Behaviour Tree PiCar-V

Generated by Doxygen 1.9.5

Chapter 1

Sunfounder PiCar Daemon

1.1 Installation:

Run the follow commands to install the Daemon:

```
sudo chmod +x ./install_script.sh # Grant permission to install_script to run as exe
systemctl stop rpi_daemon # Stop any previous rpi_daemon
sudo ./install_script.sh install # Install the Daemon
sudo systemctl daemon-reload # Reload the Daemon if there was any previous
sudo systemctl start rpi_daemon # Start the new Daemon
```

Or use the single command:

```
sudo chmod +x ./install_script.sh && systemctl stop rpi_daemon && sudo ./install_script.sh install && sudo
       systemctl daemon-reload && sudo systemctl start rpi_daemon
```

1.2 View Logs

`systemctl status rpi_daemon.service` `journalctl -u rpi_daemon`

Chapter 2

Raspberry Pi Daemon

This will only work for Linux.

2.1 Install Raspberry Pi Daemon

```
xmake install --admin rpi_daemon
```

2.2 Enable / Disable Daemon to run on startup

```
sudo systemctl [enable|disable] rpi_daemon
```

2.3 Start / Stop / Restart Daemon

```
sudo systemctl [start/stop/restart] rpi_daemon
```

2.4 View Logs

```
systemctl status rpi_daemon.service journalctl -u rpi_daemon
```

2.5 Clear Past Logs

```
sudo journalctl -m --vacuum-time=1s
```

2.6 Reload Daemon

```
sudo systemctl reload rpi_daemon
```

2.7 Change Config

```
sudo nano /etc/rpi_daemon/rpi_daemon.conf
```

2.8 Uninstall Raspberry Pi Daemon

```
xmake uninstall --admin rpi_daemon
```

2.9 Speedrun

```
sudo systemctl stop rpi_daemon && xmake && xmake install --admin rpi_daemon
&& sudo systemctl daemon-reload && sudo systemctl start rpi_daemon
```

Chapter 3

Raspberry Pi

This sub-project contains three other sub-projects:

- common - Main functionality is stored here
- daemon - Daemon Application for seamless startup and termination
- tui - Terminal User Interface

3.1 Main Technologies

- C++17
- `ixwebsocket`
- `ftxui`
- `rapidjson`

3.2 Setting up Raspberry Pi

If you want to setup the raspberry pi, you can go to `SETUP.md`

Chapter 4

SETUP

Set up the following first:

- Wifi - Use the Mobile Hotspot feature of your device
- SSH - Enable
- I2C - Enable

See if Wifi does not connect: Raspberrypi Stackexchange

Now connect to the Raspberry Pi with SSH

Run the following commands to get the raspberry pi running:

```
sudo apt-get install build-essential
curl -fsSL https://xmake.io/shget.text | bash
source ~/.xmake/profile
```

Run the following commands before cross compilation:

```
sudo apt update
sudo apt upgrade
sudo apt dist-upgrade
```

or `sudo apt update && sudo apt upgrade && sudo apt dist-upgrade`

Afterwards run `chmod +x ./rpi_tui` on the new executable

Use CyberDuck to copy the raspberry_pi folder.

Chapter 5

Raspberry Pi

This sub-project handles all the logic of the Sunfounder Raspberry Pi.

The system is broken down to 3 sub systems:

- lidar - Handles the lidar scanner of the Raspberry Pi
- messaging - Handles the websocket of the Raspberry Pi
- movement - Handles the movement of the Sunfounder Raspberry Pi

The system is then displayed by the `CarConsole`.

5.1 Installation

Run the following command to build the sub-project:

```
xmake build raspberry_pi
```

To connect the Raspberry Pi to the backend, you would need the `IPv4 Address` of your Computer/Server and change the host to the value in `IPv4 Address`.

If this doesn't work, you may need to restart your Computer/Server.

5.2 Main Technologies

- C++17
- `ixwebsocket`
- `ftxui`
- `nlohmann_json`

5.3 Setting up Raspberry Pi

If you want to setup the raspberry pi, you can go to `SETUP.md`

Chapter 6

SETUP

Set up the following first:

- Wifi - Use the Mobile Hotspot feature of your device
- SSH - Enable
- I2C - Enable

See if Wifi does not connect: Raspberrypi Stackexchange

Now connect to the Raspberry Pi with SSH

Run the following commands to get the raspberry pi running:

```
sudo apt-get install build-essential
curl -fsSL https://xmake.io/shget.text | bash
source ~/.xmake/profile
```

Run the following commands before cross compilation:

```
sudo apt update
sudo apt upgrade
sudo apt dist-upgrade
```

or `sudo apt update && sudo apt upgrade && sudo apt dist-upgrade`

Afterwards run `chmod +x ./raspberry_pi` on the new executable

Use CyberDuck to copy the raspberry_pi folder.

Chapter 7

Namespace Index

7.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 8

Hierarchical Index

8.1 Class Hierarchy

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

Chapter 9

Class Index

9.1 Class List

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

Chapter 10

File Index

10.1 File List

Here is a list of all files with brief descriptions:

Chapter 11

Namespace Documentation

11.1 behaviour_tree Namespace Reference

Classes

- class BehaviourTreeHandler
- class CarContext

11.2 car Namespace Reference

Namespaces

- namespace configuration
- namespace display
- namespace plugin
- namespace system

11.3 car::configuration Namespace Reference

Classes

- struct Configuration
- class JsonConfiguration

11.4 car::display Namespace Reference

Namespaces

- namespace console

11.5 car::display::console Namespace Reference

Namespaces

- namespace component
- namespace screen

Classes

- class CarConsole

11.6 car::display::console::component Namespace Reference

Namespaces

- namespace debug
- namespace main
- namespace settings

11.7 car::display::console::component::debug Namespace Reference

Classes

- class DebugEnabler
- class DebugLidarCheckbox
- class DebugMessagingTextbox
- class DebugMovementRenderer

11.8 car::display::console::component::main Namespace Reference

Classes

- class ConnectButton
- class MainErrorModal
- class MainExitModal

11.9 car::display::console::component::settings Namespace Reference

Classes

- class SettingsEditConfig

11.10 car::display::console::screen Namespace Reference

Classes

- class LoggingScreen
- class MainScreen
- class SettingsScreen

11.11 car::plugin Namespace Reference

Classes

- class Plugin
- class PluginManager

11.12 car::system Namespace Reference

Namespaces

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

Classes

- class CarSystem

11.13 car::system::device Namespace Reference

Namespaces

- namespace lidar

Classes

- class CameraDevice
- class DeviceManager

11.14 car::system::device::lidar Namespace Reference

Classes

- class LidarDevice
- class LidarDummy
- class LidarScanner

11.15 car::system::logging Namespace Reference

Classes

- class VectorSink

Typedefs

- using vector_sink_mt = VectorSink< std::mutex >

11.15.1 Typedef Documentation

11.15.1.1 vector_sink_mt

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

11.16 car::system::messaging Namespace Reference

Classes

- class MessagingSystem

11.17 car::system::movement Namespace Reference

Namespaces

- namespace controller

Classes

- class MovementSystem

11.18 car::system::movement::controller Namespace Reference

Classes

- class AbstractMovementController
- class DummyMovementController

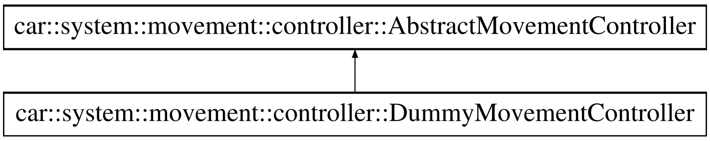
Chapter 12

Class Documentation

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

```
#include <AbstractMovementController.h>
```

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


car::system::movement::controller::AbstractMovementController
car::system::movement::controller::DummyMovementController

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

12.1.1 Member Function Documentation

12.1.1.1 initialize()

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

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

12.1.1.2 setCameraServo1Angle()

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

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

12.1.1.3 setCameraServo2Angle()

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

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

12.1.1.4 setFrontWheelsAngle()

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

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

12.1.1.5 setRearLeftWheelDirectionToBackward()

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

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

12.1.1.6 setRearLeftWheelDirectionToForward()

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

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

12.1.1.7 setRearLeftWheelSpeed()

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

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

12.1.1.8 setRearRightWheelDirectionToBackward()

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

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

12.1.1.9 setRearRightWheelDirectionToForward()

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

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

12.1.1.10 setRearRightWheelSpeed()

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

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

12.1.1.11 setRearWheelsDirectionToBackward()

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

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

12.1.1.12 setRearWheelsDirectionToForward()

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

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

12.1.1.13 setRearWheelsSpeed()

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

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

12.1.1.14 stop()

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

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

12.1.1.15 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:

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

12.2 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

12.2.1 Constructor & Destructor Documentation

12.2.1.1 BackWheels()

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

12.2.2 Member Function Documentation

12.2.2.1 backward()

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

12.2.2.2 calibration()

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

12.2.2.3 caliLeft()

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

12.2.2.4 caliOK()

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

12.2.2.5 caliRight()

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

12.2.2.6 forward()

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

12.2.2.7 getSpeed()

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

12.2.2.8 ready()

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

12.2.2.9 setSpeed()

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

12.2.2.10 stop()

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

12.2.3 Member Data Documentation

12.2.3.1 cali_forward_A

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

12.2.3.2 cali_forward_B

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

12.2.3.3 forward_A

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

12.2.3.4 forward_B

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

12.2.3.5 left_wheel

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

12.2.3.6 pca9685

```
PCA9685 BackWheels::pca9685
```

12.2.3.7 right_wheel

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

12.2.3.8 speed

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

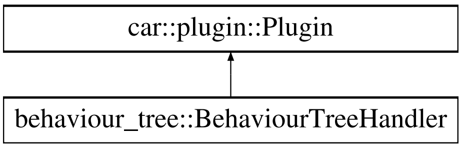
The documentation for this class was generated from the following file:

- common/tests/tb6612/test_rear_wheels.cpp

12.3 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

12.3.1 Member Function Documentation

12.3.1.1 _setBehaviourTree()

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

12.3.1.2 getName()

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

Implements car::plugin::Plugin.

12.3.1.3 handleCommand()

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

12.3.1.4 initialize()

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

Implements car::plugin::Plugin.

12.3.1.5 setBehaviourTree()

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

12.3.1.6 startBehaviourTree()

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

12.3.1.7 stop()

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

Implements car::plugin::Plugin.

12.3.1.8 stopBehaviourTree()

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

12.3.1.9 update()

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

Implements car::plugin::Plugin.

12.3.2 Member Data Documentation

12.3.2.1 behaviour_tree

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

12.3.2.2 car_system

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

12.3.2.3 context

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

12.3.2.4 last_connected

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

12.3.2.5 tick_count

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

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

- common/include/behaviour_tree/BehaviourTreeHandler.hpp

12.4 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

12.4.1 Constructor & Destructor Documentation

12.4.1.1 CameraDevice() [1/3]

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

12.4.1.2 CameraDevice() [2/3]

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

12.4.1.3 CameraDevice() [3/3]

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

12.4.1.4 ∼CameraDevice()

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

12.4.2 Member Function Documentation

12.4.2.1 create()

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

12.4.2.2 disconnect()

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

12.4.2.3 getFrameBuffer()

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

12.4.2.4 operator=() [1/2]

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

12.4.2.5 operator=() [2/2]

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

12.4.2.6 start()

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

12.4.2.7 stop()

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

12.4.2.8 terminate()

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

12.4.2.9 update()

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

12.4.3 Friends And Related Function Documentation

12.4.3.1 DeviceManager

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

12.4.4 Member Data Documentation

12.4.4.1 camera_

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

12.4.4.2 camera_mutex_

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

12.4.4.3 configuration

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

12.4.4.4 connected_

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

12.4.4.5 frame_buffer_

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

12.4.4.6 last

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

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

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

12.5 car::display::console::CarConsole Class Reference

```
#include <CarConsole.h>
```

Public Member Functions

- CarConsole (std::shared_ptr< CarSystem > car_system, std::shared_ptr< JsonConfiguration > json_configuration, std::shared_ptr< logging::vector_sink_mt > vector_sink)
- void initialize ()
- void run ()
- void terminate ()

Private Attributes

- std::shared_ptr< CarSystem > car_system
- std::shared_ptr< JsonConfiguration > json_configuration
- std::shared_ptr< logging::vector_sink_mt > vector_sink

12.5.1 Constructor & Destructor Documentation

12.5.1.1 CarConsole()

```
car::display::console::CarConsole::CarConsole (
            std::shared_ptr< CarSystem > car_system,
            std::shared_ptr< JsonConfiguration > json_configuration,
            std::shared_ptr< logging::vector_sink_mt > vector_sink )
```

12.5.2 Member Function Documentation

12.5.2.1 initialize()

```
void car::display::console::CarConsole::initialize ( )
```

12.5.2.2 run()

```
void car::display::console::CarConsole::run ( )
```

12.5.2.3 terminate()

```
void car::display::console::CarConsole::terminate ( )
```

