SRM Vision Remastered

0.4 Alpha

Generated by Doxygen 1.8.17

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 coordinate Namespace Reference

Namespaces

- **transform**

Typedefs

- typedef Eigen::Vector3d **TranslationVector**
- typedef Eigen::Vector3d **RotationVector**
- typedef Eigen::Matrix< double, 3, 1 > **TranslationMatrix**
- typedef Eigen::Matrix3d **RotationMatrix**
- typedef Eigen::Quaternionf **Quaternion**

5.1.1 Detailed Description

Coordinate transformer header.

Author

trantuan-20048607

Date

2022.1.30

Attention

It's recommended to include **battlefield.h** (p. 168) for complete function.

5.1.2 Typedef Documentation

5.1.2.1 Quaternion

```
typedef Eigen::Quaternionf coordinate::Quaternion
```

Definition at line 24 of file coordinate.h.

5.1.2.2 RotationMatrix

```
typedef Eigen::Matrix3d coordinate::RotationMatrix
```

Definition at line 22 of file coordinate.h.

5.1.2.3 RotationVector

```
typedef Eigen::Vector3d coordinate::RotationVector
```

Definition at line 19 of file coordinate.h.

5.1.2.4 TranslationMatrix

```
typedef Eigen::Matrix<double, 3, 1> coordinate::TranslationMatrix
```

Definition at line 21 of file coordinate.h.

5.1.2.5 TranslationVector

```
typedef Eigen::Vector3d coordinate::TranslationVector
```

Definition at line 18 of file coordinate.h.

5.2 coordinate::transform Namespace Reference

Functions

- **RotationMatrix** **QuaternionToRotationMatrix** (const **Quaternion** &quaternion)
- **TranslationVector** **CameraToWorld** (const **TranslationVector** &tv_cam, const **RotationMatrix** &rm_imu, const **TranslationMatrix** &tm_cam_to_imu, const **RotationMatrix** &rm_cam_to_imu)
- **TranslationVector** **WorldToCamera** (const **TranslationVector** &tv_world, const **RotationMatrix** &rm_↩ imu_to_world, const **TranslationMatrix** &tm_cam_to_imu, const **RotationMatrix** &rm_cam_to_imu)

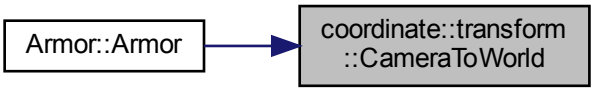
5.2.1 Function Documentation

5.2.1.1 CameraToWorld()

```
TranslationVector coordinate::transform::CameraToWorld (
            const  TranslationVector & tv_cam,
            const  RotationMatrix & rm_imu,
            const  TranslationMatrix & tm_cam_to_imu,
            const  RotationMatrix & rm_cam_to_imu )  [inline]
```

Definition at line 84 of file coordinate.h.

Here is the caller graph for this function:

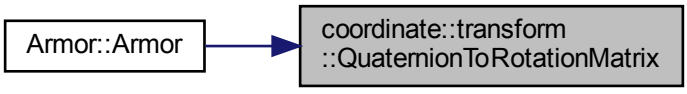


5.2.1.2 QuaternionToRotationMatrix()

```
RotationMatrix coordinate::transform::QuaternionToRotationMatrix (
            const  Quaternion & quaternion )  [inline]
```

Definition at line 33 of file coordinate.h.

Here is the caller graph for this function:



5.2.1.3 WorldToCamera()

```
TranslationVector coordinate::transform::WorldToCamera (
            const  TranslationVector & tv_world,
            const  RotationMatrix & rm_imu_to_world,
            const  TranslationMatrix & tm_cam_to_imu,
            const  RotationMatrix & rm_cam_to_imu )  [inline]
```

Definition at line 92 of file coordinate.h.

