg )  [inline], [override]
```

6.18.3 Member Data Documentation

6.18.3.1 log_messages

```
template<typename Mutex >
std::vector<std::string> car::system::logging::VectorSink< Mutex >::log_messages  [private]
```

6.18.3.2 max_lines

```
template<typename Mutex >
const int car::system::logging::VectorSink< Mutex >::max_lines  [private]
```

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

- include/car/system/logging/VectorSink.h

Chapter 7

File Documentation

7.1 include/behaviour_tree/BehaviourTreeHandler.hpp File Reference

```
#include <string>
#include <vector>
#include <nod/nod.hpp>
#include "utils/Utility.hpp"
#include "car/plugin/Plugin.h"
#include "behaviour_tree/BehaviourTreeParser.hpp"
#include "behaviour_tree/node/custom/CarCustomNodeParser.hpp"
#include "CarContext.hpp"
```

Classes

- class behaviour_tree::BehaviourTreeHandler

Namespaces

- namespace behaviour_tree

7.2 BehaviourTreeHandler.hpp

Go to the documentation of this file.

```
1 #ifndef BEHAVIOURTREEHANDLER_HPP
2 #define BEHAVIOURTREEHANDLER_HPP
3
4 #pragma once
5
6 #include <string>
7 #include <vector>
8
9 #include <nod/nod.hpp>
10
11 #include "utils/Utility.hpp"
12
13 #include "car/plugin/Plugin.h"
14
15 #include "behaviour_tree/BehaviourTreeParser.hpp"
16 #include "behaviour_tree/node/custom/CarCustomNodeParser.hpp"
17
```

```
18 #include "CarContext.hpp"
19
20 namespace behaviour_tree
21 {
22     class BehaviourTreeHandler : public car::plugin::Plugin
23     {
24     public:
25         void initialize(std::shared_ptr<car::system::CarSystem> car_system) final override
26         {
27             this->car_system = car_system;
28             // The BehaviourTreeParser does not come with a CustomNodeParser since each program can have
    a different set of Action nodes
29
    BehaviourTreeParser::instance().setCustomNodeParser(std::make_shared<node::custom::CarCustomNodeParser>(CarCustomNodePa
30
    this->car_system->getMessagingSystem()->getCommandSignal().connect(std::bind(&BehaviourTreeHandler::handleCommand,
    this, std::placeholders::_1, std::placeholders::_2));
31         }
32
33         void handleCommand(const std::string message, const rapidjson::Document &message_json)
34         {
35             const std::string command = message_json["command"].GetString();
36             if (command != "behaviour_tree")
37             {
38                 spdlog::error(R"(The property "command" does not match "behaviour_tree", {})", command);
39                 return;
40             }
41             if (!message_json.HasMember("action") || !message_json["action"].IsString())
42             {
43                 spdlog::error(R"(The property "action" does not exist in the given json.)");
44                 return;
45             }
46             const std::string action = message_json["action"].GetString();
47             switch (utils::hash(action))
48             {
49             case utils::hash("set"):
50             {
51                 this->setBehaviourTree(message_json);
52                 break;
53             }
54             case utils::hash("start"):
55             {
56                 this->startBehaviourTree();
57                 break;
58             }
59             case utils::hash("stop"):
60             {
61                 this->stopBehaviourTree();
62                 break;
63             }
64             default:
65                 spdlog::error(R"(The property "action" does not match "set" or "start", {})", action);
66                 break;
67             };
68         }
69
70         void setBehaviourTree(const rapidjson::Document &message_json)
71         {
72             if (!message_json.HasMember("data") || !message_json["data"].IsString())
73             {
74                 spdlog::error(R"(The property "data" does not exist in the given json.)");
75                 return;
76             }
77             try
78             {
79                 auto maybe_behaviour_tree =
    BehaviourTreeParser::instance().parseXML(message_json["data"].GetString());
80                 if (!maybe_behaviour_tree.has_value())
81                 {
82                     spdlog::error(R"(Unable to parse the given behaviour tree | {})",
    maybe_behaviour_tree.error());
83                     return;
84                 }
85                 auto &behaviour_tree = maybe_behaviour_tree.value();
86                 spdlog::info("Behaviour tree parsed successfully | {}", behaviour_tree->toString());
87                 this->_setBehaviourTree(behaviour_tree);
88             }
89             catch (std::exception &e)
90             {
91                 spdlog::error("An error has occurred while parsing the given behaviour tree: {}",
    e.what());
92             }
93         }
94
95         void startBehaviourTree()
96         {
97             assert(this->car_system != nullptr);
```

```
98              if (this->behaviour_tree == nullptr)
99              {
100                 spdlog::error("The Behaviour tree has not been set");
101                 return;
102             }
103             this->behaviour_tree->resetCycles();
104             this->tick_count = 0;
105             std::shared_ptr<Context> context = std::make_shared<CarContext>(this->behaviour_tree,
    this->car_system);
106             this->context = context;
107             spdlog::info("Starting the given Behaviour tree");
108         }
109
110         void stopBehaviourTree()
111         {
112             assert(this->car_system != nullptr);
113             this->context = nullptr;
114             spdlog::info("Stopped any Behaviour Tree context");
115         }
116
117         void update() final override
118         {
119             if (this->context == nullptr)
120             {
121                 return;
122             }
123             if (this->context->canRun())
124             {
125                 const std::chrono::time_point<std::chrono::steady_clock> now =
    std::chrono::steady_clock::now();
126                 // TODO:
127                 if (now - this->last_connected >=
    this->car_system->getConfiguration()->behaviour_tree_update_ms_interval) {
128                     this->context->update(this->tick_count);
129                     this->tick_count++;
130                     this->last_connected = now;
131                 }
132             }
133             else
134             {
135                 this->context = nullptr;
136             }
137         }
138
139         void stop() final override
140         {
141             this->context = nullptr;
142         }
143
144         std::string getName() final override
145         {
146             return "BehaviourTreeHandler";
147         }
148
149         void _setBehaviourTree(std::shared_ptr<BehaviourTree> behaviour_tree)
150         {
151             this->behaviour_tree = behaviour_tree;
152         }
153
154     private:
155         std::shared_ptr<car::system::CarSystem> car_system;
156
157         std::shared_ptr<BehaviourTree> behaviour_tree;
158         std::shared_ptr<Context> context;
159
160         int tick_count = 0;
161
162         // This is initialized as 0
163         std::chrono::time_point<std::chrono::steady_clock> last_connected;
164     };
165 } // namespace behaviour_tree
166
167 #endif
```

7.3 include/behaviour_tree/CarContext.hpp File Reference