12.5.3 Member Data Documentation

12.5.3.1 car_system

```
std::shared_ptr<CarSystem> car::display::console::CarConsole::car_system  [private]
```

12.5.3.2 json_configuration

```
std::shared_ptr<JsonConfiguration> car::display::console::CarConsole::json_configuration
[private]
```

12.5.3.3 vector_sink

```
std::shared_ptr<logging::vector_sink_mt> car::display::console::CarConsole::vector_sink  [private]
```

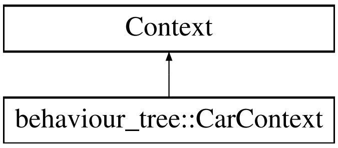
The documentation for this class was generated from the following files:

- tui/src/car/display/console/CarConsole.h
- tui/src/car/display/console/CarConsole.cpp

12.6 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

12.6.1 Constructor & Destructor Documentation

12.6.1.1 CarContext()

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

12.6.2 Member Function Documentation

12.6.2.1 _()

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

12.6.2.2 getCarSystem()

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

12.6.3 Member Data Documentation

12.6.3.1 car_system

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

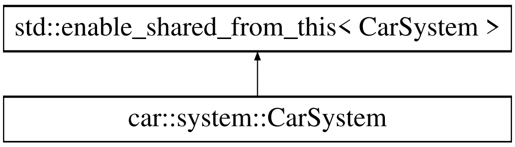
The documentation for this class was generated from the following file:

- common/include/behaviour_tree/CarContext.hpp

12.7 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

12.7.1 Constructor & Destructor Documentation

12.7.1.1 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 )
```

12.7.2 Member Function Documentation

12.7.2.1 disconnect()

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

12.7.2.2 getConfiguration()

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

12.7.2.3 getDeviceManager()

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

12.7.2.4 getMessagingSystem()

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

12.7.2.5 getMovementSystem()

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

12.7.2.6 getPlugin()

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

12.7.2.7 initialize()

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

12.7.2.8 reload()

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

12.7.2.9 sendData()

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

12.7.2.10 setConfiguration()

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

12.7.2.11 start()

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

12.7.2.12 stop()

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

12.7.2.13 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.

12.7.2.14 tryConnect()

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

12.7.2.15 update()

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

12.7.3 Member Data Documentation

12.7.3.1 configuration_

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

12.7.3.2 device_manager_

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

12.7.3.3 initialized

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

12.7.3.4 messaging_system_

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

12.7.3.5 movement_system_

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

12.7.3.6 plugin_manager_

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

12.7.3.7 started

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

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

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

12.8 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

12.8.1 Member Function Documentation

12.8.1.1 getCameraFpsInterval()

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

12.8.1.2 setCameraFps()

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

12.8.2 Member Data Documentation

12.8.2.1 behaviour_tree_update_ms_interval

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

12.8.2.2 camera_fps

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

12.8.2.3 camera_fps_interval

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

12.8.2.4 camera_index

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

12.8.2.5 host

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

12.8.2.6 lidar_port

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

12.8.2.7 use_camera

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

12.8.2.8 use_lidar

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

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

- common/include/car/configuration/Configuration.h

12.9 car::display::console::component::main::ConnectButton Class Reference

Public Member Functions

- ConnectButton (std::shared_ptr< CarSystem > car_system, Box &box)
- Component element ()

Public Attributes

- std::function< void(std::string)> on_connect_failure = [ ](std::string _) {}

Private Attributes

- std::shared_ptr< CarSystem > car_system
- bool main_debounce = false
- bool button_pressed = false
- std::string main_button_text = "Start Car Application"
- Component main_button

12.9.1 Constructor & Destructor Documentation

12.9.1.1 ConnectButton()

```
car::display::console::component::main::ConnectButton::ConnectButton (
            std::shared_ptr< CarSystem > car_system,
            Box & box )  [inline]
```

12.9.2 Member Function Documentation

12.9.2.1 element()

```
Component car::display::console::component::main::ConnectButton::element ( )  [inline]
```

12.9.3 Member Data Documentation

12.9.3.1 button_pressed

```
bool car::display::console::component::main::ConnectButton::button_pressed = false  [private]
```

12.9.3.2 car_system

```
std::shared_ptr<CarSystem> car::display::console::component::main::ConnectButton::car_system
[private]
```

12.9.3.3 main_button

```
Component car::display::console::component::main::ConnectButton::main_button  [private]
```

12.9.3.4 main_button_text

```
std::string car::display::console::component::main::ConnectButton::main_button_text = "Start
Car Application"  [private]
```

12.9.3.5 main_debounce

```
bool car::display::console::component::main::ConnectButton::main_debounce = false  [private]
```

12.9.3.6 on_connect_failure

```
std::function<void(std::string)> car::display::console::component::main::ConnectButton::on_↩
connect_failure = [](std::string _) {}
```

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

- tui/src/car/display/console/component/main/ConnectButton.cxx

12.10 car::display::console::component::debug::DebugEnabler Class Reference

Public Member Functions

- Component getCheckbox ()
- ComponentDecorator getWarningModal ()
- const bool & isEnabled () const

Private Attributes

- bool debounce = false
- bool enabled = false
- bool checkbox_value = false
- bool display_warn_debug_modal = false
- std::string status = DEBUG_MODE_DISABLED_MESSAGE
- Component component

Static Private Attributes

- static constexpr auto DEBUG_ENABLE_WARNING_MESSAGE = "Enabling debug mode temporarily disables connecting to online. Are you sure you want to do this?"
- static constexpr auto DEBUG_MODE_ENABLED_MESSAGE = "Debug Status: Enabled"
- static constexpr auto DEBUG_MODE_DISABLED_MESSAGE = "Debug Status: Disabled"
- static constexpr auto DEBUG_MODE_WAIT_MESSAGE = "Debug Status: Waiting for user input..."

12.10.1 Member Function Documentation

12.10.1.1 getCheckbox()

```
Component car::display::console::component::debug::DebugEnabler::getCheckbox ( )  [inline]
```

12.10.1.2 getWarningModal()

```
ComponentDecorator car::display::console::component::debug::DebugEnabler::getWarningModal ( )
[inline]
```

12.10.1.3 isEnabled()

```
const bool & car::display::console::component::debug::DebugEnabler::isEnabled ( ) const  [inline]
```

12.10.2 Member Data Documentation

12.10.2.1 checkbox_value

```
bool car::display::console::component::debug::DebugEnabler::checkbox_value = false  [private]
```

12.10.2.2 component

```
Component car::display::console::component::debug::DebugEnabler::component  [private]
```

12.10.2.3 debounce

```
bool car::display::console::component::debug::DebugEnabler::debounce = false  [private]
```

12.10.2.4 DEBUG_ENABLE_WARNING_MESSAGE

```
constexpr auto car::display::console::component::debug::DebugEnabler::DEBUG_ENABLE_WARNING←↩
_MESSAGE = "Enabling debug mode temporarily disables connecting to online.  Are you sure you
want to do this?"  [static], [constexpr], [private]
```

12.10.2.5 DEBUG_MODE_DISABLED_MESSAGE

```
constexpr auto car::display::console::component::debug::DebugEnabler::DEBUG_MODE_DISABLED_←↩
MESSAGE = "Debug Status:  Disabled"  [static], [constexpr], [private]
```

12.10.2.6 DEBUG_MODE_ENABLED_MESSAGE

```
constexpr auto car::display::console::component::debug::DebugEnabler::DEBUG_MODE_ENABLED_←↩
MESSAGE = "Debug Status:  Enabled"  [static], [constexpr], [private]
```

12.10.2.7 DEBUG_MODE_WAIT_MESSAGE

```
constexpr auto car::display::console::component::debug::DebugEnabler::DEBUG_MODE_WAIT_MESSAGE
= "Debug Status:  Waiting for user input..."  [static], [constexpr], [private]
```

12.10.2.8 display_warn_debug_modal

```
bool car::display::console::component::debug::DebugEnabler::display_warn_debug_modal = false
[private]
```

12.10.2.9 enabled

```
bool car::display::console::component::debug::DebugEnabler::enabled = false  [private]
```

12.10.2.10 status

```
std::string car::display::console::component::debug::DebugEnabler::status = DEBUG_MODE_DISABLED_MESSAGE
[private]
```

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

- tui/src/car/display/console/component/debug/DebugEnabler.cxx

12.11 car::display::console::component::debug::DebugLidarCheckbox Class Reference

Public Member Functions

- DebugLidarCheckbox ()
- Component element ()
- nod::signal< void(bool)> & getLidarMotorSignal ()

Private Attributes

- nod::signal< void(bool)> lidar_motor_signal
- std::string lidar_motor_status = LIDAR_MOTOR_DISABLED_MESSAGE
- bool lidar_motor_loading_debounce = false
- bool lidar_motor_enabled = false
- Component lidar_motor_checkbox_component

Static Private Attributes

- static constexpr auto LIDAR_MOTOR_ENABLED_MESSAGE = "Lidar Motor Status: Enabled"
- static constexpr auto LIDAR_MOTOR_DISABLED_MESSAGE = "Lidar Motor Status: Disconnected"

12.11.1 Constructor & Destructor Documentation

12.11.1.1 DebugLidarCheckbox()

```
car::display::console::component::debug::DebugLidarCheckbox::DebugLidarCheckbox ( )  [inline]
```

12.11.2 Member Function Documentation

12.11.2.1 element()

```
Component car::display::console::component::debug::DebugLidarCheckbox::element ( )  [inline]
```

12.11.2.2 getLidarMotorSignal()

```
nod::signal< void(bool)> & car::display::console::component::debug::DebugLidarCheckbox::get←↩
LidarMotorSignal ( )  [inline]
```

12.11.3 Member Data Documentation

12.11.3.1 lidar_motor_checkbox_component

```
Component car::display::console::component::debug::DebugLidarCheckbox::lidar_motor_checkbox_←↩
component  [private]
```

12.11.3.2 LIDAR_MOTOR_DISABLED_MESSAGE

```
constexpr auto car::display::console::component::debug::DebugLidarCheckbox::LIDAR_MOTOR_←↩
DISABLED_MESSAGE = "Lidar Motor Status:  Disconnected"  [static], [constexpr], [private]
```

12.11.3.3 lidar_motor_enabled

```
bool car::display::console::component::debug::DebugLidarCheckbox::lidar_motor_enabled = false
[private]
```

12.11.3.4 LIDAR_MOTOR_ENABLED_MESSAGE

```
constexpr auto car::display::console::component::debug::DebugLidarCheckbox::LIDAR_MOTOR_←↩
ENABLED_MESSAGE = "Lidar Motor Status:  Enabled"  [static], [constexpr], [private]
```

12.11.3.5 lidar_motor_loading_debounce

```
bool car::display::console::component::debug::DebugLidarCheckbox::lidar_motor_loading_debounce
= false [private]
```

12.11.3.6 lidar_motor_signal

```
nod::signal<void(bool)> car::display::console::component::debug::DebugLidarCheckbox::lidar_←
motor_signal [private]
```

12.11.3.7 lidar_motor_status

```
std::string car::display::console::component::debug::DebugLidarCheckbox::lidar_motor_status =
LIDAR_MOTOR_DISABLED_MESSAGE [private]
```

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

- tui/src/car/display/console/component/debug/DebugLidarCheckbox.cxx

12.12 car::display::console::component::debug::DebugMessaging← Textbox Class Reference

Public Member Functions

- DebugMessagingTextbox (nod::signal< void(const std::string)> &message_signal)
- ftxui::Component element ()

Private Attributes

- std::string message
- Component messaging_title
- Component messaging_textbox
- Component messaging_container
- nod::signal< void(const std::string)> & message_signal

12.12.1 Constructor & Destructor Documentation

12.12.1.1 DebugMessagingTextbox()

```
car::display::console::component::debug::DebugMessagingTextbox::DebugMessagingTextbox (
            nod::signal< void(const std::string)> & message_signal )  [inline]
```

12.12.2 Member Function Documentation

12.12.2.1 element()

```
ftxui::Component car::display::console::component::debug::DebugMessagingTextbox::element ( )
[inline]
```

12.12.3 Member Data Documentation

12.12.3.1 message

```
std::string car::display::console::component::debug::DebugMessagingTextbox::message  [private]
```

12.12.3.2 message_signal

```
nod::signal<void(const std::string)>& car::display::console::component::debug::DebugMessaging←↩
Textbox::message_signal  [private]
```

12.12.3.3 messaging_container

```
Component car::display::console::component::debug::DebugMessagingTextbox::messaging_container
[private]
```

12.12.3.4 messaging_textbox

```
Component car::display::console::component::debug::DebugMessagingTextbox::messaging_textbox
[private]
```

12.12.3.5 messaging_title