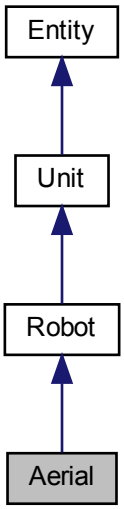
Chapter 6

Class Documentation

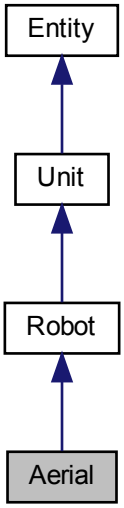
6.1 Aerial Class Reference

```
#include <aerial.h>
```

Inheritance diagram for Aerial:



Collaboration diagram for Aerial:



Public Member Functions

- **Aerial** (**Colors** color, double health, **RobotTypes** type= **kAerial**)

Additional Inherited Members

6.1.1 Detailed Description

Aerial (p. 13) robot header.

Author

trantuan-20048607

Date

2022.1.28

Attention

It's recommended to include **battlefield.h** (p. 168) for complete function.

Definition at line 13 of file aerial.h.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Aerial()

```
Aerial::Aerial (
              Colors color,
             double health,
              RobotTypes type =  kAerial )  [inline]
```

Definition at line 15 of file aerial.h.

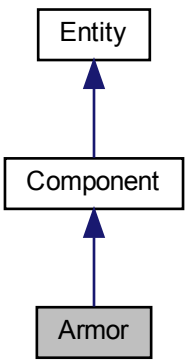
The documentation for this class was generated from the following file:

- modules/digital-twin/robots/ **aerial.h**

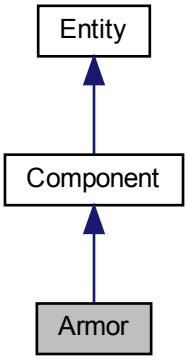
6.2 Armor Class Reference

```
#include <armor.h>
```

Inheritance diagram for Armor:



Collaboration diagram for Armor:



Public Member Functions

- **Armor** (const **bbox_t** &box, const cv::Mat &intrinsic_mat, const cv::Mat &distortion_mat, const Eigen::↩ Quaternionf &quaternion)

Additional Inherited Members

6.2.1 Detailed Description

Armor (p. 15) definition header.

Author

trantuan-20048607

Date

2022.1.28

Attention

It's recommended to include **battlefield.h** (p. 168) for complete function.

Definition at line 16 of file armor.h.

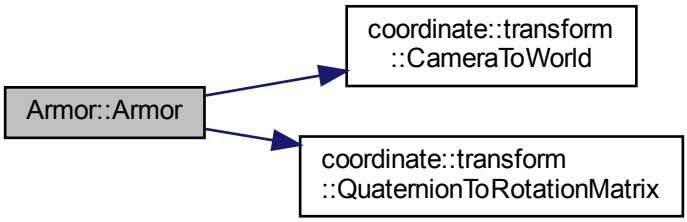
6.2.2 Constructor & Destructor Documentation

6.2.2.1 Armor()

```
Armor::Armor (
            const  bbox_t & box,
            const cv::Mat & intrinsic_mat,
            const cv::Mat & distortion_mat,
            const Eigen::Quaternionf & quaternion )  [inline]
```

Definition at line 36 of file armor.h.

Here is the call graph for this function:



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

- modules/digital-twin/components/ **armor.h**

6.3 ArmorDetector Class Reference

`#include <detector_armor.h>`

Public Member Functions

- **ArmorDetector** ()
- ~**ArmorDetector** ()
- void **Initialize** (const std::string &onnx_file)

  Load and initialize model.
- std::vector< **bbox_t** > **operator()** (const cv::Mat &image) const

  Run detection model.

6.3.1 Detailed Description

Armor (p. 15) detector header.

Author

anonymity, screw-44

Date

2022.1.28

Definition at line 14 of file detector_armor.h.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 ArmorDetector()

```
ArmorDetector::ArmorDetector ( )  [inline]
```

Definition at line 19 of file detector_armor.h.