```
#include "car/system/CarSystem.h"
#include "behaviour_tree/Context.h"
```

Classes

- class behaviour_tree::CarContext

Namespaces

- namespace behaviour_tree

7.4 CarContext.hpp

Go to the documentation of this file.

```
1 #ifndef BEHAVIOUR_TREE_CARCONTEXT_HPP
2 #define BEHAVIOUR_TREE_CARCONTEXT_HPP
3
4 #pragma once
5
6 #include "car/system/CarSystem.h"
7 #include "behaviour_tree/Context.h"
8
9 namespace behaviour_tree
10 {
11     class CarContext : public Context
12     {
13     public:
14         CarContext(std::shared_ptr<BehaviourTree> behaviour_tree, std::shared_ptr<car::system::CarSystem>
      car_system) : Context(std::move(behaviour_tree)), car_system(std::move(car_system))
15         {
16         }
17
18         std::shared_ptr<car::system::CarSystem> getCarSystem() const
19         {
20             return this->car_system;
21         }
22
23         void _() override{};
24
25     private:
26         std::shared_ptr<car::system::CarSystem> car_system;
27     };
28 }
29
30 #endif
```

7.5 include/car/configuration/Configuration.h File Reference

```
#include <chrono>
#include <optional>
#include <string>
#include <tl/expected.hpp>
```

Classes

- struct car::configuration::Configuration

Namespaces

- namespace car
- namespace car::configuration

7.6 Configuration.h

Go to the documentation of this file.

```
1 #ifndef CONFIGURATION_H
2 #define CONFIGURATION_H
3
4 #pragma once
5
6 #include <chrono>
7 #include <optional>
8 #include <string>
9
10 #include <tl/expected.hpp>
11
12 namespace car::configuration
13 {
14     struct Configuration
15     {
16         std::string host = "127.0.0.1:3000";
17
18         int camera_index = 0;
19         void setCameraFps(const int camera_fps)
20         {
21             this->camera_fps = camera_fps;
22             this->camera_fps_interval = 1000 / camera_fps;
23         }
24         const int getCameraFpsInterval() { return this->camera_fps_interval; }
25         bool use_camera = true;
26
27         std::string lidar_port = "";
28         bool use_lidar = true;
29
30         std::chrono::milliseconds behaviour_tree_update_ms_interval = std::chrono::milliseconds(100);
31
32     private:
33         int camera_fps = 60;
34         int camera_fps_interval = 1000;
35     };
36 };
37
38 #endif
```

7.7 include/car/plugin/Plugin.h File Reference

```
#include <string>
#include <memory>
```

Classes

- class car::plugin::Plugin

Namespaces

- namespace car
- namespace car::system
- namespace car::plugin

7.8 Plugin.h

Go to the documentation of this file.

```
1 #ifndef PLUGIN_H
2 #define PLUGIN_H
3
4 #pragma once
5
6 #include <string>
7 #include <memory>
8
9 namespace car::system
10 {
11     class CarSystem;
12 }
13
14 namespace car::plugin
15 {
16     class Plugin
17     {
18     public:
19         virtual void initialize(std::shared_ptr<car::system::CarSystem> car_system) = 0;
20         virtual void update() = 0;
21         virtual void stop() = 0;
22         virtual std::string getName() = 0;
23     };
24 }
25
26 #endif
```

7.9 include/car/plugin/PluginManager.h File Reference

```
#include <vector>
#include <memory>
#include "utils/Utility.hpp"
#include "utils/TypeName.hpp"
#include "Plugin.h"
```

Classes

- class car::plugin::PluginManager

Namespaces

- namespace car
- namespace car::system
- namespace car::plugin

7.10 PluginManager.h

Go to the documentation of this file.

```
1 #ifndef PLUGIN_MANAGER_H
2 #define PLUGIN_MANAGER_H
3
4 #pragma once
5
6 #include <vector>
7 #include <memory>
8
9 #include "utils/Utility.hpp"
```

```
10 #include "utils/TypeName.hpp"
11
12 #include "Plugin.h"
13
14 namespace car::system
15 {
16     class CarSystem;
17 }
18
19 namespace car::plugin
20 {
21     class PluginManager
22     {
23     public:
24         void initialize(std::shared_ptr<system::CarSystem> car_system)
25         {
26             for (std::shared_ptr<Plugin>& plugin : this->plugins)
27             {
28                 plugin->initialize(car_system);
29             }
30         }
31
32         void update()
33         {
34             for (std::shared_ptr<Plugin>& plugin : this->plugins)
35             {
36                 plugin->update();
37             }
38         }
39
40         void stop()
41         {
42             for (std::shared_ptr<Plugin>& plugin : this->plugins)
43             {
44                 plugin->stop();
45             }
46         }
47
48         void terminate()
49         {
50             this->stop();
51         }
52
53         void addPlugin(std::shared_ptr<Plugin> plugin)
54         {
55             this->plugins.push_back(plugin);
56         }
57
58         template<typename T>
59         std::shared_ptr<T> getPlugin()
60         {
61             static_assert(std::is_base_of<Plugin, T>::value, "T must be a Plugin");
62             std::string type_name = std::string(utils::TypeName<T>());
63             type_name = utils::getStringAfterLastColon(type_name);
64
65             for (std::shared_ptr<Plugin>& plugin : this->plugins)
66             {
67                 if (plugin->getName() == type_name)
68                 {
69                     return plugin;
70                 }
71             }
72
73             return nullptr;
74         }
75
76
77     private:
78         std::vector<std::shared_ptr<Plugin» plugins;
79     };
80 }
81
82 #endif
```

7.11 include/car/system/CarSystem.h File Reference