```
Component car::display::console::component::debug::DebugMessagingTextbox::messaging_title
[private]
```

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

- tui/src/car/display/console/component/debug/DebugMessagingTextbox.cxx

12.13 car::display::console::component::debug::DebugMovement↩ Renderer Class Reference

Public Member Functions

- DebugMovementRenderer ()
- ftxui::Component element ()
- bool updateFrontWheels ()
- bool updateCameraServo1 ()
- bool updateCameraServo2 ()
- bool updateRearWheels ()
- nod::signal< void(bool)> & getRearWheelDirectionSignal ()
- const int getFrontWheelsAngleSliderValue () const
- const int getCameraServo1AngleSliderValue () const
- const int getCameraServo2AngleSliderValue () const
- const int getRearLeftWheelSpeedSliderValue () const
- const int getRearRightWheelSpeedSliderValue () const

Private Attributes

- nod::signal< void(bool)> rear_wheel_direction_signal
- int previous_rear_wheels_speed_slider_value = DEFAULT_REAR_WHEEL_SPEED
- int rear_wheels_speed_slider_value = DEFAULT_REAR_WHEEL_SPEED
- int previous_rear_left_wheel_speed_slider_value = DEFAULT_REAR_WHEEL_SPEED
- int rear_left_wheel_speed_slider_value = DEFAULT_REAR_WHEEL_SPEED
- int previous_rear_right_wheel_speed_slider_value = DEFAULT_REAR_WHEEL_SPEED
- int rear_right_wheel_speed_slider_value = DEFAULT_REAR_WHEEL_SPEED
- int previous_front_wheels_angle_slider_value = DEFAULT_FRONT_WHEEL_ANGLE
- int front_wheels_angle_slider_value = DEFAULT_FRONT_WHEEL_ANGLE
- int previous_camera_servo_1_angle_slider_angle = DEFAULT_FRONT_WHEEL_ANGLE
- int camera_servo_1_angle_slider_angle = DEFAULT_FRONT_WHEEL_ANGLE
- int previous_camera_servo_2_angle_slider_angle = DEFAULT_FRONT_WHEEL_ANGLE
- int camera_servo_2_angle_slider_angle = DEFAULT_FRONT_WHEEL_ANGLE
- bool rear_wheel_direction_debounce = false
- std::string rear_wheel_direction_status = REAR_WHEEL_DIRECTION_FORWARD_MESSAGE
- bool rear_wheel_direction = true
- Component rear_wheel_speed_slider
- Component rear_left_wheel_speed_slider
- Component rear_right_wheel_speed_slider
- Component rear_wheel_direction_checkbox_component
- Component front_wheels_angle_slider
- Component camera_servo_1_angle_slider
- Component camera_servo_2_angle_slider
- Component rear_wheel_menu_entry
- Component servo_menu_entry
- Component slider_container

Static Private Attributes

- static constexpr int DEFAULT_REAR_WHEEL_SPEED = 0
- static constexpr int DEFAULT_FRONT_WHEEL_ANGLE = 90
- static constexpr auto REAR_WHEEL_DIRECTION_FORWARD_MESSAGE = "Rear Wheel Direction: Forward"
- static constexpr auto REAR_WHEEL_DIRECTION_BACKWARD_MESSAGE = "Rear Wheel Direction: Backward"

12.13.1 Constructor & Destructor Documentation

12.13.1.1 DebugMovementRenderer()

```
car::display::console::component::debug::DebugMovementRenderer::DebugMovementRenderer ( )
[inline]
```

12.13.2 Member Function Documentation

12.13.2.1 element()

```
ftxui::Component car::display::console::component::debug::DebugMovementRenderer::element ( )
[inline]
```

12.13.2.2 getCameraServo1AngleSliderValue()

```
const int car::display::console::component::debug::DebugMovementRenderer::getCameraServo1↩
AngleSliderValue ( ) const  [inline]
```

12.13.2.3 getCameraServo2AngleSliderValue()

```
const int car::display::console::component::debug::DebugMovementRenderer::getCameraServo2↩
AngleSliderValue ( ) const  [inline]
```

12.13.2.4 getFrontWheelsAngleSliderValue()

```
const int car::display::console::component::debug::DebugMovementRenderer::getFrontWheels↩
AngleSliderValue ( ) const [inline]
```

12.13.2.5 getRearLeftWheelSpeedSliderValue()

```
const int car::display::console::component::debug::DebugMovementRenderer::getRearLeftWheel↩
SpeedSliderValue ( ) const [inline]
```

12.13.2.6 getRearRightWheelSpeedSliderValue()

```
const int car::display::console::component::debug::DebugMovementRenderer::getRearRightWheel↩
SpeedSliderValue ( ) const [inline]
```

12.13.2.7 getRearWheelDirectionSignal()

```
nod::signal< void(bool)> & car::display::console::component::debug::DebugMovementRenderer↩
::getRearWheelDirectionSignal ( ) [inline]
```

12.13.2.8 updateCameraServo1()

```
bool car::display::console::component::debug::DebugMovementRenderer::updateCameraServo1 ( )
[inline]
```

12.13.2.9 updateCameraServo2()

```
bool car::display::console::component::debug::DebugMovementRenderer::updateCameraServo2 ( )
[inline]
```

12.13.2.10 updateFrontWheels()

```
bool car::display::console::component::debug::DebugMovementRenderer::updateFrontWheels ( )
[inline]
```

12.13.2.11 updateRearWheels()

```
bool car::display::console::component::debug::DebugMovementRenderer::updateRearWheels ( )
[inline]
```

12.13.3 Member Data Documentation

12.13.3.1 camera_servo_1_angle_slider

```
Component car::display::console::component::debug::DebugMovementRenderer::camera_servo_1_↩
angle_slider  [private]
```

12.13.3.2 camera_servo_1_angle_slider_angle

```
int car::display::console::component::debug::DebugMovementRenderer::camera_servo_1_angle_↩
slider_angle = DEFAULT_FRONT_WHEEL_ANGLE  [private]
```

12.13.3.3 camera_servo_2_angle_slider

```
Component car::display::console::component::debug::DebugMovementRenderer::camera_servo_2_↩
angle_slider  [private]
```

12.13.3.4 camera_servo_2_angle_slider_angle

```
int car::display::console::component::debug::DebugMovementRenderer::camera_servo_2_angle_↩
slider_angle = DEFAULT_FRONT_WHEEL_ANGLE  [private]
```

12.13.3.5 DEFAULT_FRONT_WHEEL_ANGLE

```
constexpr int car::display::console::component::debug::DebugMovementRenderer::DEFAULT_FRONT_↩
WHEEL_ANGLE = 90  [static], [constexpr], [private]
```

12.13.3.6 DEFAULT_REAR_WHEEL_SPEED

```
constexpr int car::display::console::component::debug::DebugMovementRenderer::DEFAULT_REAR_↩
WHEEL_SPEED = 0  [static], [constexpr], [private]
```

12.13.3.7 front_wheels_angle_slider

```
Component car::display::console::component::debug::DebugMovementRenderer::front_wheels_angle↩
_slider  [private]
```

12.13.3.8 front_wheels_angle_slider_value

```
int car::display::console::component::debug::DebugMovementRenderer::front_wheels_angle_↩
slider_value = DEFAULT_FRONT_WHEEL_ANGLE  [private]
```

12.13.3.9 previous_camera_servo_1_angle_slider_angle

```
int car::display::console::component::debug::DebugMovementRenderer::previous_camera_servo_1_↩
angle_slider_angle = DEFAULT_FRONT_WHEEL_ANGLE  [private]
```

12.13.3.10 previous_camera_servo_2_angle_slider_angle

```
int car::display::console::component::debug::DebugMovementRenderer::previous_camera_servo_2_↩
angle_slider_angle = DEFAULT_FRONT_WHEEL_ANGLE  [private]
```

12.13.3.11 previous_front_wheels_angle_slider_value

```
int car::display::console::component::debug::DebugMovementRenderer::previous_front_wheels_↩
angle_slider_value = DEFAULT_FRONT_WHEEL_ANGLE  [private]
```

12.13.3.12 previous_rear_left_wheel_speed_slider_value

```
int car::display::console::component::debug::DebugMovementRenderer::previous_rear_left_wheel↩
_speed_slider_value = DEFAULT_REAR_WHEEL_SPEED  [private]
```

12.13.3.13 previous_rear_right_wheel_speed_slider_value

```
int car::display::console::component::debug::DebugMovementRenderer::previous_rear_right_↩
wheel_speed_slider_value = DEFAULT_REAR_WHEEL_SPEED  [private]
```

12.13.3.14 previous_rear_wheels_speed_slider_value

```
int car::display::console::component::debug::DebugMovementRenderer::previous_rear_wheels_↩
speed_slider_value = DEFAULT_REAR_WHEEL_SPEED  [private]
```

12.13.3.15 rear_left_wheel_speed_slider

```
Component car::display::console::component::debug::DebugMovementRenderer::rear_left_wheel_↩
speed_slider  [private]
```

12.13.3.16 rear_left_wheel_speed_slider_value

```
int car::display::console::component::debug::DebugMovementRenderer::rear_left_wheel_speed_↩
slider_value = DEFAULT_REAR_WHEEL_SPEED  [private]
```

12.13.3.17 rear_right_wheel_speed_slider

```
Component car::display::console::component::debug::DebugMovementRenderer::rear_right_wheel_↩
speed_slider  [private]
```

12.13.3.18 rear_right_wheel_speed_slider_value

```
int car::display::console::component::debug::DebugMovementRenderer::rear_right_wheel_speed_↩
slider_value = DEFAULT_REAR_WHEEL_SPEED  [private]
```

12.13.3.19 rear_wheel_direction

```
bool car::display::console::component::debug::DebugMovementRenderer::rear_wheel_direction =
true  [private]
```

12.13.3.20 REAR_WHEEL_DIRECTION_BACKWARD_MESSAGE

```
constexpr auto car::display::console::component::debug::DebugMovementRenderer::REAR_WHEEL_↩
DIRECTION_BACKWARD_MESSAGE = "Rear Wheel Direction:  Backward"  [static], [constexpr], [private]
```

12.13.3.21 rear_wheel_direction_checkbox_component

```
Component car::display::console::component::debug::DebugMovementRenderer::rear_wheel_direction↩
_checkbox_component  [private]
```

12.13.3.22 rear_wheel_direction_debounce

```
bool car::display::console::component::debug::DebugMovementRenderer::rear_wheel_direction_↩
debounce = false  [private]
```

12.13.3.23 REAR_WHEEL_DIRECTION_FORWARD_MESSAGE

```
constexpr auto car::display::console::component::debug::DebugMovementRenderer::REAR_WHEEL_↩
DIRECTION_FORWARD_MESSAGE = "Rear Wheel Direction:  Forward"  [static], [constexpr], [private]
```

12.13.3.24 rear_wheel_direction_signal

```
nod::signal<void(bool)> car::display::console::component::debug::DebugMovementRenderer::rear↩
_wheel_direction_signal  [private]
```

12.13.3.25 rear_wheel_direction_status

```
std::string car::display::console::component::debug::DebugMovementRenderer::rear_wheel_↩
direction_status = REAR_WHEEL_DIRECTION_FORWARD_MESSAGE  [private]
```

12.13.3.26 rear_wheel_menu_entry

```
Component car::display::console::component::debug::DebugMovementRenderer::rear_wheel_menu_↩
entry  [private]
```

12.13.3.27 rear_wheel_speed_slider

```
Component car::display::console::component::debug::DebugMovementRenderer::rear_wheel_speed_↩
slider  [private]
```

12.13.3.28 rear_wheels_speed_slider_value

```
int car::display::console::component::debug::DebugMovementRenderer::rear_wheels_speed_slider↩
_value = DEFAULT_REAR_WHEEL_SPEED  [private]
```

12.13.3.29 servo_menu_entry

```
Component car::display::console::component::debug::DebugMovementRenderer::servo_menu_entry
[private]
```

12.13.3.30 slider_container