6.3.2.2 ~ArmorDetector()

```
ArmorDetector::~ArmorDetector ( )
```

Definition at line 102 of file detector_armor.cpp.

6.3.3 Member Function Documentation

6.3.3.1 Initialize()

```
void ArmorDetector::Initialize (
            const std::string & onnx_file )
```

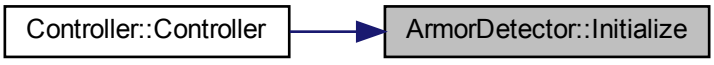
Load and initialize model.

Parameters

in	*onnx_file*	ONNX file path.

Definition at line 72 of file detector_armor.cpp.

Here is the caller graph for this function:



6.3.3.2 operator()()

```
std::vector<  bbox_t > ArmorDetector::operator() (
            const cv::Mat & image ) const
```

Run detection model.

Parameters

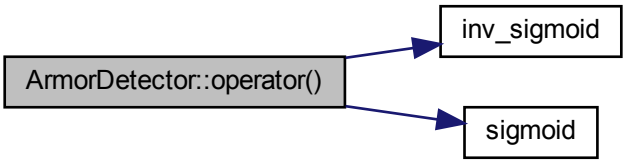
in	*image*	Input image.

Returns

4-point structures in a vector.

Definition at line 209 of file detector_armor.cpp.

Here is the call graph for this function:



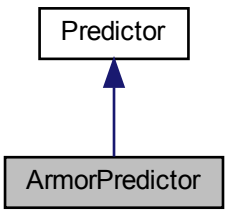
The documentation for this class was generated from the following files:

- modules/detector-armor/ **detector_armor.h**
- modules/detector-armor/ **detector_armor.cpp**

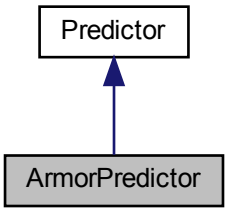
6.4 ArmorPredictor Class Reference

`#include <predictor_armor.h>`

Inheritance diagram for ArmorPredictor:



Collaboration diagram for ArmorPredictor:



Classes

- struct **ArmorPredictorNode**

Public Member Functions

- **ArmorPredictor** (**Entity::Colors** enemy_color, bool debug)
- ~**ArmorPredictor** ()=default

6.4.1 Detailed Description

Kalman predictor header.

Author

trantuan-20048607

Date

2022.1.30

Definition at line 13 of file predictor_armor.h.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 ArmorPredictor()

```
ArmorPredictor::ArmorPredictor (
            Entity::Colors enemy_color,
            bool debug )  [inline]
```

Definition at line 67 of file predictor_armor.h.

6.4.2.2 ~ArmorPredictor()

```
ArmorPredictor::~ArmorPredictor ( )  [default]
```

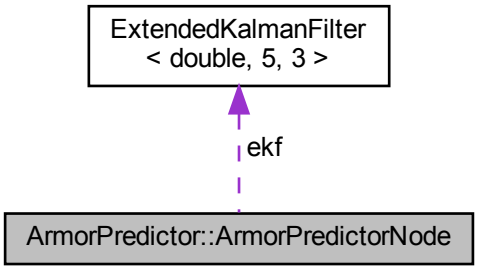
The documentation for this class was generated from the following file:

- modules/predictor-armor/ **predictor_armor.h**

6.5 ArmorPredictor::ArmorPredictorNode Struct Reference

```
#include <predictor_armor.h>
```

Collaboration diagram for ArmorPredictor::ArmorPredictorNode:



Public Member Functions

- **ArmorPredictorNode** ()
- **ArmorPredictorNode** (std::shared_ptr< **Armor** > _armor)
- **ArmorPredictorNode** & **operator=** (const **ArmorPredictorNode** &rhs)