```
#include <memory>
#include "car/configuration/Configuration.h"
#include "car/system/device/DeviceManager.h"
#include "car/system/messaging/MessagingSystem.h"
```

```
#include "car/system/movement/MovementSystem.h"
#include "car/plugin/PluginManager.h"
```

Classes

- class car::system::CarSystem

Namespaces

- namespace car
- namespace car::system

7.12 CarSystem.h

Go to the documentation of this file.

```
1 #ifndef CARSYSTEM_H
2 #define CARSYSTEM_H
3
4 #pragma once
5
6 #include <memory>
7
8 #include "car/configuration/Configuration.h"
9
10 #include "car/system/device/DeviceManager.h"
11 #include "car/system/messaging/MessagingSystem.h"
12 #include "car/system/movement/MovementSystem.h"
13
14 #include "car/plugin/PluginManager.h"
15
16 using namespace car::configuration;
17 using namespace car::plugin;
18 using namespace car::system::device;
19 using namespace car::system::messaging;
20 using namespace car::system::movement;
21
22 namespace car::system
23 {
24     // Make sure this is stored as a shared_ptr
25     class CarSystem : public std::enable_shared_from_this<CarSystem>
26     {
27     public:
28         CarSystem(
29             std::shared_ptr<Configuration> configuration,
30             std::unique_ptr<DeviceManager> device_manager,
31             std::unique_ptr<MessagingSystem> messaging_system,
32             std::unique_ptr<MovementSystem> movement_system,
33             std::unique_ptr<PluginManager> plugin_manager);
34
35         void initialize();
36         void reload();
37
38         void start();
39         void stop();
40
41         tl::expected<nullptr_t, std::string> tryConnect();
42         void disconnect();
43
44         void terminate();
45
46         void update();
47
48         const std::shared_ptr<Configuration> getConfiguration() const { return this->configuration_; };
49         void setConfiguration(std::shared_ptr<Configuration> configuration);
50
51         DeviceManager *getDeviceManager() const
52         {
53             return this->device_manager_.get();
54         }
55
```

```
56          MessagingSystem *getMessagingSystem() const
57          {
58              return this->messaging_system_.get();
59          }
60
61          MovementSystem *getMovementSystem() const
62          {
63              return this->movement_system_.get();
64          }
65
66          template <typename T>
67          const std::shared_ptr<T> getPlugin() const { return this->plugin_manager_->getPlugin<T>(); }
68
69      private:
70          void sendData();
71
72          std::shared_ptr<Configuration> configuration_;
73
74          const std::unique_ptr<DeviceManager> device_manager_;
75          const std::unique_ptr<MessagingSystem> messaging_system_;
76          const std::unique_ptr<MovementSystem> movement_system_;
77          const std::unique_ptr<PluginManager> plugin_manager_;
78
79          bool initialized = false;
80          bool started = false;
81      };
82  }
83
84  #endif
```

7.13 include/car/system/device/CameraDevice.h File Reference

```
#include <vector>
#include <tl/expected.hpp>
#include <opencv2/opencv.hpp>
#include "car/configuration/Configuration.h"
```

Classes

- class car::system::device::CameraDevice

Namespaces

- namespace car
- namespace car::system
- namespace car::system::device

7.14 CameraDevice.h

Go to the documentation of this file.

```
1  #ifndef CAMERADEVICE_H
2  #define CAMERADEVICE_H
3
4  #pragma once
5
6  #include <vector>
7
8  #include <tl/expected.hpp>
9  #include <opencv2/opencv.hpp>
10
11 #include "car/configuration/Configuration.h"
12
13 namespace car::system::device
```

```
14 {
15     class DeviceManager;
16     class CameraDevice
17     {
18     public:
19         CameraDevice(std::shared_ptr<configuration::Configuration> configuration) :
       configuration(configuration) {}
20
21         CameraDevice(const CameraDevice&) = delete;
22         CameraDevice& operator=(const CameraDevice&) = delete;
23
24         CameraDevice(CameraDevice&&) = delete;
25         CameraDevice& operator=(CameraDevice&&) = delete;
26
27         ~CameraDevice() = default;
28
29     public:
30         [[nodiscard]] static tl::expected<std::unique_ptr<CameraDevice>, std::string>
       create(std::shared_ptr<configuration::Configuration> configuration);
31         std::string getFrameBuffer() const;
32
33     protected:
34         void start();
35         void update();
36         void stop();
37         void disconnect();
38         void terminate();
39
40         friend class DeviceManager;
41
42     private:
43         std::shared_ptr<configuration::Configuration> configuration;
44
45         std::unique_ptr<cv::VideoCapture> camera_;
46
47         bool connected_ = false;
48         std::string frame_buffer_;
49
50         std::mutex camera_mutex_;
51
52         std::chrono::steady_clock::time_point last;
53     };
54 }
55
56 #endif
```

7.15 include/car/system/device/DeviceManager.h File Reference

```
#include <memory>
#include <tl/expected.hpp>
#include "car/configuration/Configuration.h"
#include "CameraDevice.h"
#include "lidar/LidarDevice.h"
#include "lidar/LidarScanner.h"
```

Classes

- class car::system::device::DeviceManager

Namespaces

- namespace car
- namespace car::system
- namespace car::system::device

7.16 DeviceManager.h

Go to the documentation of this file.