```
Component car::display::console::component::debug::DebugMovementRenderer::slider_container
[private]
```

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

- tui/src/car/display/console/component/debug/DebugMovementRenderer.cxx

12.14 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_

12.14.1 Constructor & Destructor Documentation

12.14.1.1 DeviceManager()

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

12.14.2 Member Function Documentation

12.14.2.1 create()

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

12.14.2.2 getCameraDevice()

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

12.14.2.3 getLidarDevice()

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

12.14.2.4 initialize()

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

12.14.2.5 isRunning()

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

12.14.2.6 start()

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

12.14.2.7 stop()

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

12.14.2.8 terminate()

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

12.14.2.9 update()

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

12.14.3 Member Data Documentation

12.14.3.1 camera_device_

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

12.14.3.2 car_system

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

12.14.3.3 is_initialized_

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

12.14.3.4 is_running_

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

12.14.3.5 lidar_device_

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

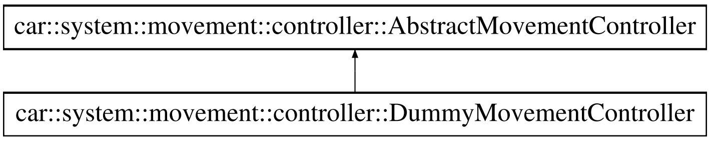
The documentation for this class was generated from the following files:

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

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

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

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


car::system::movement::controller::AbstractMovementController
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

12.15.1 Member Function Documentation

12.15.1.1 initialize()

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

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

12.15.1.2 setCameraServo1Angle()

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

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

12.15.1.3 setCameraServo2Angle()

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

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

12.15.1.4 setFrontWheelsAngle()

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

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

12.15.1.5 setRearLeftWheelDirectionToBackward()

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

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

12.15.1.6 setRearLeftWheelDirectionToForward()

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

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

12.15.1.7 setRearLeftWheelSpeed()

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

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

12.15.1.8 setRearRightWheelDirectionToBackward()

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

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

12.15.1.9 setRearRightWheelDirectionToForward()

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

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

12.15.1.10 setRearRightWheelSpeed()

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

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

12.15.1.11 setRearWheelsDirectionToBackward()

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

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

12.15.1.12 setRearWheelsDirectionToForward()

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

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

12.15.1.13 setRearWheelsSpeed()

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

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

12.15.1.14 stop()

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

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

12.15.1.15 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:

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

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

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

Public Attributes

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

12.16.1 Member Data Documentation

12.16.1.1 condition

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

12.16.1.2 error_message

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

12.16.1.3 uuid

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

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

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

12.17 car::configuration::JsonConfiguration Class Reference

Public Member Functions

- JsonConfiguration (std::string exe_dir)
- void setConfigFilePath (std::string config_file_path)
- const std::string & getConfigFilePath () const
- tl::expected< Configuration, std::string > loadConfiguration ()

Private Attributes

- const std::string exe_dir
- std::string config_file_path

12.17.1 Constructor & Destructor Documentation

12.17.1.1 JsonConfiguration()

```
car::configuration::JsonConfiguration::JsonConfiguration (
            std::string exe_dir )  [inline]
```

12.17.2 Member Function Documentation

12.17.2.1 getConfigFilePath()

```
const std::string & car::configuration::JsonConfiguration::getConfigFilePath ( ) const  [inline]
```

12.17.2.2 loadConfiguration()

```
tl::expected< Configuration, std::string > car::configuration::JsonConfiguration::loadConfiguration
( )  [inline]
```

12.17.2.3 setConfigFilePath()

```
void car::configuration::JsonConfiguration::setConfigFilePath (
            std::string config_file_path )  [inline]
```

12.17.3 Member Data Documentation

12.17.3.1 config_file_path

```
std::string car::configuration::JsonConfiguration::config_file_path  [private]
```

12.17.3.2 exe_dir

```
const std::string car::configuration::JsonConfiguration::exe_dir  [private]
```

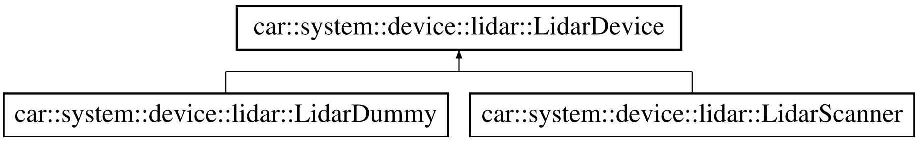
The documentation for this class was generated from the following file:

- tui/src/car/configuration/JsonConfiguration.cxx

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

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

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


car::system::device::lidar::LidarDevice
car::system::device::lidar::LidarDummy
car::system::device::lidar::LidarScanner

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

12.18.1 Member Function Documentation

12.18.1.1 disconnect()

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

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

12.18.1.2 getScanData()

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

12.18.1.3 initialize()

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

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

12.18.1.4 setScanData()

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

12.18.1.5 start()

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

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

12.18.1.6 stop()

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

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

12.18.1.7 terminate()

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

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

12.18.1.8 update()

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

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

12.18.2 Friends And Related Function Documentation

12.18.2.1 DeviceManager

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

12.18.3 Member Data Documentation

12.18.3.1 scan_data_

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

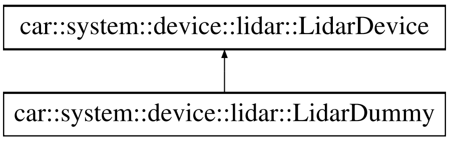
The documentation for this class was generated from the following file:

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

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

```
#include <LidarDummy.h>
```

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


car::system::device::lidar::LidarDevice
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

12.19.1 Constructor & Destructor Documentation

12.19.1.1 LidarDummy()

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

12.19.2 Member Function Documentation

12.19.2.1 disconnect()

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

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

12.19.2.2 initialize()

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

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

12.19.2.3 start()

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

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

12.19.2.4 stop()

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

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

12.19.2.5 terminate()

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

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

12.19.2.6 update()

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

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

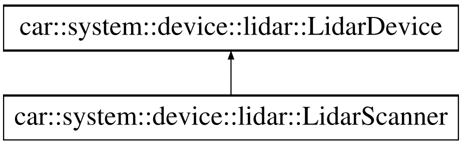
The documentation for this class was generated from the following file:

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

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

```
#include <LidarScanner.h>
```

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


car::system::device::lidar::LidarDevice
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

12.20.1 Constructor & Destructor Documentation

12.20.1.1 LidarScanner()

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

12.20.2 Member Function Documentation

12.20.2.1 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]
```

12.20.2.2 disconnect()

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

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

12.20.2.3 initialize()

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

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

12.20.2.4 start()

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

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

12.20.2.5 stop()

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

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

12.20.2.6 terminate()

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

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

12.20.2.7 update()

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

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

12.20.3 Member Data Documentation

12.20.3.1 configuration_

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

12.20.3.2 lidar_

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

12.20.3.3 running

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

12.20.3.4 scan_data_

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

12.20.3.5 scan_data_mutex_

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

12.20.3.6 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:

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

12.21 car::display::console::screen::LoggingScreen Class Reference

Public Member Functions

- LoggingScreen (std::shared_ptr< logging::vector_sink_mt > vector_sink)
- Component element ()

Private Attributes

- int selected_line = 0
- std::shared_ptr< logging::vector_sink_mt > vector_sink
- Component menu
- Component my_custom_menu
- ftxui::Elements line_elements

12.21.1 Constructor & Destructor Documentation

12.21.1.1 LoggingScreen()

```
car::display::console::screen::LoggingScreen::LoggingScreen (
             std::shared_ptr< logging::vector_sink_mt > vector_sink )  [inline]
```

12.21.2 Member Function Documentation

12.21.2.1 element()

```
Component car::display::console::screen::LoggingScreen::element ( )  [inline]
```

12.21.3 Member Data Documentation

12.21.3.1 line_elements

```
ftxui::Elements car::display::console::screen::LoggingScreen::line_elements  [private]
```

12.21.3.2 menu

```
Component car::display::console::screen::LoggingScreen::menu  [private]
```

12.21.3.3 my_custom_menu

```
Component car::display::console::screen::LoggingScreen::my_custom_menu  [private]
```

12.21.3.4 selected_line

```
int car::display::console::screen::LoggingScreen::selected_line = 0  [private]
```

12.21.3.5 vector_sink

```
std::shared_ptr<logging::vector_sink_mt> car::display::console::screen::LoggingScreen::vector↩
_sink  [private]
```

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

- tui/src/car/display/console/screen/LoggingScreen.cxx

12.22 car::display::console::component::main::MainErrorModal Class Reference

Public Member Functions

- MainErrorModal ()
- Component element ()
- void setErrorMessage (std::string message)

Public Attributes

- bool error_modal_shown = false

Private Attributes

- Component main_error_modal
- Element error_element

12.22.1 Constructor & Destructor Documentation

12.22.1.1 MainErrorModal()

```
car::display::console::component::main::MainErrorModal::MainErrorModal ( )  [inline]
```

12.22.2 Member Function Documentation

12.22.2.1 element()

```
Component car::display::console::component::main::MainErrorModal::element ( )  [inline]
```

12.22.2.2 setErrorMessage()

```
void car::display::console::component::main::MainErrorModal::setErrorMessage (
             std::string message )  [inline]
```

12.22.3 Member Data Documentation

12.22.3.1 error_element

```
Element car::display::console::component::main::MainErrorModal::error_element  [private]
```

12.22.3.2 error_modal_shown

```
bool car::display::console::component::main::MainErrorModal::error_modal_shown = false
```

12.22.3.3 main_error_modal

```
Component car::display::console::component::main::MainErrorModal::main_error_modal  [private]
```

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

- tui/src/car/display/console/component/main/MainErrorModal.cxx

12.23 car::display::console::component::main::MainExitModal Class Reference

Public Member Functions

- MainExitModal (std::function< void()> exit)
- Component element ()

Public Attributes

- bool exit_modal_shown = false

Private Attributes

- std::function< void()> exit
- Component main_exit_modal

12.23.1 Constructor & Destructor Documentation

12.23.1.1 MainExitModal()

```
car::display::console::component::main::MainExitModal::MainExitModal (
            std::function< void()> exit )  [inline]
```

12.23.2 Member Function Documentation

12.23.2.1 element()

```
Component car::display::console::component::main::MainExitModal::element ( )  [inline]
```

12.23.3 Member Data Documentation

12.23.3.1 exit

```
std::function<void()> car::display::console::component::main::MainExitModal::exit  [private]
```

12.23.3.2 exit_modal_shown

```
bool car::display::console::component::main::MainExitModal::exit_modal_shown = false
```

12.23.3.3 main_exit_modal

```
Component car::display::console::component::main::MainExitModal::main_exit_modal  [private]
```

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

- tui/src/car/display/console/component/main/MainExitModal.cxx

12.24 car::display::console::screen::MainScreen Class Reference

Public Member Functions

- MainScreen (std::shared_ptr< CarSystem > car_system, std::function< void()> exit)
- Component element ()

Private Attributes

- std::shared_ptr< CarSystem > car_system
- Box box
- ConnectButton connect_button
- MainExitModal main_exit_modal
- MainErrorModal main_error_modal
- Component info
- Component main_screen
- Component main_component

12.24.1 Constructor & Destructor Documentation

12.24.1.1 MainScreen()

```
car::display::console::screen::MainScreen::MainScreen (
            std::shared_ptr< CarSystem > car_system,
            std::function< void()> exit )  [inline]
```

12.24.2 Member Function Documentation

12.24.2.1 element()

```
Component car::display::console::screen::MainScreen::element ( )  [inline]
```

12.24.3 Member Data Documentation

12.24.3.1 box

```
Box car::display::console::screen::MainScreen::box  [private]
```

12.24.3.2 car_system

```
std::shared_ptr<CarSystem> car::display::console::screen::MainScreen::car_system  [private]
```

12.24.3.3 connect_button

```
ConnectButton car::display::console::screen::MainScreen::connect_button  [private]
```

12.24.3.4 info

```
Component car::display::console::screen::MainScreen::info  [private]
```

12.24.3.5 main_component

```
Component car::display::console::screen::MainScreen::main_component  [private]
```

12.24.3.6 main_error_modal

```
MainErrorModal car::display::console::screen::MainScreen::main_error_modal  [private]
```

12.24.3.7 main_exit_modal

```
MainExitModal car::display::console::screen::MainScreen::main_exit_modal  [private]
```

12.24.3.8 main_screen

```
Component car::display::console::screen::MainScreen::main_screen  [private]
```

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

- tui/src/car/display/console/screen/MainScreen.cxx

12.25 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

12.25.1 Constructor & Destructor Documentation

12.25.1.1 MessagingSystem()

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

12.25.2 Member Function Documentation

12.25.2.1 getCommandSignal()

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

12.25.2.2 getDisconnectSignal()

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

12.25.2.3 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>

12.25.2.4 getMessageSignal()

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

12.25.2.5 getSelectionSignal()

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

12.25.2.6 getUUID()

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

12.25.2.7 handleMessage()

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

Sends out signals depending on the type of message.

Parameters

message	

12.25.2.8 initialize()

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

Initializes the use of Websockets and initializes the Signals.

Parameters

configuration	

12.25.2.9 initializeWebSocket()

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

Creates a new Websocket object for use.