Public Attributes

- std::shared_ptr< **Armor** > **armor**
- **ExtendedKalmanFilter**< double, 5, 3 > **ekf**
- bool **need_update** = false
- bool **need_init** = false
- double **last_yaw** = 0
- double **last_pitch** = 0
- float **yaw**
- float **pitch**
- float **yaw_speed**
- float **pitch_speed**
- **coordinate::TranslationVector** **tv_imu**
- bool **long_distance**
- int **lasting_time**

6.5.1 Detailed Description

Definition at line 15 of file predictor_armor.h.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 ArmorPredictorNode() [1/2]

```
ArmorPredictor::ArmorPredictorNode::ArmorPredictorNode ( )  [inline]
```

Definition at line 25 of file predictor_armor.h.

6.5.2.2 ArmorPredictorNode() [2/2]

```
ArmorPredictor::ArmorPredictorNode::ArmorPredictorNode (
            std::shared_ptr< Armor > _armor )  [inline], [explicit]
```

Definition at line 36 of file predictor_armor.h.

6.5.3 Member Function Documentation

6.5.3.1 operator=()

```
ArmorPredictorNode& ArmorPredictor::ArmorPredictorNode::operator= (
            const ArmorPredictorNode & rhs )  [inline]
```

Definition at line 49 of file predictor_armor.h.

6.5.4 Member Data Documentation

6.5.4.1 armor

```
std::shared_ptr< Armor> ArmorPredictor::ArmorPredictorNode::armor
```

Definition at line 16 of file predictor_armor.h.

6.5.4.2 ekf

```
ExtendedKalmanFilter<double, 5, 3> ArmorPredictor::ArmorPredictorNode::ekf
```

Definition at line 17 of file predictor_armor.h.

6.5.4.3 last_pitch

```
double ArmorPredictor::ArmorPredictorNode::last_pitch = 0
```

Definition at line 19 of file predictor_armor.h.

6.5.4.4 last_yaw

```
double ArmorPredictor::ArmorPredictorNode::last_yaw = 0
```

Definition at line 19 of file predictor_armor.h.

6.5.4.5 lasting_time

```
int ArmorPredictor::ArmorPredictorNode::lasting_time
```

Definition at line 23 of file predictor_armor.h.

6.5.4.6 long_distance

```
bool ArmorPredictor::ArmorPredictorNode::long_distance
```

Definition at line 22 of file predictor_armor.h.

6.5.4.7 need_init

```
bool ArmorPredictor::ArmorPredictorNode::need_init = false
```

Definition at line 18 of file predictor_armor.h.

6.5.4.8 need_update

```
bool ArmorPredictor::ArmorPredictorNode::need_update = false
```

Definition at line 18 of file predictor_armor.h.

6.5.4.9 pitch

```
float ArmorPredictor::ArmorPredictorNode::pitch
```

Definition at line 20 of file predictor_armor.h.

6.5.4.10 pitch_speed

```
float ArmorPredictor::ArmorPredictorNode::pitch_speed
```

Definition at line 20 of file predictor_armor.h.

6.5.4.11 tv_imu

```
coordinate::TranslationVector ArmorPredictor::ArmorPredictorNode::tv_imu
```

Definition at line 21 of file predictor_armor.h.

6.5.4.12 yaw

```
float ArmorPredictor::ArmorPredictorNode::yaw
```

Definition at line 20 of file predictor_armor.h.

6.5.4.13 yaw_speed

```
float ArmorPredictor::ArmorPredictorNode::yaw_speed
```

Definition at line 20 of file predictor_armor.h.

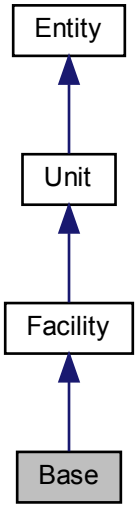
The documentation for this struct was generated from the following file:

- modules/predictor-armor/ **predictor_armor.h**

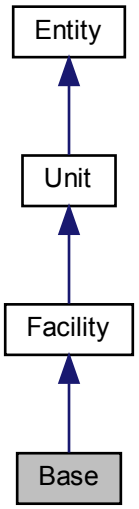
6.6 Base Class Reference

`#include <base.h>`

Inheritance diagram for Base:



Collaboration diagram for Base:



Public Member Functions

- **Base** (**Colors** color, double health, **FacilityTypes** type= **kBase**)

Additional Inherited Members

6.6.1 Detailed Description

Base (p. 25) definition header.