```
1 #ifndef DEVICE_MANAGER_H
2 #define DEVICE_MANAGER_H
3
4 #pragma once
5
6 #include <memory>
7
8 #include <tl/expected.hpp>
9
10 #include "car/configuration/Configuration.h"
11
12 #include "CameraDevice.h"
13 #include "lidar/LidarDevice.h"
14 #include "lidar/LidarScanner.h"
15
16 using namespace car::configuration;
17
18 namespace car::system
19 {
20     class CarSystem;
21 }
22
23 namespace car::system::device
24 {
25     class DeviceManager {
26     public:
27         [[nodiscard]] static tl::expected<std::unique_ptr<DeviceManager>, std::string>
      create(std::shared_ptr<Configuration> configuration);
28
29         DeviceManager(std::unique_ptr<CameraDevice> camera_device, std::unique_ptr<lidar::LidarDevice>
      lidar_device) :
30             camera_device_(std::move(camera_device)),
31             lidar_device_(std::move(lidar_device))
32         {
33         }
34
35         CameraDevice* getCameraDevice() {
36             return this->camera_device_.get();
37         }
38
39         lidar::LidarDevice* getLidarDevice() {
40             return this->lidar_device_.get();
41         }
42
43         const bool isRunning() const {
44             return this->is_running_;
45         }
46
47         void initialize(std::shared_ptr<system::CarSystem> car_system);
48         void start();
49         void update();
50         void stop();
51         void terminate();
52
53     private:
54         std::shared_ptr<car::system::CarSystem> car_system;
55
56         bool is_initialized_ = false;
57         bool is_running_ = false;
58
59         std::unique_ptr<lidar::LidarDevice> lidar_device_;
60         std::unique_ptr<CameraDevice> camera_device_;
61     };
62 }
63
64 #endif
```

7.17 include/car/system/device/lidar/LidarDevice.h File Reference

```
#include <vector>
#include <rapidjson/document.h>
#include <RPLidar.h>
```

Classes

- class car::system::device::lidar::LidarDevice

Namespaces

- namespace car
- namespace car::system
- namespace car::system::device
- namespace car::system::device::lidar

7.18 LidarDevice.h

Go to the documentation of this file.

```
1 #ifndef LIDARDEVICE_H
2 #define LIDARDEVICE_H
3
4 #pragma once
5
6 #include <vector>
7
8 #include <rapidjson/document.h>
9
10 #include <RPLidar.h>
11
12 using namespace rplidar;
13
14 namespace car::system::device {
15     class DeviceManager;
16 }
17
18 namespace car::system::device::lidar
19 {
20     class LidarDevice
21     {
22     public:
23         std::vector<Measure> getScanData() const { return this->scan_data_; }
24
25         virtual void start() = 0;
26         virtual void update() = 0;
27         virtual void stop() = 0;
28
29         virtual void initialize() = 0;
30         virtual void terminate() = 0;
31         virtual void disconnect() = 0;
32
33     protected:
34         friend class DeviceManager;
35
36         void setScanData(const std::vector<Measure>& scan_data)
37         {
38             this->scan_data_ = scan_data;
39         }
40
41         std::vector<Measure> scan_data_;
42     };
43 }
44
45 #endif
```

7.19 include/car/system/device/lidar/LidarDummy.h File Reference

```
#include <fstream>
#include <spdlog/spdlog.h>
#include "LidarDevice.h"
```

Classes

- class car::system::device::lidar::LidarDummy

Namespaces

- namespace car
- namespace car::system
- namespace car::system::device
- namespace car::system::device::lidar

7.20 LidarDummy.h

Go to the documentation of this file.

```
1 #ifndef LIDARDUMMY_H
2 #define LIDARDUMMY_H
3
4 #pragma once
5
6 #include <fstream>
7 #include <spdlog/spdlog.h>
8
9 #include "LidarDevice.h"
10
11 namespace car::system::device::lidar
12 {
13     class LidarDummy final : public LidarDevice
14     {
15     public:
16         LidarDummy()
17         {
18             spdlog::warn("Currently using the LidarDummy");
19         };
20
21         void start() final override {};
22         void update() final override {};
23         void stop() final override {};
24         void initialize() final override {};
25         void terminate() final override {};
26         void disconnect() final override {};
27
28     private:
29     };
30 }
31
32 #endif
```

7.21 include/car/system/device/lidar/LidarScanner.h File Reference

```
#include "LidarDevice.h"
#include <memory>
#include <variant>
#include <RPLidar.h>
#include <tl/expected.hpp>
#include "car/configuration/Configuration.h"
```

Classes

- class car::system::device::lidar::LidarScanner

Namespaces

- namespace car
- namespace car::system
- namespace car::system::device
- namespace car::system::device::lidar

7.22 LidarScanner.h

Go to the documentation of this file.

```
1 #ifndef LIDARSCANNER_H
2 #define LIDARSCANNER_H
3
4 #pragma once
5
6 #include "LidarDevice.h"
7
8 #include <memory>
9 #include <variant>
10
11 #include <RPLidar.h>
12 #include <tl/expected.hpp>
13
14 #include "car/configuration/Configuration.h"
15
16 using namespace rplidar;
17
18 namespace car::system::device::lidar
19 {
20     class LidarScanner final : public LidarDevice
21     {
22     public:
23         [[nodiscard]] static tl::expected<std::unique_ptr<LidarScanner>, std::string>
      create(std::shared_ptr<configuration::Configuration> configuration) noexcept
24         {
25             auto maybe_lidar = RPLidar::create(configuration->lidar_port);
26             if (maybe_lidar.has_value())
27             {
28                 return std::make_unique<LidarScanner>(configuration, std::move(maybe_lidar.value()));
29             }
30             else
31             {
32                 return tl::make_unexpected(maybe_lidar.error());
33             }
34         }
35
36         // Do not call this constructor directly. Use the create method instead.
37         LidarScanner(std::shared_ptr<configuration::Configuration> configuration,
      std::unique_ptr<RPLidar> lidar) : configuration_(configuration), lidar_(std::move(lidar)) {
38         }
39
40         void start() final override
41         {
42             this->running = true;
43             this->lidar_->start_motor();
44             std::lock_guard<std::mutex> lock(this->scan_data_mutex_);
45             this->scan_generator_ = this->lidar_->iter_scans();
46         };
47
48         void update() final override
49         {
50             if (this->running) {
51                 std::lock_guard<std::mutex> lock(this->scan_data_mutex_);
52                 const auto& scan_generator =
      std::get<std::function<std::vector<Measure>()>>(this->scan_generator_);
53                 this->setScanData(scan_generator());
54             }
55         };
56
57         void stop() final override
58         {
59             if (this->running) {
60                 this->running = false;
61                 std::lock_guard<std::mutex> lock(this->scan_data_mutex_);
62                 this->scan_generator_ = nullptr;
63                 this->lidar_->stop();
64                 this->lidar_->stop_motor();
65             }
```