12.25.2.10 isConnected()

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

12.25.2.11 onDisconnect()

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

12.25.2.12 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	

12.25.2.13 onMessageCallback()

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

12.25.2.14 sendMessage()

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

12.25.2.15 setConfiguration()

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

12.25.2.16 stop()

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

12.25.2.17 terminate()

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

12.25.2.18 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>

12.25.3 Member Data Documentation

12.25.3.1 command_signal_

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

12.25.3.2 configuration_

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

12.25.3.3 connected_

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

12.25.3.4 message_signal_

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

12.25.3.5 on_disconnect_signal_

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

12.25.3.6 selection_signal_

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

12.25.3.7 uuid_

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

12.25.3.8 websocket_

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

12.25.3.9 websocket_url_

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

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

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

12.26 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

12.26.1 Constructor & Destructor Documentation

12.26.1.1 MovementSystem()

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

12.26.1.2 ∼MovementSystem()

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

12.26.2 Member Function Documentation

12.26.2.1 initialize()

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

12.26.2.2 setCameraServo1Angle()

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

12.26.2.3 setCameraServo2Angle()

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

12.26.2.4 setFrontWheelsAngle()

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

12.26.2.5 setRearLeftWheelDirectionToBackward()

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

12.26.2.6 setRearLeftWheelDirectionToForward()

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

12.26.2.7 setRearLeftWheelSpeed()

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

12.26.2.8 setRearRightWheelDirectionToBackward()

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

12.26.2.9 setRearRightWheelDirectionToForward()

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

12.26.2.10 setRearRightWheelSpeed()

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

12.26.2.11 setRearWheelsDirectionToBackward()

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

12.26.2.12 setRearWheelsDirectionToForward()

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

12.26.2.13 setRearWheelsSpeed()

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

12.26.2.14 start()

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

12.26.2.15 stop()

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

12.26.2.16 terminate()

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

12.26.3 Member Data Documentation

12.26.3.1 movement_controller

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

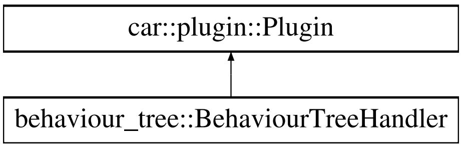
The documentation for this class was generated from the following file:

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

12.27 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

12.27.1 Member Function Documentation

12.27.1.1 getName()

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

Implemented in behaviour_tree::BehaviourTreeHandler.

12.27.1.2 initialize()

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

Implemented in behaviour_tree::BehaviourTreeHandler.

12.27.1.3 stop()

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

Implemented in behaviour_tree::BehaviourTreeHandler.

12.27.1.4 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:

- common/include/car/plugin/Plugin.h

12.28 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

12.28.1 Member Function Documentation

12.28.1.1 addPlugin()

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

12.28.1.2 getPlugin()

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

12.28.1.3 initialize()

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

12.28.1.4 stop()

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

12.28.1.5 terminate()

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

12.28.1.6 update()

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

12.28.2 Member Data Documentation

12.28.2.1 plugins

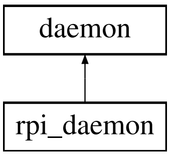
```
std::vector<std::shared_ptr<Plugin> > car::plugin::PluginManager::plugins  [private]
```

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

- common/include/car/plugin/PluginManager.h

12.29 rpi_daemon Class Reference

Inheritance diagram for rpi_daemon:



Public Member Functions

- void on_start (const INIReader reader) override
- void update ()
- void connect (const std::chrono::time_point< std::chrono::steady_clock > &now)
- void on_update () override
- void on_stop () override
- void on_reload (const INIReader reader) override

Private Attributes

- std::shared_ptr< CarSystem > car_system
- bool any_configuration_empty = false
- bool attempted_to_reconnect = false
- std::chrono::milliseconds connection_ms_interval = std::chrono::milliseconds(1000)
- std::chrono::time_point< std::chrono::steady_clock > last_connected

12.29.1 Member Function Documentation

12.29.1.1 connect()

```
void rpi_daemon::connect (
             const std::chrono::time_point< std::chrono::steady_clock > & now )  [inline]
```

12.29.1.2 on_reload()

```
void rpi_daemon::on_reload (
            const INIReader reader )  [inline], [override]
```

12.29.1.3 on_start()

```
void rpi_daemon::on_start (
            const INIReader reader )  [inline], [override]
```

12.29.1.4 on_stop()

```
void rpi_daemon::on_stop ( )  [inline], [override]
```

12.29.1.5 on_update()

```
void rpi_daemon::on_update ( )  [inline], [override]
```

12.29.1.6 update()

```
void rpi_daemon::update ( )  [inline]
```

12.29.2 Member Data Documentation

12.29.2.1 any_configuration_empty

```
bool rpi_daemon::any_configuration_empty = false  [private]
```

12.29.2.2 attempted_to_reconnect

```
bool rpi_daemon::attempted_to_reconnect = false  [private]
```

12.29.2.3 car_system

```
std::shared_ptr<CarSystem> rpi_daemon::car_system  [private]
```

12.29.2.4 connection_ms_interval

```
std::chrono::milliseconds rpi_daemon::connection_ms_interval = std::chrono::milliseconds(1000)
[private]
```

12.29.2.5 last_connected

```
std::chrono::time_point<std::chrono::steady_clock> rpi_daemon::last_connected  [private]
```

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

- daemon/src/main.cpp

12.30 car::display::console::component::settings::SettingsEditConfig Class Reference

Public Member Functions

- SettingsEditConfig (std::shared_ptr< system::CarSystem > car_system, std::shared_ptr< JsonConfiguration > json_configuration)
- Component element ()

Private Attributes

- std::shared_ptr< system::CarSystem > car_system
- std::shared_ptr< configuration::JsonConfiguration > json_configuration
- std::string placeholder = "settings/config.jsonc"
- std::string settings_file_path = "settings/config.jsonc"
- Component input_settings_file_path
- Component load_button

12.30.1 Constructor & Destructor Documentation

12.30.1.1 SettingsEditConfig()

```
car::display::console::component::settings::SettingsEditConfig::SettingsEditConfig (
            std::shared_ptr< system::CarSystem > car_system,
            std::shared_ptr< JsonConfiguration > json_configuration )  [inline]
```

12.30.2 Member Function Documentation

12.30.2.1 element()

```
Component car::display::console::component::settings::SettingsEditConfig::element ( )  [inline]
```

12.30.3 Member Data Documentation

12.30.3.1 car_system

```
std::shared_ptr<system::CarSystem> car::display::console::component::settings::SettingsEdit↵
Config::car_system  [private]
```

12.30.3.2 input_settings_file_path

```
Component car::display::console::component::settings::SettingsEditConfig::input_settings_↵
file_path  [private]
```

12.30.3.3 json_configuration

```
std::shared_ptr<configuration::JsonConfiguration> car::display::console::component::settings↵
::SettingsEditConfig::json_configuration  [private]
```

12.30.3.4 load_button

```
Component car::display::console::component::settings::SettingsEditConfig::load_button  [private]
```

12.30.3.5 placeholder

```
std::string car::display::console::component::settings::SettingsEditConfig::placeholder =
"settings/config.jsonc"  [private]
```

12.30.3.6 settings_file_path

```
std::string car::display::console::component::settings::SettingsEditConfig::settings_file_path
= "settings/config.jsonc"  [private]
```

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

- tui/src/car/display/console/component/settings/SettingsEditConfig.cxx

12.31 car::display::console::screen::SettingsScreen Class Reference

Public Member Functions

- SettingsScreen (std::shared_ptr< CarSystem > car_system, std::shared_ptr< JsonConfiguration > json_↩ configuration)
- Component element ()
- void update ()

Private Attributes

- std::shared_ptr< CarSystem > car_system
- SettingsEditConfig settings_edit_config
- DebugEnabler debug_enabler
- DebugLidarCheckbox debug_lidar_checkbox
- DebugMovementRenderer debug_movement_renderer
- DebugMessagingTextbox debug_messaging_text_box

12.31.1 Constructor & Destructor Documentation

12.31.1.1 SettingsScreen()

```
car::display::console::screen::SettingsScreen::SettingsScreen (
             std::shared_ptr< CarSystem > car_system,
             std::shared_ptr< JsonConfiguration > json_configuration )  [inline]
```

12.31.2 Member Function Documentation

12.31.2.1 element()

```
Component car::display::console::screen::SettingsScreen::element ( ) [inline]
```

12.31.2.2 update()

```
void car::display::console::screen::SettingsScreen::update ( ) [inline]
```

12.31.3 Member Data Documentation

12.31.3.1 car_system

```
std::shared_ptr<CarSystem> car::display::console::screen::SettingsScreen::car_system [private]
```

12.31.3.2 debug_enabler

```
DebugEnabler car::display::console::screen::SettingsScreen::debug_enabler [private]
```

12.31.3.3 debug_lidar_checkbox

```
DebugLidarCheckbox car::display::console::screen::SettingsScreen::debug_lidar_checkbox [private]
```

12.31.3.4 debug_messaging_text_box

```
DebugMessagingTextbox car::display::console::screen::SettingsScreen::debug_messaging_text_box
[private]
```

12.31.3.5 debug_movement_renderer

```
DebugMovementRenderer car::display::console::screen::SettingsScreen::debug_movement_renderer
[private]
```

12.31.3.6 settings_edit_config

```
SettingsEditConfig car::display::console::screen::SettingsScreen::settings_edit_config  [private]
```

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

- tui/src/car/display/console/screen/SettingsScreen.cxx

12.32 TB6612 Class Reference

```
#include <TB6612.h>
```

Public Member Functions

- TB6612 (int motor_pin, int pwm_pin)
- void setPWM (int value)
- void forward ()
- void backward ()
- void stop ()
- void setOffset (bool offset)
- const int & getMotorPin () const
- const int & getPWMPin () const

Private Attributes

- const int motor_pin
- const int pwm_pin
- bool offset = true

12.32.1 Constructor & Destructor Documentation

12.32.1.1 TB6612()

```
TB6612::TB6612 (
            int motor_pin,
            int pwm_pin )
```

12.32.2 Member Function Documentation

12.32.2.1 backward()

```
void TB6612::backward ( )
```

12.32.2.2 forward()

```
void TB6612::forward ( )
```

12.32.2.3 getMotorPin()

```
const int & TB6612::getMotorPin ( ) const
```

12.32.2.4 getPWMPin()

```
const int & TB6612::getPWMPin ( ) const
```

12.32.2.5 setOffset()

```
void TB6612::setOffset (
             bool offset )
```

12.32.2.6 setPWM()

```
void TB6612::setPWM (
             int value )
```

12.32.2.7 stop()

```
void TB6612::stop ( )
```

12.32.3 Member Data Documentation

12.32.3.1 motor_pin

```
const int TB6612::motor_pin  [private]
```

12.32.3.2 offset

```
bool TB6612::offset = true  [private]
```

12.32.3.3 pwm_pin

```
const int TB6612::pwm_pin  [private]
```

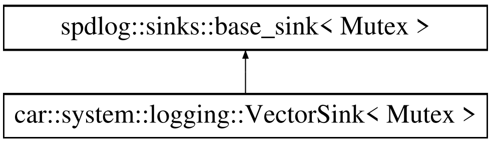
The documentation for this class was generated from the following files:

- repository/packages/t/tb6612/tb6612/include/TB6612.h
- repository/packages/t/tb6612/tb6612/src/TB6612.cpp

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

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

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


spdlog::sinks::base_sink< Mutex >
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

12.33.1 Constructor & Destructor Documentation

12.33.1.1 VectorSink()

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

12.33.2 Member Function Documentation

12.33.2.1 flush_()

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

12.33.2.2 get_log_messages()

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

12.33.2.3 sink_it_()

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

12.33.3 Member Data Documentation

12.33.3.1 log_messages

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

12.33.3.2 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:

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

Chapter 13

File Documentation

13.1 behaviour_tree/src/main.cpp File Reference

```
#include <iostream>
#include <chrono>
#include <filesystem>
#include <memory>
#include <cxxopts.hpp>
#include "car/system/CarSystem.h"
#include "car/system/device/lidar/LidarScanner.h"
#include "car/system/device/lidar/LidarDummy.h"
#include "car/system/movement/controller/DummyMovementController.h"
#include "car/system/movement/controller/DeviceMovementController.h"
#include "car/plugin/PluginManager.h"
#include "behaviour_tree/BehaviourTreeParser.hpp"
#include "behaviour_tree/node/custom/CarCustomNodeParser.hpp"
#include "behaviour_tree/BehaviourTreeHandler.hpp"
#include <thread>
#include <unistd.h>
#include <termios.h>
```

Functions

- int kbhit (void)
- int main (int argc, const char ∗argv[ ])

13.1.1 Function Documentation

13.1.1.1 kbhit()

```
int kbhit (
            void  )
```

13.1.1.2 main()

```
int main (
            int argc,
            const char * argv[] )
```

13.2 daemon/src/main.cpp File Reference

```
#include <iostream>
#include <chrono>
#include <filesystem>
#include <memory>
#include <daemonpp/daemon.hpp>
#include <cpptrace/cpptrace.hpp>
#include <fmt/format.h>
#include <spdlog/sinks/callback_sink.h>
#include "car/system/CarSystem.h"
#include "car/system/device/lidar/LidarScanner.h"
#include "car/system/device/lidar/LidarDummy.h"
#include "car/system/movement/controller/DummyMovementController.h"
#include "car/system/movement/controller/DeviceMovementController.h"
#include "behaviour_tree/BehaviourTreeHandler.hpp"
#include "car/plugin/PluginManager.h"
```

Classes

- class rpi_daemon

Functions

- std::unique_ptr< LidarDevice > getLidarDevice (std::shared_ptr< Configuration > configuration)
- std::unique_ptr< AbstractMovementController > getMovementController ()
- void terminate_handler ()
- int main (int argc, const char *argv[])

13.2.1 Function Documentation

13.2.1.1 getLidarDevice()

```
std::unique_ptr< lidar::LidarDevice > getLidarDevice (
            std::shared_ptr< Configuration > configuration )
```