Author

trantuan-20048607

Date

2022.1.28

Attention

It's recommended to include **battlefield.h** (p. 168) for complete function.

Definition at line 13 of file base.h.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 Base()

```
Base::Base (
             Colors color,
            double health,
             FacilityTypes type =  kBase )  [inline]
```

Definition at line 15 of file base.h.

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

- modules/digital-twin/facilities/ **base.h**

6.7 Battlefield Class Reference

Battlefield (p. 26) simulation data model.

```
#include <battlefield.h>
```

Public Member Functions

- **__ATTR_READER__** (quaternion_, Quaternion)
- **Battlefield** ()
- **Battlefield** (uint64_t time_stamp, const Eigen::Quaternionf &quaternion, const std::vector< **Armor** > &armors)

  Constructor with complete battlefield information.

6.7.1 Detailed Description

Battlefield (p. 26) simulation data model.

Definition at line 27 of file battlefield.h.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 Battlefield() [1/2]

```
Battlefield::Battlefield ( )  [inline]
```

Definition at line 37 of file battlefield.h.

6.7.2.2 Battlefield() [2/2]

```
Battlefield::Battlefield (
            uint64_t time_stamp,
            const Eigen::Quaternionf & quaternion,
            const std::vector< Armor > & armors )  [inline]
```

Constructor with complete battlefield information.

Parameters

time_stamp	Time stamp from its source for tracking.
armors	**Armor** (p. 15) sources.

Store last battlefield data for eliminating shakes.

Definition at line 88 of file battlefield.h.

6.7.3 Member Function Documentation

6.7.3.1 __ATTR_READER__()

```
Battlefield::__ATTR_READER__ (
             quaternion_ ,
             Quaternion  )
```

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

- modules/digital-twin/ **battlefield.h**

6.8 bbox_t Struct Reference

4-point model structure for detection result.

`#include <bbox_t.h>`

Public Member Functions

- bool **operator==** (const **bbox_t** &bbox) const
- bool **operator!=** (const **bbox_t** &bbox) const

Public Attributes

- cv::Point2f **points** [4]
- float **confidence**
- int **color**

  0: blue, 1: red, 2: purple, 3: grey.
- int **id**

  0: sentry, 1-5: cars, 6: base.

6.8.1 Detailed Description

4-point model structure for detection result.

4-point data model header.

Author

anonymity

Date

2022.1.28

Definition at line 15 of file bbox_t.h.

6.8.2 Member Function Documentation

6.8.2.1 operator"!=()

```
bool bbox_t::operator!= (
            const  bbox_t & bbox ) const  [inline]
```

Definition at line 30 of file bbox_t.h.

6.8.2.2 operator==()

```
bool bbox_t::operator== (
            const  bbox_t & bbox ) const  [inline]
```

Definition at line 21 of file bbox_t.h.

6.8.3 Member Data Documentation

6.8.3.1 color

```
int bbox_t::color
```

0: blue, 1: red, 2: purple, 3: grey.

Definition at line 18 of file bbox_t.h.

6.8.3.2 confidence

```
float bbox_t::confidence
```

Definition at line 17 of file bbox_t.h.

6.8.3.3 id

```
int bbox_t::id
```

0: sentry, 1-5: cars, 6: base.

Definition at line 19 of file bbox_t.h.

6.8.3.4 points

```
cv::Point2f bbox_t::points[4]
```

Definition at line 16 of file bbox_t.h.