```
66          }
67
68          void initialize() final override
69          {
70          };
71
72          void disconnect() final override
73          {
74              if (this->running) {
75                  this->running = false;
76                  std::lock_guard<std::mutex> lock(this->scan_data_mutex_);
77                  this->scan_generator_ = nullptr;
78                  this->lidar_->disconnect();
79              }
80          }
81
82          void terminate() final override
83          {
84              this->stop();
85              this->disconnect();
86          }
87
88      private:
89          std::atomic_bool running = false;
90
91          std::shared_ptr<configuration::Configuration> configuration_;
92
93          std::vector<Measure> scan_data_;
94
95          std::unique_ptr<RPLidar> lidar_;
96          std::variant<std::function<std::vector<Measure>()>, nullptr_t> scan_generator_ = nullptr;
97
98          std::mutex scan_data_mutex_;
99      };
100 }
101
102 #endif
```

7.23 include/car/system/logging/VectorSink.h File Reference

```
#include <algorithm>
#include <vector>
#include <fmt/format.h>
#include <spdlog/sinks/base_sink.h>
#include <spdlog/details/synchronous_factory.h>
#include <iostream>
```

Classes

- class car::system::logging::VectorSink< Mutex >

Namespaces

- namespace car
- namespace car::system
- namespace car::system::logging

Typedefs

- using car::system::logging::vector_sink_mt = VectorSink< std::mutex >

7.24 VectorSink.h

Go to the documentation of this file.

```
1 #ifndef VECTORSINK_CXX
2 #define VECTORSINK_CXX
3
4 #include <algorithm>
5 #include <vector>
6
7 #include <fmt/format.h>
8
9 #include <spdlog/sinks/base_sink.h>
10 #include <spdlog/details/synchronous_factory.h>
11 #include <iostream>
12
13 namespace car::system::logging
14 {
15     template <typename Mutex>
16     class VectorSink : public spdlog::sinks::base_sink<Mutex>
17     {
18     public:
19         VectorSink(int max_lines) : max_lines(max_lines)
20         {
21         }
22
23         void sink_it_(const spdlog::details::log_msg &msg) override
24         {
25             spdlog::memory_buf_t formatted;
26             spdlog::sinks::base_sink<Mutex>::formatter_->format(msg, formatted);
27             if (this->log_messages.size() < this->max_lines)
28             {
29                 this->log_messages.push_back(std::string(formatted.data(), formatted.size()));
30             }
31             else
32             {
33                 std::rotate(this->log_messages.begin(), this->log_messages.begin() + 1,
      this->log_messages.end());
34                 this->log_messages[this->log_messages.size() - 1] = std::string(formatted.data(),
      formatted.size());
35             }
36         };
37
38         void flush_() override
39         {
40             this->log_messages.clear();
41         };
42
43         const std::vector<std::string> &get_log_messages() const
44         {
45             return this->log_messages;
46         }
47
48     private:
49         const int max_lines;
50
51         std::vector<std::string> log_messages;
52     };
53     using vector_sink_mt = VectorSink<std::mutex>;
54 }
55
56 #endif
```

7.25 include/car/system/messaging/MessagingSystem.h File Reference

```
#include <functional>
#include <memory>
#include <ixwebsocket/IXNetSystem.h>
#include <ixwebsocket/IXWebSocket.h>
#include <nod/nod.hpp>
#include <rapidjson/rapidjson.h>
#include <rapidjson/document.h>
#include "utils/Utility.hpp"
#include "car/configuration/Configuration.h"
```

Classes

- class car::system::messaging::MessagingSystem
- struct car::system::messaging::MessagingSystem::FirstMessageStruct

Namespaces

- namespace car
- namespace car::system
- namespace car::system::messaging

7.26 MessagingSystem.h

Go to the documentation of this file.

```
1 #ifndef MESSAGINGSYSTEM_H
2 #define MESSAGINGSYSTEM_H
3
4 #pragma once
5
6 #include <functional>
7 #include <memory>
8
9 #include <ixwebsocket/IXNetSystem.h>
10 #include <ixwebsocket/IXWebSocket.h>
11
12 #include <nod/nod.hpp>
13
14 #include <rapidjson/rapidjson.h>
15 #include <rapidjson/document.h>
16
17 #include "utils/Utility.hpp"
18
19 #include "car/configuration/Configuration.h"
20
21 namespace car::system::messaging
22 {
23     class MessagingSystem
24     {
25     public:
26         MessagingSystem();
27
28         void initialize(std::shared_ptr<configuration::Configuration> configuration);
29         void initializeWebSocket();
30         const tl::expected<nullptr_t, std::string> tryConnect();
31         void stop();
32         void terminate();
33
34         // Necessary for the reloading the configuration
35         void setConfiguration(std::shared_ptr<configuration::Configuration> configuration);
36
37         nod::signal<void(const std::string, const rapidjson::Document&)>& getCommandSignal() { return
      this->command_signal_; }
38         nod::signal<void(const std::string, const rapidjson::Document&)>& getSelectionSignal() { return
      this->selection_signal_; }
39         nod::signal<void(const std::string)>& getMessageSignal() { return this->message_signal_; }
40         nod::signal<void(const std::string)>& getDisconnectSignal() { return this->on_disconnect_signal_;
      }
41
42         void onMessageCallback(const ix::WebSocketMessagePtr& msg) const;
43         void onDisconnect(const std::string);
44
45         const std::string getUUID() const { return this->uuid_; }
46         void handleMessage(const std::string& message) const;
47         void sendMessage(const std::string& message);
48
49         struct FirstMessageStruct
50         {
51             std::string error_message;
52             std::string uuid;
53             std::condition_variable condition;
54         };
55         void onFirstMessage(const ix::WebSocketMessagePtr& msg, FirstMessageStruct&
      first_message_struct);
56
```