13.2.1.2 getMovementController()

```
std::unique_ptr< AbstractMovementController > getMovementController ( )
```

13.2.1.3 main()

```
int main (
            int argc,
            const char * argv[ ] )
```

13.2.1.4 terminate_handler()

```
void terminate_handler ( )
```

13.3 tui/src/main.cpp File Reference

```
#include <optional>
#include <string>
#include <thread>
#include <chrono>
#include <fmt/format.h>
#include "car/display/console/CarConsole.h"
#include "car/configuration/JsonConfiguration.cxx"
#include "car/system/CarSystem.h"
#include "car/system/device/DeviceManager.h"
#include "car/system/device/lidar/LidarDevice.h"
#include "car/system/device/lidar/LidarDummy.h"
#include "car/system/device/lidar/LidarScanner.h"
#include "car/system/device/CameraDevice.h"
#include "car/system/movement/controller/DummyMovementController.h"
#include "car/system/movement/controller/DeviceMovementController.h"
#include "car/plugin/PluginManager.h"
#include "car/system/logging/VectorSink.h"
#include "behaviour_tree/BehaviourTreeHandler.hpp"
```

Functions

- std::unique_ptr< LidarDevice > getLidarDevice (std::shared_ptr< Configuration > configuration)
- std::unique_ptr< AbstractMovementController > getMovementController ()
- int main (int argc, char *argv[ ])

13.3.1 Function Documentation

13.3.1.1 getLidarDevice()

```
std::unique_ptr< LidarDevice > getLidarDevice (
            std::shared_ptr< Configuration > configuration )
```

13.3.1.2 getMovementController()

```
std::unique_ptr< AbstractMovementController > getMovementController ( )
```

13.3.1.3 main()

```
int main (
            int argc,
            char * argv[] )
```

13.4 common/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

13.5 BehaviourTreeHandler.hpp

Go to the documentation of this file.

```
#ifndef BEHAVIOURTREEHANDLER_HPP
#define BEHAVIOURTREEHANDLER_HPP

#pragma once

#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"

namespace behaviour_tree
{
    class BehaviourTreeHandler : public car::plugin::Plugin
    {
    public:
        void initialize(std::shared_ptr<car::system::CarSystem> car_system) final override
        {
            this->car_system = car_system;
            // The BehaviourTreeParser does not come with a CustomNodeParser since each program can have
    a different set of Action nodes

    BehaviourTreeParser::instance().setCustomNodeParser(std::make_shared<node::custom::CarCustomNodeParser>(CarCustomNodePa

    this->car_system->getMessagingSystem()->getCommandSignal().connect(std::bind(&BehaviourTreeHandler::handleCommand,
    this, std::placeholders::_1, std::placeholders::_2));
        }

        void handleCommand(const std::string message, const rapidjson::Document &message_json)
        {
            const std::string command = message_json["command"].GetString();
            if (command != "behaviour_tree")
            {
                spdlog::error(R"(The property "command" does not match "behaviour_tree", {})", command);
                return;
            }
            if (!message_json.HasMember("action") || !message_json["action"].IsString())
            {
                spdlog::error(R"(The property "action" does not exist in the given json.)");
                return;
            }
            const std::string action = message_json["action"].GetString();
            switch (utils::hash(action))
            {
            case utils::hash("set"):
            {
                this->setBehaviourTree(message_json);
                break;
            }
            case utils::hash("start"):
            {
                this->startBehaviourTree();
                break;
            }
            case utils::hash("stop"):
            {
                this->stopBehaviourTree();
                break;
            }
            default:
                spdlog::error(R"(The property "action" does not match "set" or "start", {})", action);
                break;
            };
        }

        void setBehaviourTree(const rapidjson::Document &message_json)
        {
            if (!message_json.HasMember("data") || !message_json["data"].IsString())
            {
                spdlog::error(R"(The property "data" does not exist in the given json.)");
                return;
            }
            try
            {
```

```
            auto maybe_behaviour_tree = BehaviourTreeParser::instance().parseXML(message_json["data"].GetString());
            if (!maybe_behaviour_tree.has_value())
            {
                spdlog::error(R"(Unable to parse the given behaviour tree | {})", maybe_behaviour_tree.error());
                return;
            }
            auto &behaviour_tree = maybe_behaviour_tree.value();
            spdlog::info("Behaviour tree parsed successfully | {}", behaviour_tree->toString());
            this->_setBehaviourTree(behaviour_tree);
        }
        catch (std::exception &e)
        {
            spdlog::error("An error has occurred while parsing the given behaviour tree: {}", e.what());
        }
    }

    void startBehaviourTree()
    {
        assert(this->car_system != nullptr);
        if (this->behaviour_tree == nullptr)
        {
            spdlog::error("The Behaviour tree has not been set");
            return;
        }
        this->behaviour_tree->resetCycles();
        this->tick_count = 0;
        std::shared_ptr<Context> context = std::make_shared<CarContext>(this->behaviour_tree, this->car_system);
        this->context = context;
        spdlog::info("Starting the given Behaviour tree");
    }

    void stopBehaviourTree()
    {
        assert(this->car_system != nullptr);
        this->context = nullptr;
        spdlog::info("Stopped any Behaviour Tree context");
    }

    void update() final override
    {
        if (this->context == nullptr)
        {
            return;
        }
        if (this->context->canRun())
        {
            const std::chrono::time_point<std::chrono::steady_clock> now = std::chrono::steady_clock::now();
            // TODO:
            if (now - this->last_connected >= this->car_system->getConfiguration()->behaviour_tree_update_ms_interval) {
                this->context->update(this->tick_count);
                this->tick_count++;
                this->last_connected = now;
            }
        }
        else
        {
            this->context = nullptr;
        }
    }

    void stop() final override
    {
        this->context = nullptr;
    }

    std::string getName() final override
    {
        return "BehaviourTreeHandler";
    }

    void _setBehaviourTree(std::shared_ptr<BehaviourTree> behaviour_tree)
    {
        this->behaviour_tree = behaviour_tree;
    }

private:
    std::shared_ptr<car::system::CarSystem> car_system;

    std::shared_ptr<BehaviourTree> behaviour_tree;
    std::shared_ptr<Context> context;

```

```
        int tick_count = 0;

        // This is initialized as 0
        std::chrono::time_point<std::chrono::steady_clock> last_connected;
    };
} // namespace behaviour_tree

#endif
```

13.6 common/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

13.7 CarContext.hpp

Go to the documentation of this file.

```
#ifndef BEHAVIOUR_TREE_CARCONTEXT_HPP
#define BEHAVIOUR_TREE_CARCONTEXT_HPP

#pragma once

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

namespace behaviour_tree
{
    class CarContext : public Context
    {
    public:
        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))
        {
        }

        std::shared_ptr<car::system::CarSystem> getCarSystem() const
        {
            return this->car_system;
        }

        void _() override{};

    private:
        std::shared_ptr<car::system::CarSystem> car_system;
    };
}

#endif
```

13.8 common/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

13.9 Configuration.h

Go to the documentation of this file.

```
#ifndef CONFIGURATION_H
#define CONFIGURATION_H

#pragma once

#include <chrono>
#include <optional>
#include <string>

#include <tl/expected.hpp>

namespace car::configuration
{
    struct Configuration
    {
        std::string host = "127.0.0.1:3000";

        int camera_index = 0;
        void setCameraFps(const int camera_fps)
        {
            this->camera_fps = camera_fps;
            this->camera_fps_interval = 1000 / camera_fps;
        }
        const int getCameraFpsInterval() { return this->camera_fps_interval; }
        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:
        int camera_fps = 60;
        int camera_fps_interval = 1000;
    };
};

#endif
```

13.10 common/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

13.11 Plugin.h

Go to the documentation of this file.

```
#ifndef PLUGIN_H
#define PLUGIN_H

#pragma once

#include <string>
#include <memory>

namespace car::system
{
    class CarSystem;
}

namespace car::plugin
{
    class Plugin
    {
    public:
        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;
    };
}

#endif
```

13.12 common/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

13.13 PluginManager.h

Go to the documentation of this file.

```
#ifndef PLUGIN_MANAGER_H
#define PLUGIN_MANAGER_H

#pragma once

#include <vector>
#include <memory>

#include "utils/Utility.hpp"
#include "utils/TypeName.hpp"

#include "Plugin.h"

namespace car::system
{
    class CarSystem;
}

namespace car::plugin
{
    class PluginManager
    {
    public:
        void initialize(std::shared_ptr<system::CarSystem> car_system)
        {
            for (std::shared_ptr<Plugin>& plugin : this->plugins)
            {
                plugin->initialize(car_system);
            }
        }

        void update()
        {
            for (std::shared_ptr<Plugin>& plugin : this->plugins)
            {
                plugin->update();
            }
        }

        void stop()
        {
            for (std::shared_ptr<Plugin>& plugin : this->plugins)
            {
                plugin->stop();
            }
        }

        void terminate()
        {
            this->stop();
        }

        void addPlugin(std::shared_ptr<Plugin> plugin)
        {
            this->plugins.push_back(plugin);
        }

        template<typename T>
        std::shared_ptr<T> getPlugin()
        {
            static_assert(std::is_base_of<Plugin, T>::value, "T must be a Plugin");
            std::string type_name = std::string(utils::TypeName<T>());
            type_name = utils::getStringAfterLastColon(type_name);

            for (std::shared_ptr<Plugin>& plugin : this->plugins)
            {
                if (plugin->getName() == type_name)
                {
                    return plugin;
                }
            }

            return nullptr;
        }


    private:
        std::vector<std::shared_ptr<Plugin» plugins;
    };
}

#endif
```

13.14 common/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

13.15 CarSystem.h

Go to the documentation of this file.

```
#ifndef CARSYSTEM_H
#define CARSYSTEM_H

#pragma once

#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"

using namespace car::configuration;
using namespace car::plugin;
using namespace car::system::device;
using namespace car::system::messaging;
using namespace car::system::movement;

namespace car::system
{
    // Make sure this is stored as a shared_ptr
    class CarSystem : public std::enable_shared_from_this<CarSystem>
    {
    public:
        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();

        void update();

        const std::shared_ptr<Configuration> getConfiguration() const { return this->configuration_; };
        void setConfiguration(std::shared_ptr<Configuration> configuration);

        DeviceManager *getDeviceManager() const
        {
            return this->device_manager_.get();
        }

        MessagingSystem *getMessagingSystem() const
        {
            return this->messaging_system_.get();
        }

        MovementSystem *getMovementSystem() const
        {
            return this->movement_system_.get();
        }

        template <typename T>
        const std::shared_ptr<T> getPlugin() const { return this->plugin_manager_->getPlugin<T>(); }

    private:
        void sendData();

        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;
    };
}

#endif
```

13.16 common/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

13.17 CameraDevice.h

Go to the documentation of this file.

```
#ifndef CAMERADEVICE_H
#define CAMERADEVICE_H

#pragma once

#include <vector>

#include <tl/expected.hpp>
#include <opencv2/opencv.hpp>

#include "car/configuration/Configuration.h"

namespace car::system::device
{
    class DeviceManager;
    class CameraDevice
    {
    public:
        CameraDevice(std::shared_ptr<configuration::Configuration> configuration) :
    configuration(configuration) {}

        CameraDevice(const CameraDevice&) = delete;
        CameraDevice& operator=(const CameraDevice&) = delete;

        CameraDevice(CameraDevice&&) = delete;
        CameraDevice& operator=(CameraDevice&&) = delete;

        ~CameraDevice() = default;

    public:
        [[nodiscard]] static tl::expected<std::unique_ptr<CameraDevice>, std::string>
    create(std::shared_ptr<configuration::Configuration> configuration);
        std::string getFrameBuffer() const;

    protected:
        void start();
        void update();
        void stop();
        void disconnect();
        void terminate();

        friend class DeviceManager;

    private:
        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;
    };
}

#endif
```

13.18 common/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

13.19 DeviceManager.h

Go to the documentation of this file.

```
#ifndef DEVICE_MANAGER_H
#define DEVICE_MANAGER_H

#pragma once

#include <memory>

#include <tl/expected.hpp>

#include "car/configuration/Configuration.h"

#include "CameraDevice.h"
#include "lidar/LidarDevice.h"
#include "lidar/LidarScanner.h"

using namespace car::configuration;

namespace car::system
{
    class CarSystem;
}

namespace car::system::device
{
    class DeviceManager {
    public:
        [[nodiscard]] static tl::expected<std::unique_ptr<DeviceManager>, std::string>
    create(std::shared_ptr<Configuration> configuration);

        DeviceManager(std::unique_ptr<CameraDevice> camera_device, std::unique_ptr<lidar::LidarDevice>
    lidar_device) :
            camera_device_(std::move(camera_device)),
            lidar_device_(std::move(lidar_device))
        {
        }

        CameraDevice* getCameraDevice() {
            return this->camera_device_.get();
        }

        lidar::LidarDevice* getLidarDevice() {
            return this->lidar_device_.get();
        }

        const bool isRunning() const {
            return this->is_running_;
        }

        void initialize(std::shared_ptr<system::CarSystem> car_system);
        void start();
        void update();
        void stop();
        void terminate();

    private:
        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_;
    };
}

#endif
```