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

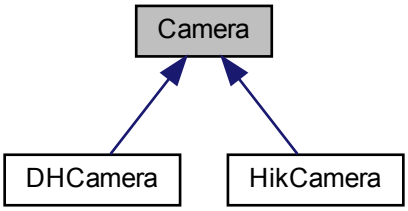
- modules/data-structure/ **bbox_t.h**

6.9 Camera Class Reference

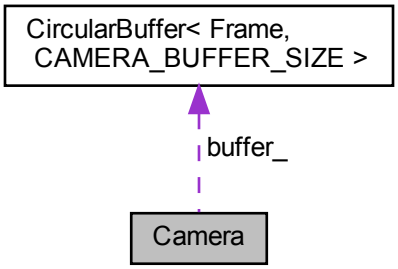
Camera (p. 30) base class.

```
#include <camera_base.h>
```

Inheritance diagram for Camera:



Collaboration diagram for Camera:



Public Member Functions

- **Camera** ()
- virtual ~**Camera** ()=default
- virtual bool **OpenCamera** (const std::string &serial_number, const std::string &config_file)=0

  Open a camera.
- virtual bool **CloseCamera** ()=0

  Close the opened camera.
- virtual bool **GetFrame** (**Frame** &frame)=0

  Get a frame with image and time stamp from internal image buffer.
- virtual bool **StartStream** ()=0

  Run the stream.
- virtual bool **StopStream** ()=0

  Stop the stream.
- virtual bool **IsConnected** ()=0

  Check if current device is connected.
- virtual bool **ImportConfigurationFile** (const std::string &file_path)=0

  Import current config to specified file.
- virtual bool **ExportConfigurationFile** (const std::string &file_path)=0

  Export current config to specified file.
- virtual bool **SetExposureTime** (uint32_t exposure_time)=0

  Set exposure time.
- virtual bool **SetGainValue** (float gain)=0

  Set gain value.

Protected Attributes

- std::string **serial_number_**

  Serial *(p. 136) number.*
- bool **stream_running_**

  Stream running flag.
- pthread_t **daemon_thread_id_**

  Daemon thread id.
- bool **stop_daemon_thread_flag_**

  Flag to stop daemon thread.
- **CircularBuffer**< **Frame**, **CAMERA_BUFFER_SIZE** > **buffer_**

  A ring buffer to store images.

6.9.1 Detailed Description

Camera (p. 30) base class.

Note

You cannot directly construct objects.
Instead, find camera types in subclass documents, include **camera_factory.h** (p. 145) and use **Camera↩ Factory::Instance()** (p. 39)::CreateCamera(camera_type_name).

Definition at line 24 of file camera_base.h.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 Camera()

```
Camera::Camera ( )  [inline]
```

Definition at line 26 of file camera_base.h.

6.9.2.2 ~Camera()

```
virtual Camera::~Camera ( )  [virtual], [default]
```

6.9.3 Member Function Documentation

6.9.3.1 CloseCamera()

```
virtual bool Camera::CloseCamera ( )  [pure virtual]
```

Close the opened camera.

Returns

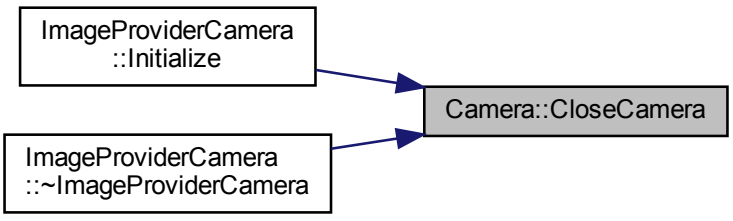
Whether the camera is closed normally.

Attention

No matter what is returned, the camera handle will be unreachable.

Implemented in **DHCamera** (p. 68), and **HikCamera** (p. 93).

Here is the caller graph for this function:



6.9.3.2 ExportConfigurationFile()

```
virtual bool Camera::ExportConfigurationFile (
            const std::string & file_path )  [pure virtual]
```

Export current config to specified file.