```
57          const bool isConnected() const { return this->connected_; }
58
59          nod::signal<void(std::string)> on_disconnect_signal_;
60
61          nod::signal<void(const std::string)> message_signal_;
62          nod::signal<void(const std::string, const rapidjson::Document&)> command_signal_;
63          nod::signal<void(const std::string, const rapidjson::Document&)> selection_signal_;
64
65      private:
66          tl::expected<std::string, std::string> getFirstMessage();
67
68          std::shared_ptr<configuration::Configuration> configuration_;
69
70          std::unique_ptr<ix::WebSocket> websocket_;
71          std::string websocket_url_;
72
73          std::string uuid_;
74
75          bool connected_ = false;
76      };
77 };
78
79 #endif
```

7.27 include/car/system/messaging/StreamType.h File Reference

Enumerations

- enum StreamType { None = 0 , Lidar , Camera , Both }

7.27.1 Enumeration Type Documentation

7.27.1.1 StreamType

```
enum StreamType
```

Enumerator

None	
Lidar	
Camera	
Both	

7.28 StreamType.h

Go to the documentation of this file.

```
1 #ifndef STREAM_TYPE_H
2 #define STREAM_TYPE_H
3
4 #pragma once
5
6 enum StreamType {
7     None = 0,
8     Lidar,
9     Camera,
```

```
10     Both,
11 };
12
13 #endif
```

7.29 include/car/system/movement/controller/AbstractMovement↩ Controller.h File Reference

Classes

- class car::system::movement::controller::AbstractMovementController

Namespaces

- namespace car
- namespace car::system
- namespace car::system::movement
- namespace car::system::movement::controller

7.30 AbstractMovementController.h

Go to the documentation of this file.

```
1 #ifndef ABSTRACTWHEELCONTROLLER_H
2 #define ABSTRACTWHEELCONTROLLER_H
3
4 #pragma once
5
6 namespace car::system::movement::controller
7 {
8     class AbstractMovementController
9     {
10     public:
11         virtual void initialize() = 0;
12         virtual void stop() = 0;
13         virtual void terminate() = 0;
14
15         virtual void setRearWheelsSpeed(const int speed) = 0;
16
17         virtual void setRearLeftWheelSpeed(const int speed) = 0;
18         virtual void setRearRightWheelSpeed(const int speed) = 0;
19
20         virtual void setFrontWheelsAngle(const float angle) = 0;
21         virtual void setCameraServo1Angle(const float angle) = 0;
22         virtual void setCameraServo2Angle(const float angle) = 0;
23
24         virtual void setRearWheelsDirectionToForward() = 0;
25         virtual void setRearLeftWheelDirectionToForward() = 0;
26         virtual void setRearRightWheelDirectionToForward() = 0;
27
28         virtual void setRearWheelsDirectionToBackward() = 0;
29         virtual void setRearLeftWheelDirectionToBackward() = 0;
30         virtual void setRearRightWheelDirectionToBackward() = 0;
31     };
32 } // namespace car::system::movement::controller
33
34 #endif
```

7.31 include/car/system/movement/controller/DeviceMovementController.h File Reference

7.32 DeviceMovementController.h

Go to the documentation of this file.

```
1 #ifdef __linux__
2 #ifndef DEVICEMOVEMENTCONTROLLER_H
3 #define DEVICEMOVEMENTCONTROLLER_H
4
5 #pragma once
6
7 #include <memory>
8
9 #include "AbstractMovementController.h"
10
11 #include "car/system/movement/devices/Servo.h"
12 #include "car/system/movement/devices/RearWheel.h"
13
14 using namespace car::system::movement::devices;
15
16 namespace car::system::movement::controller
17 {
18     static constexpr int Motor_A = 17;
19     static constexpr int Motor_B = 27;
20     static constexpr int PWM_A = 4;
21     static constexpr int PWM_B = 5;
22
23     static constexpr int MIN_PULSE_WIDTH = 900;
24     static constexpr int MAX_PULSE_WIDTH = 2100;
25     static constexpr int FREQUENCY = 50;
26
27     static constexpr int BUS_NUMBER = 1;
28
29     class DeviceMovementController : public AbstractMovementController
30     {
31     public:
32         [[nodiscard]] DeviceMovementController();
33
34         void initialize() final override;
35
36         void stop() final override;
37
38         void terminate() final override;
39
40         void setRearWheelsSpeed(const int speed) final override;
41
42         void setRearLeftWheelSpeed(const int speed) final override;
43
44         void setRearRightWheelSpeed(const int speed) final override;
45
46         void setFrontWheelsAngle(const float angle) final override;
47
48         void setCameraServo1Angle(const float angle) final override;
49
50         void setCameraServo2Angle(const float angle) final override;
51
52         void setRearWheelsDirectionToForward() final override;
53
54         void setRearLeftWheelDirectionToForward() final override;
55
56         void setRearRightWheelDirectionToForward() final override;
57
58         void setRearWheelsDirectionToBackward() final override;
59
60         void setRearLeftWheelDirectionToBackward() final override;
61
62         void setRearRightWheelDirectionToBackward() final override;
63
64     private:
65         std::shared_ptr<PCA9685> pwm;
66
67         std::unique_ptr<Servo> front_wheels_;
68         std::unique_ptr<Servo> camera_servo_1_;
69         std::unique_ptr<Servo> camera_servo_2_;
70
71         std::unique_ptr<RearWheel> rear_left_wheel_;
72         std::unique_ptr<RearWheel> rear_right_wheel_;
73     };
74 } // namespace car::system::movement::controller
75
```