13.20 common/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

13.21 LidarDevice.h

Go to the documentation of this file.

```
#ifndef LIDARDEVICE_H
#define LIDARDEVICE_H

#pragma once

#include <vector>

#include <rapidjson/document.h>

#include <RPLidar.h>

using namespace rplidar;

namespace car::system::device {
    class DeviceManager;
}

namespace car::system::device::lidar
{
    class LidarDevice
    {
    public:
        std::vector<Measure> getScanData() const { return this->scan_data_; }

        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:
        friend class DeviceManager;

        void setScanData(const std::vector<Measure>& scan_data)
        {
            this->scan_data_ = scan_data;
        }

        std::vector<Measure> scan_data_;
    };
}

#endif
```

13.22 common/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

13.23 LidarDummy.h

Go to the documentation of this file.

```
#ifndef LIDARDUMMY_H
#define LIDARDUMMY_H

#pragma once

#include <fstream>
#include <spdlog/spdlog.h>

#include "LidarDevice.h"

namespace car::system::device::lidar
{
    class LidarDummy final : public LidarDevice
    {
    public:
        LidarDummy()
        {
            spdlog::warn("Currently using the 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 {};

    private:
    };
}

#endif
```

13.24 common/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

13.25 LidarScanner.h

Go to the documentation of this file.

```
#ifndef LIDARSCANNER_H
#define LIDARSCANNER_H

#pragma once

#include "LidarDevice.h"

#include <memory>
#include <variant>

#include <RPLidar.h>
#include <tl/expected.hpp>

#include "car/configuration/Configuration.h"

using namespace rplidar;

namespace car::system::device::lidar
{
    class LidarScanner final : public LidarDevice
    {
    public:
        [[nodiscard]] static tl::expected<std::unique_ptr<LidarScanner>, std::string> create(std::shared_ptr<configuration::Configuration> configuration) noexcept
        {
            auto maybe_lidar = RPLidar::create(configuration->lidar_port);
            if (maybe_lidar.has_value())
            {
                return std::make_unique<LidarScanner>(configuration, std::move(maybe_lidar.value()));
            }
            else
            {
                return tl::make_unexpected(maybe_lidar.error());
            }
        }

        // Do not call this constructor directly. Use the create method instead.
        LidarScanner(std::shared_ptr<configuration::Configuration> configuration, std::unique_ptr<RPLidar> lidar) : configuration_(configuration), lidar_(std::move(lidar)) {
        }

        void start() final override
        {
            this->running = true;
            this->lidar_->start_motor();
            std::lock_guard<std::mutex> lock(this->scan_data_mutex_);
            this->scan_generator_ = this->lidar_->iter_scans();
        };

        void update() final override
        {
            if (this->running) {
                std::lock_guard<std::mutex> lock(this->scan_data_mutex_);
                const auto& scan_generator = std::get<std::function<std::vector<Measure>()»(this->scan_generator_);
                this->setScanData(scan_generator());
            }
        };

        void stop() final override
        {
            if (this->running) {
                this->running = false;
                std::lock_guard<std::mutex> lock(this->scan_data_mutex_);
                this->scan_generator_ = nullptr;
                this->lidar_->stop();
                this->lidar_->stop_motor();
            }
        }

        void initialize() final override
        {
        };

        void disconnect() final override
        {
            if (this->running) {
                this->running = false;
                std::lock_guard<std::mutex> lock(this->scan_data_mutex_);
                this->scan_generator_ = nullptr;
                this->lidar_->disconnect();
            }
        }

        void terminate() final override
        {
            this->stop();
            this->disconnect();
        }

    private:
        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_;
    };
}

#endif
```

13.26 common/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 >

13.27 VectorSink.h

Go to the documentation of this file.

```
#ifndef VECTORSINK_CXX
#define VECTORSINK_CXX

#include <algorithm>
#include <vector>

#include <fmt/format.h>

#include <spdlog/sinks/base_sink.h>
#include <spdlog/details/synchronous_factory.h>
#include <iostream>

namespace car::system::logging
{
    template <typename Mutex>
    class VectorSink : public spdlog::sinks::base_sink<Mutex>
    {
    public:
        VectorSink(int max_lines) : max_lines(max_lines)
        {
        }

        void sink_it_(const spdlog::details::log_msg &msg) override
        {
            spdlog::memory_buf_t formatted;
            spdlog::sinks::base_sink<Mutex>::formatter_->format(msg, formatted);
            if (this->log_messages.size() < this->max_lines)
            {
                this->log_messages.push_back(std::string(formatted.data(), formatted.size()));
            }
            else
            {
                std::rotate(this->log_messages.begin(), this->log_messages.begin() + 1, this->log_messages.end());
                this->log_messages[this->log_messages.size() - 1] = std::string(formatted.data(), formatted.size());
            }
        };

        void flush_() override
        {
            this->log_messages.clear();
        };

        const std::vector<std::string> &get_log_messages() const
        {
            return this->log_messages;
        }

    private:
        const int max_lines;

        std::vector<std::string> log_messages;
    };
    using vector_sink_mt = VectorSink<std::mutex>;
}

#endif
```

13.28 common/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

13.29 MessagingSystem.h

Go to the documentation of this file.

```
#ifndef MESSAGINGSYSTEM_H
#define MESSAGINGSYSTEM_H

#pragma once

#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"

namespace car::system::messaging
{
    class MessagingSystem
    {
    public:
        MessagingSystem();

        void initialize(std::shared_ptr<configuration::Configuration> configuration);
        void initializeWebSocket();
        const tl::expected<nullptr_t, std::string> tryConnect();
        void stop();
        void terminate();

        // Necessary for the reloading the configuration
        void setConfiguration(std::shared_ptr<configuration::Configuration> configuration);

        nod::signal<void(const std::string, const rapidjson::Document&)>& getCommandSignal() { return this->command_signal_; }
        nod::signal<void(const std::string, const rapidjson::Document&)>& getSelectionSignal() { return this->selection_signal_; }
        nod::signal<void(const std::string)>& getMessageSignal() { return this->message_signal_; }
        nod::signal<void(const std::string)>& getDisconnectSignal() { return this->on_disconnect_signal_; }

        void onMessageCallback(const ix::WebSocketMessagePtr& msg) const;
        void onDisconnect(const std::string);

        const std::string getUUID() const { return this->uuid_; }
        void handleMessage(const std::string& message) const;
        void sendMessage(const std::string& message);

        struct FirstMessageStruct
        {
            std::string error_message;
            std::string uuid;
            std::condition_variable condition;
        };
        void onFirstMessage(const ix::WebSocketMessagePtr& msg, FirstMessageStruct&
    first_message_struct);

        const bool isConnected() const { return this->connected_; }

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

        std::shared_ptr<configuration::Configuration> configuration_;

        std::unique_ptr<ix::WebSocket> websocket_;
        std::string websocket_url_;

        std::string uuid_;

        bool connected_ = false;
    };
};

#endif
```

13.30 common/include/car/system/messaging/StreamType.h File Reference

Enumerations

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

13.30.1 Enumeration Type Documentation

13.30.1.1 StreamType

```
enum StreamType
```

Enumerator

None	
Lidar	
Camera	
Both	

13.31 StreamType.h

Go to the documentation of this file.

```
#ifndef STREAM_TYPE_H
#define STREAM_TYPE_H

#pragma once

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

#endif
```

13.32 common/include/car/system/movement/controller/AbstractMovementController.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

13.33 AbstractMovementController.h

Go to the documentation of this file.

```
#ifndef ABSTRACTWHEELCONTROLLER_H
#define ABSTRACTWHEELCONTROLLER_H

#pragma once

namespace car::system::movement::controller
{
    class AbstractMovementController
    {
    public:
        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;
    };
} // namespace car::system::movement::controller

#endif
```

13.34 common/include/car/system/movement/controller/DeviceMovementController.h File Reference

13.35 DeviceMovementController.h

Go to the documentation of this file.

```
#ifdef __linux__
#ifndef DEVICEMOVEMENTCONTROLLER_H
#define DEVICEMOVEMENTCONTROLLER_H

#pragma once

#include <memory>

#include "AbstractMovementController.h"

#include "car/system/movement/devices/Servo.h"
#include "car/system/movement/devices/RearWheel.h"

using namespace car::system::movement::devices;

namespace car::system::movement::controller
{
    static constexpr int Motor_A = 17;
    static constexpr int Motor_B = 27;
    static constexpr int PWM_A = 4;
    static constexpr int PWM_B = 5;

    static constexpr int MIN_PULSE_WIDTH = 900;
    static constexpr int MAX_PULSE_WIDTH = 2100;
    static constexpr int FREQUENCY = 50;

    static constexpr int BUS_NUMBER = 1;

    class DeviceMovementController : public AbstractMovementController
    {
    public:
        [[nodiscard]] DeviceMovementController();

        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;

    private:
        std::shared_ptr<PCA9685> pwm;

        std::unique_ptr<Servo> front_wheels_;
        std::unique_ptr<Servo> camera_servo_1_;
        std::unique_ptr<Servo> camera_servo_2_;

        std::unique_ptr<RearWheel> rear_left_wheel_;
        std::unique_ptr<RearWheel> rear_right_wheel_;
    };
} // namespace car::system::movement::controller

```

```
#endif
#endif // __linux__
```

13.36 common/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

13.37 DummyMovementController.h

Go to the documentation of this file.

```
#ifndef DUMMYWHEELCONTROLLER_H
#define DUMMYWHEELCONTROLLER_H

#pragma once

#include "AbstractMovementController.h"

namespace car::system::movement::controller
{
    class DummyMovementController : public AbstractMovementController
    {
    public:
        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;

    private:
    };
} // namespace car::system::movement::controller

#endif
```

13.38 common/include/car/system/movement/devices/RearWheel.h File Reference

13.39 RearWheel.h

Go to the documentation of this file.

```
#ifdef __linux__
#ifndef REARWHEEL_H
#define REARWHEEL_H

#include <memory>

#include <PCA9685.h>
#include <TB6612.h>

// Made with the help of ChatGPT

namespace car::system::movement::devices
{
    class RearWheel
    {
    public:
        RearWheel(std::shared_ptr<PCA9685> pwm, std::unique_ptr<TB6612> motor);

        void forward();

        void backward();

        void stop();

        int getSpeed() const;

        void setSpeed(const int speed);

        void ready();

    private:
        std::shared_ptr<PCA9685> pwm_;
        std::unique_ptr<TB6612> motor_;

        int speed_;
    };
} // namespace car::system::movement::wheels

#endif
#endif
```

13.40 common/include/car/system/movement/devices/Servo.h File Reference

13.41 Servo.h

Go to the documentation of this file.

```
#ifdef __linux__
#ifndef SERVO_H
#define SERVO_H

#include <algorithm>
#include <memory>

#include <PCA9685.h>

namespace car::system::movement::devices
{
    class Servo
    {
    private:
        static int map(int x, int in_min, int in_max, int out_min, int out_max)
        {
            return ((x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min);
        }

        static constexpr int MIN_PULSE_WIDTH = 900;
        static constexpr int MAX_PULSE_WIDTH = 2100;
        static constexpr int FREQUENCY = 50;

    public:
        Servo(std::shared_ptr<PCA9685> pwm, int channel);

        // Some of the code was from: https://github.com/chaoticmachinery/pca9685
        int getAnalogAngle() const;

        int getAngle() const;

        // Some of the code was from: https://github.com/chaoticmachinery/pca9685
        void setAngle(const int angle);

        void reset();

    private:
        const std::shared_ptr<PCA9685> pwm_;
        const int channel_;

        int angle_;
    };
} // namespace car::system::movement::wheels

#endif
#endif // __linux__
```