```
76 #endif
77 #endif // __linux__
```

7.33 include/car/system/movement/controller/DummyMovementController.h File Reference

```
#include "AbstractMovementController.h"
```

Classes

- class car::system::movement::controller::DummyMovementController

Namespaces

- namespace car
- namespace car::system
- namespace car::system::movement
- namespace car::system::movement::controller

7.34 DummyMovementController.h

Go to the documentation of this file.

```
1 #ifndef DUMMYWHEELCONTROLLER_H
2 #define DUMMYWHEELCONTROLLER_H
3
4 #pragma once
5
6 #include "AbstractMovementController.h"
7
8 namespace car::system::movement::controller
9 {
10     class DummyMovementController : public AbstractMovementController
11     {
12     public:
13         void initialize() final override {};
14
15         void stop() final override;
16
17         void terminate() final override {};
18
19         void setRearWheelsSpeed(const int speed) final override;
20
21         void setRearLeftWheelSpeed(const int speed) final override;
22
23         void setRearRightWheelSpeed(const int speed) final override;
24
25         void setFrontWheelsAngle(const float angle) final override;
26
27         void setCameraServo1Angle(const float angle) final override;
28
29         void setCameraServo2Angle(const float angle) final override;
30
31         void setRearWheelsDirectionToForward() final override;
32
33         void setRearLeftWheelDirectionToForward() final override;
34
35         void setRearRightWheelDirectionToForward() final override;
36
37         void setRearWheelsDirectionToBackward() final override;
38
39         void setRearLeftWheelDirectionToBackward() final override;
40
41         void setRearRightWheelDirectionToBackward() final override;
42
43     private:
44     };
45 } // namespace car::system::movement::controller
46
47 #endif
```

7.35 include/car/system/movement/devices/RearWheel.h File Reference

7.36 RearWheel.h

Go to the documentation of this file.

```
1 #ifdef __linux__
2 #ifndef REARWHEEL_H
3 #define REARWHEEL_H
4
5 #include <memory>
6
7 #include <PCA9685.h>
8 #include <TB6612.h>
9
10 // Made with the help of ChatGPT
11
12 namespace car::system::movement::devices
13 {
14     class RearWheel
15     {
16     public:
17         RearWheel(std::shared_ptr<PCA9685> pwm, std::unique_ptr<TB6612> motor);
18
19         void forward();
20
21         void backward();
22
23         void stop();
24
25         int getSpeed() const;
26
27         void setSpeed(const int speed);
28
29         void ready();
30
31     private:
32         std::shared_ptr<PCA9685> pwm_;
33         std::unique_ptr<TB6612> motor_;
34
35         int speed_;
36     };
37 } // namespace car::system::movement::wheels
38
39 #endif
40 #endif
```

7.37 include/car/system/movement/devices/Servo.h File Reference

7.38 Servo.h

Go to the documentation of this file.

```
1 #ifdef __linux__
2 #ifndef SERVO_H
3 #define SERVO_H
4
5 #include <algorithm>
6 #include <memory>
7
8 #include <PCA9685.h>
9
10 namespace car::system::movement::devices
11 {
12     class Servo
13     {
14     private:
19         static int map(int x, int in_min, int in_max, int out_min, int out_max)
20         {
21             return ((x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min);
22         }
23
24         static constexpr int MIN_PULSE_WIDTH = 900;
25         static constexpr int MAX_PULSE_WIDTH = 2100;
26         static constexpr int FREQUENCY = 50;
```

```
27
28      public:
29          Servo(std::shared_ptr<PCA9685> pwm, int channel);
30
31          // Some of the code was from: https://github.com/chaoticmachinery/pca9685
32          int getAnalogAngle() const;
33
34          int getAngle() const;
35
36          // Some of the code was from: https://github.com/chaoticmachinery/pca9685
37          void setAngle(const int angle);
38
39          void reset();
40
41      private:
42          const std::shared_ptr<PCA9685> pwm_;
43          const int channel_;
44
45          int angle_;
46      };
47  } // namespace car::system::movement::wheels
48
49  #endif
50  #endif // __linux__
```

7.39 include/car/system/movement/MovementSystem.h File Reference

```
#include <memory>
#include "car/system/movement/controller/AbstractMovementController.h"
```

Classes

- class car::system::movement::MovementSystem

Namespaces

- namespace car
- namespace car::system
- namespace car::system::movement

7.40 MovementSystem.h

Go to the documentation of this file.