13.42 common/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

13.43 MovementSystem.h

Go to the documentation of this file.

```
#ifndef MOVEMENTSYSTEM_H
#define MOVEMENTSYSTEM_H

#pragma once

#include <memory>

#include "car/system/movement/controller/AbstractMovementController.h"

using namespace car::system::movement::controller;

namespace car::system::movement
{
    class MovementSystem
    {
    public:
        MovementSystem(std::unique_ptr<AbstractMovementController> movement_controller) :
    movement_controller(std::move(movement_controller)){};

        void initialize()
        {
            this->movement_controller->initialize();
        }

        void start()
        {
        }

        void stop()
        {
            this->movement_controller->stop();
        }

        void terminate()
        {
            this->movement_controller->terminate();
        }

#pragma region Wheels
        void setRearWheelsSpeed(const int speed) const
        {
            this->movement_controller->setRearWheelsSpeed(speed);
        }

        void setRearLeftWheelSpeed(const int speed) const
        {
            this->movement_controller->setRearLeftWheelSpeed(speed);
        }

        void setRearRightWheelSpeed(const int speed) const
        {
            this->movement_controller->setRearRightWheelSpeed(speed);
        }

        void setFrontWheelsAngle(const float angle) const
        {
            this->movement_controller->setFrontWheelsAngle(angle);
        }

        void setCameraServo1Angle(const float angle) const
        {
            this->movement_controller->setCameraServo1Angle(angle);
        }

        void setCameraServo2Angle(const float angle) const
        {
            this->movement_controller->setCameraServo2Angle(angle);
        }

        void setRearWheelsDirectionToForward() const
        {
            this->movement_controller->setRearWheelsDirectionToForward();
        }

        void setRearLeftWheelDirectionToForward() const
        {
            this->movement_controller->setRearLeftWheelDirectionToForward();
        }

        void setRearRightWheelDirectionToForward() const
        {
            this->movement_controller->setRearRightWheelDirectionToForward();
        }

        void setRearWheelsDirectionToBackward() const
        {
            this->movement_controller->setRearWheelsDirectionToBackward();
        }

        void setRearLeftWheelDirectionToBackward() const
        {
            this->movement_controller->setRearLeftWheelDirectionToBackward();
        }

        void setRearRightWheelDirectionToBackward() const
        {
            this->movement_controller->setRearRightWheelDirectionToBackward();
        }
#pragma endregion

        ~MovementSystem(){};

    private:
        std::unique_ptr<AbstractMovementController> movement_controller;
    };
};

#endif
```

13.44 common/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

13.45 common/src/car/system/device/CameraDevice.cpp File Reference

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

Namespaces

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

13.46 common/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

13.47 common/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

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

13.49 common/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

13.50 common/src/car/system/movement/devices/RearWheel.cpp File Reference

13.51 common/src/car/system/movement/devices/Servo.cpp File Reference

13.52 common/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

13.52.1 Function Documentation

13.52.1.1 main()

```
int main ( )
```

13.52.1.2 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

13.52.1.3 setAngle()

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

13.52.1.4 setAngleToAnalog()

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

13.52.2 Variable Documentation

13.52.2.1 offset

```
int offset = 0
```

13.53 common/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 ()

13.53.1 Function Documentation

13.53.1.1 main()

```
int main ( )
```

13.53.1.2 test()

```
void test ( )
```

13.54 daemon/install/README.md File Reference

13.55 daemon/README.md File Reference

13.56 README.md File Reference

13.57 tui/README.md File Reference

13.58 repository/packages/t/tb6612/tb6612/include/TB6612.h File Reference

```
#include "pigpio.h"
#include "pigpiod_if2.h"
```

Classes

- class TB6612

13.59 TB6612.h

Go to the documentation of this file.

```
#ifndef TB6612_HPP
#define TB6612_HPP

#pragma once

// Made with the help of ChatGPT

#include "pigpio.h"
#include "pigpiod_if2.h"

class TB6612
{
public:
    TB6612(int motor_pin, int pwm_pin);

    void setPWM(int value);

    void forward();

    void backward();

    void stop();

    void setOffset(bool offset);

    const int &getMotorPin() const;

    const int &getPWMPin() const;

private:
    const int motor_pin;
    const int pwm_pin;
    bool offset = true;
};

#endif
```

13.60 repository/packages/t/tb6612/tb6612/src/TB6612.cpp File Reference

```
#include "TB6612.h"
```

13.61 SETUP.md File Reference

13.62 tui/SETUP.md File Reference

13.63 tui/src/car/configuration/JsonConfiguration.cxx File Reference

```
#include <iostream>
#include <fstream>
#include <variant>
#include <optional>
#include <rapidjson/document.h>
#include <rapidjson/istreamwrapper.h>
#include <spdlog/spdlog.h>
#include <fmt/format.h>
#include <tl/expected.hpp>
#include "car/configuration/Configuration.h"
```

Classes

- class car::configuration::JsonConfiguration

Namespaces

- namespace car
- namespace car::configuration

Macros

- #define JSONCONFIGURATION_CXX

13.63.1 Macro Definition Documentation

13.63.1.1 JSONCONFIGURATION_CXX

```
#define JSONCONFIGURATION_CXX
```

13.64 tui/src/car/display/console/CarConsole.cpp File Reference

```
#include "CarConsole.h"
#include <ftxui/component/component.hpp>
#include <ftxui/component/screen_interactive.hpp>
#include <ftxui/dom/elements.hpp>
#include <ftxui/component/loop.hpp>
#include <nod/nod.hpp>
```

Namespaces

- namespace car
- namespace car::display
- namespace car::display::console

13.65 tui/src/car/display/console/CarConsole.h File Reference

```
#include <memory>
#include "car/system/CarSystem.h"
#include "car/system/logging/VectorSink.h"
#include "screen/MainScreen.cxx"
#include "screen/SettingsScreen.cxx"
#include "screen/LoggingScreen.cxx"
```

Classes

- class car::display::console::CarConsole

Namespaces

- namespace car
- namespace car::display
- namespace car::display::console

13.66 CarConsole.h

Go to the documentation of this file.

```
#ifndef CARCONSOLE_H
#define CARCONSOLE_H

#pragma once

#include <memory>

#include "car/system/CarSystem.h"
#include "car/system/logging/VectorSink.h"

#include "screen/MainScreen.cxx"
#include "screen/SettingsScreen.cxx"
#include "screen/LoggingScreen.cxx"

using namespace car::system;
using namespace car::display::console::screen;

namespace car::display::console
{
    class CarConsole
    {
    public:
        CarConsole(std::shared_ptr<CarSystem> car_system, std::shared_ptr<JsonConfiguration>
    json_configuration, std::shared_ptr<logging::vector_sink_mt> vector_sink);

        void initialize();

        void run();

        void terminate();

    private:
        std::shared_ptr<CarSystem> car_system;
        std::shared_ptr<JsonConfiguration> json_configuration;
        std::shared_ptr<logging::vector_sink_mt> vector_sink;
    };
}

#endif
```

13.67 tui/src/car/display/console/component/debug/DebugEnabler.cxx File Reference

```
#include <nod/nod.hpp>
#include <ftxui/component/component.hpp>
```

Classes

- class car::display::console::component::debug::DebugEnabler

Namespaces

- namespace car
- namespace car::display
- namespace car::display::console
- namespace car::display::console::component
- namespace car::display::console::component::debug

Macros

- #define DEBUGENABLER_CXX

13.67.1 Macro Definition Documentation

13.67.1.1 DEBUGENABLER_CXX

```
#define DEBUGENABLER_CXX
```

13.68 tui/src/car/display/console/component/debug/DebugLidarCheckbox.cxx File Reference

```
#include <nod/nod.hpp>
#include <ftxui/component/component.hpp>
```

Classes

- class car::display::console::component::debug::DebugLidarCheckbox

Namespaces

- namespace car
- namespace car::display
- namespace car::display::console
- namespace car::display::console::component
- namespace car::display::console::component::debug

Macros

- #define DEBUGLIDARCHECKBOX_CXX

13.68.1 Macro Definition Documentation

13.68.1.1 DEBUGLIDARCHECKBOX_CXX

```
#define DEBUGLIDARCHECKBOX_CXX
```

13.69 tui/src/car/display/console/component/debug/DebugMessaging↩ Textbox.cxx File Reference

```
#include <nod/nod.hpp>
#include <ftxui/component/component.hpp>
```

Classes

- class car::display::console::component::debug::DebugMessagingTextbox

Namespaces

- namespace car
- namespace car::display
- namespace car::display::console
- namespace car::display::console::component
- namespace car::display::console::component::debug

Macros

- #define DEBUGMESSAGINGTEXTBOX_CXX

13.69.1 Macro Definition Documentation

13.69.1.1 DEBUGMESSAGINGTEXTBOX_CXX

```
#define DEBUGMESSAGINGTEXTBOX_CXX
```

13.70 tui/src/car/display/console/component/debug/DebugMovement↩ Renderer.cxx File Reference

```
#include <nod/nod.hpp>
#include <ftxui/component/component.hpp>
```

Classes

- class car::display::console::component::debug::DebugMovementRenderer

Namespaces

- namespace car
- namespace car::display
- namespace car::display::console
- namespace car::display::console::component
- namespace car::display::console::component::debug

Macros

- #define DEBUGMOVEMENTRENDERER_CXX

13.70.1 Macro Definition Documentation

13.70.1.1 DEBUGMOVEMENTRENDERER_CXX

```
#define DEBUGMOVEMENTRENDERER_CXX
```

13.71 tui/src/car/display/console/component/main/ConnectButton.cxx File Reference

```
#include <ftxui/component/component.hpp>
#include "car/system/CarSystem.h"
```

Classes

- class car::display::console::component::main::ConnectButton

Namespaces

- namespace car
- namespace car::display
- namespace car::display::console
- namespace car::display::console::component
- namespace car::display::console::component::main

Macros

- #define CONNECTBUTTON_CXX

13.71.1 Macro Definition Documentation

13.71.1.1 CONNECTBUTTON_CXX

```
#define CONNECTBUTTON_CXX
```

13.72 tui/src/car/display/console/component/main/MainErrorModal.cxx File Reference

```
#include <ftxui/component/component.hpp>
#include "car/system/CarSystem.h"
```

Classes

- class car::display::console::component::main::MainErrorModal

Namespaces

- namespace car
- namespace car::display
- namespace car::display::console
- namespace car::display::console::component
- namespace car::display::console::component::main

Macros

- #define MAINERRORMODAL_CXX

13.72.1 Macro Definition Documentation

13.72.1.1 MAINERRORMODAL_CXX

```
#define MAINERRORMODAL_CXX
```

13.73 tui/src/car/display/console/component/main/MainExitModal.cxx File Reference

```
#include <ftxui/component/component.hpp>
#include "car/system/CarSystem.h"
```

Classes

- class car::display::console::component::main::MainExitModal

Namespaces

- namespace car
- namespace car::display
- namespace car::display::console
- namespace car::display::console::component
- namespace car::display::console::component::main

Macros

- #define MAINEXITMODAL_CXX

13.73.1 Macro Definition Documentation

13.73.1.1 MAINEXITMODAL_CXX

```
#define MAINEXITMODAL_CXX
```

13.74 tui/src/car/display/console/component/settings/SettingsEdit↩ Config.cxx File Reference

```
#include <ftxui/component/component.hpp>
#include "car/system/CarSystem.h"
#include "../../../../configuration/JsonConfiguration.cxx"
```

Classes

- class car::display::console::component::settings::SettingsEditConfig

Namespaces

- namespace car
- namespace car::display
- namespace car::display::console
- namespace car::display::console::component
- namespace car::display::console::component::settings

Macros

- #define SETTINGSEDITCONFIG_CXX

13.74.1 Macro Definition Documentation

13.74.1.1 SETTINGSEDITCONFIG_CXX

```
#define SETTINGSEDITCONFIG_CXX
```

13.75 tui/src/car/display/console/screen/LoggingScreen.cxx File Reference

```
#include <ftxui/component/component.hpp>
#include <spdlog/spdlog.h>
#include "car/system/logging/VectorSink.h"
```

Classes

- class car::display::console::screen::LoggingScreen

Namespaces

- namespace car
- namespace car::display
- namespace car::display::console
- namespace car::display::console::screen

Macros

- #define LOGGINGSCREEN_CXX

13.75.1 Macro Definition Documentation

13.75.1.1 LOGGINGSCREEN_CXX

```
#define LOGGINGSCREEN_CXX
```

13.76 tui/src/car/display/console/screen/MainScreen.cxx File Reference

```
#include <memory>
#include <ftxui/component/component.hpp>
#include "car/system/CarSystem.h"
#include "../component/main/ConnectButton.cxx"
#include "../component/main/MainExitModal.cxx"
#include "../component/main/MainErrorModal.cxx"
```

Classes

- class car::display::console::screen::MainScreen

Namespaces

- namespace car
- namespace car::display
- namespace car::display::console
- namespace car::display::console::screen

Macros

- #define MAINSCREEN_CXX

13.76.1 Macro Definition Documentation

13.76.1.1 MAINSCREEN_CXX

```
#define MAINSCREEN_CXX
```

13.77 tui/src/car/display/console/screen/SettingsScreen.cxx File Reference

```
#include <memory>
#include <ftxui/component/component.hpp>
#include "car/system/CarSystem.h"
#include "../../../configuration/JsonConfiguration.cxx"
#include "../component/settings/SettingsEditConfig.cxx"
#include "../component/debug/DebugEnabler.cxx"
#include "../component/debug/DebugLidarCheckbox.cxx"
#include "../component/debug/DebugMovementRenderer.cxx"
#include "../component/debug/DebugMessagingTextbox.cxx"
```

Classes

- class car::display::console::screen::SettingsScreen

Namespaces

- namespace car
- namespace car::display
- namespace car::display::console
- namespace car::display::console::screen

Macros

- #define SETTINGSSCREEN_CXX

13.77.1 Macro Definition Documentation

13.77.1.1 SETTINGSSCREEN_CXX

```
#define SETTINGSSCREEN_CXX
```

Index