```
1  #ifndef MOVEMENTSYSTEM_H
2  #define MOVEMENTSYSTEM_H
3
4  #pragma once
5
6  #include <memory>
7
8  #include "car/system/movement/controller/AbstractMovementController.h"
9
10 using namespace car::system::movement::controller;
11
12 namespace car::system::movement
13 {
14      class MovementSystem
15      {
16      public:
17          MovementSystem(std::unique_ptr<AbstractMovementController> movement_controller) :
        movement_controller(std::move(movement_controller)){};
18
19          void initialize()
20          {
```

```
21              this->movement_controller->initialize();
22          }
23
24          void start()
25          {
26          }
27
28          void stop()
29          {
30              this->movement_controller->stop();
31          }
32
33          void terminate()
34          {
35              this->movement_controller->terminate();
36          }
37
38  #pragma region Wheels
39          void setRearWheelsSpeed(const int speed) const
40          {
41              this->movement_controller->setRearWheelsSpeed(speed);
42          }
43
44          void setRearLeftWheelSpeed(const int speed) const
45          {
46              this->movement_controller->setRearLeftWheelSpeed(speed);
47          }
48
49          void setRearRightWheelSpeed(const int speed) const
50          {
51              this->movement_controller->setRearRightWheelSpeed(speed);
52          }
53
54          void setFrontWheelsAngle(const float angle) const
55          {
56              this->movement_controller->setFrontWheelsAngle(angle);
57          }
58
59          void setCameraServo1Angle(const float angle) const
60          {
61              this->movement_controller->setCameraServo1Angle(angle);
62          }
63
64          void setCameraServo2Angle(const float angle) const
65          {
66              this->movement_controller->setCameraServo2Angle(angle);
67          }
68
69          void setRearWheelsDirectionToForward() const
70          {
71              this->movement_controller->setRearWheelsDirectionToForward();
72          }
73
74          void setRearLeftWheelDirectionToForward() const
75          {
76              this->movement_controller->setRearLeftWheelDirectionToForward();
77          }
78
79          void setRearRightWheelDirectionToForward() const
80          {
81              this->movement_controller->setRearRightWheelDirectionToForward();
82          }
83
84          void setRearWheelsDirectionToBackward() const
85          {
86              this->movement_controller->setRearWheelsDirectionToBackward();
87          }
88
89          void setRearLeftWheelDirectionToBackward() const
90          {
91              this->movement_controller->setRearLeftWheelDirectionToBackward();
92          }
93
94          void setRearRightWheelDirectionToBackward() const
95          {
96              this->movement_controller->setRearRightWheelDirectionToBackward();
97          }
98  #pragma endregion
99
100         ~MovementSystem(){};
101
102     private:
103         std::unique_ptr<AbstractMovementController> movement_controller;
104     };
105 };
106
107 #endif
```

7.41 src/car/system/CarSystem.cpp File Reference

```
#include "car/system/CarSystem.h"
#include <memory>
#include <rapidjson/rapidjson.h>
#include <rapidjson/document.h>
#include <rapidjson/stringbuffer.h>
#include <rapidjson/writer.h>
#include <tobiaslocker_base64/base64.hpp>
#include "car/configuration/Configuration.h"
#include "car/system/device/DeviceManager.h"
#include "car/system/device/lidar/LidarDevice.h"
#include "car/system/device/CameraDevice.h"
#include "car/system/messaging/MessagingSystem.h"
#include "car/system/movement/MovementSystem.h"
#include "car/plugin/PluginManager.h"
```

Namespaces

- namespace car
- namespace car::system

7.42 src/car/system/device/CameraDevice.cpp File Reference

```
#include "car/system/device/CameraDevice.h"
```

Namespaces

- namespace car
- namespace car::system
- namespace car::system::device

7.43 src/car/system/device/DeviceManager.cpp File Reference

```
#include "car/system/device/DeviceManager.h"
#include "car/system/CarSystem.h"
```

Namespaces

- namespace car
- namespace car::system
- namespace car::system::device

7.44 src/car/system/messaging/MessagingSystem.cpp File Reference

```
#include "car/system/messaging/MessagingSystem.h"
#include <functional>
#include <memory>
#include <ixwebsocket/IXNetSystem.h>
#include <ixwebsocket/IXWebSocket.h>
#include <nod/nod.hpp>
#include <spdlog/spdlog.h>
#include <rapidjson/rapidjson.h>
#include <rapidjson/document.h>
#include <fmt/format.h>
#include "car/configuration/Configuration.h"
```

Namespaces

- namespace car
- namespace car::system
- namespace car::system::messaging

7.45 src/car/system/movement/controller/DeviceMovement↩ Controller.cpp File Reference

7.46 src/car/system/movement/controller/DummyMovement↩ Controller.cpp File Reference

```
#include "car/system/movement/controller/DummyMovementController.h"
#include <spdlog/spdlog.h>
```

Namespaces

- namespace car
- namespace car::system
- namespace car::system::movement
- namespace car::system::movement::controller

7.47 src/car/system/movement/devices/RearWheel.cpp File Reference

7.48 src/car/system/movement/devices/Servo.cpp File Reference

7.49 tests/pca9685/test_front_wheels.cpp File Reference

```
#include "PCA9685.h"
#include <iostream>
#include <algorithm>
#include <thread>
```

Functions

- int setAngle (int &angle, PCA9685 pwm, int channel)
- int map (int x, int in_min, int in_max, int out_min, int out_max)
- int setAngleToAnalog (int angle)
- int main ()

Variables

- int offset = 0

7.49.1 Function Documentation

7.49.1.1 main()

```
int main ( )
```

7.49.1.2 map()

```
int map (
            int x,
            int in_min,
            int in_max,
            int out_min,
            int out_max )
```

Following method clamps the x to in_min and in_max. Afterwards, it puts the result of that into the range of out_min and out_max

7.49.1.3 setAngle()

```
int setAngle (
            int & angle,
            PCA9685 pwm,
            int channel )
```

7.49.1.4 setAngleToAnalog()

```
int setAngleToAnalog (
            int angle )
```

7.49.2 Variable Documentation

7.49.2.1 offset

```
int offset = 0
```

7.50 tests/tb6612/test_rear_wheels.cpp File Reference

```
#include <pigpio.h>
#include <iostream>
#include <memory>
#include <thread>
#include <chrono>
#include <algorithm>
#include "PCA9685.h"
#include "TB6612.h"
```

Classes

- class BackWheels

Functions

- void test ()
- int main ()

7.50.1 Function Documentation

7.50.1.1 main()

```
int main ( )
```

7.50.1.2 test()

```
void test ( )
```

Index