Parameters

in	*file_path*	File path.

Returns

Whether config file is saved.

Implemented in **DHCamera** (p. 68), and **HikCamera** (p. 93).

6.9.3.3 GetFrame()

```
virtual bool Camera::GetFrame (
             Frame & frame )  [pure virtual]
```

Get a frame with image and time stamp from internal image buffer.

Parameters

out	*frame*	Acquired frame will be stored here.

Returns

Whether buffer is not empty, or if you can successfully get an frame.

Implemented in **DHCamera** (p. 69), and **HikCamera** (p. 94).

6.9.3.4 ImportConfigurationFile()

```
virtual bool Camera::ImportConfigurationFile (
            const std::string & file_path )  [pure virtual]
```

Import current config to specified file.

Parameters

in	*file_path*	File path.

Returns

Whether config file is imported.

Implemented in **DHCamera** (p. 69), and **HikCamera** (p. 94).

6.9.3.5 IsConnected()

```
virtual bool Camera::IsConnected ( )  [pure virtual]
```

Check if current device is connected.

Returns

Whether current device is connected.

Implemented in **DHCamera** (p. 70), and **HikCamera** (p. 95).

6.9.3.6 OpenCamera()

```
virtual bool Camera::OpenCamera (
            const std::string & serial_number,
            const std::string & config_file )  [pure virtual]
```

Open a camera.

Parameters

in	*serial_number*	**Serial** (p. 136) number of the camera you wanna open.
in	*config_file*	Will load config from this file.

Returns

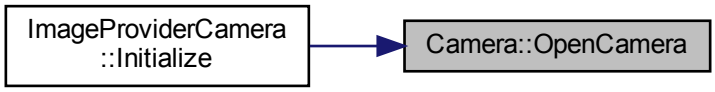
Whether the camera is opened.

Note

Another try after failures is allowed.

Implemented in **DHCamera** (p. 70), and **HikCamera** (p. 95).

Here is the caller graph for this function:



6.9.3.7 SetExposureTime()

```
virtual bool Camera::SetExposureTime (
            uint32_t exposure_time )  [pure virtual]
```

Set exposure time.

Parameters

exposure_time	Exposure time, automatically converted to corresponding data type.

Returns

Whether exposure time is set.

Implemented in **DHCamera** (p. 71), and **HikCamera** (p. 96).

6.9.3.8 SetGainValue()

```
virtual bool Camera::SetGainValue (
            float gain )  [pure virtual]
```

Set gain value.

Parameters

gain	Gain value, automatically converted to corresponding data type.

Returns

Whether gain value is set.

Implemented in **DHCamera** (p. 72), and **HikCamera** (p. 96).

6.9.3.9 StartStream()

```
virtual bool Camera::StartStream ( )  [pure virtual]
```

Run the stream.

Returns

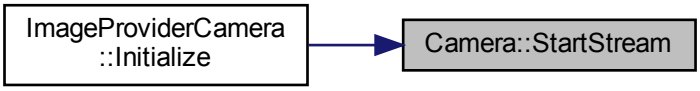
Whether stream is started normally.

Attention

This function will return false when stream is already started or camera is not opened.

Implemented in **DHCamera** (p. 72), and **HikCamera** (p. 96).

Here is the caller graph for this function:



6.9.3.10 StopStream()

```
virtual bool Camera::StopStream ( )  [pure virtual]
```

Stop the stream.

Returns

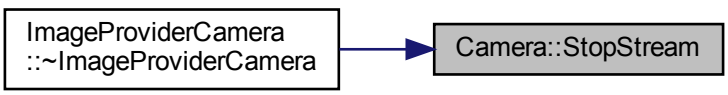
Whether stream is stopped normally.

Attention

This function will return false when stream is not started or camera is not opened.

Implemented in **DHCamera** (p. 72), and **HikCamera** (p. 97).

Here is the caller graph for this function:



6.9.4 Member Data Documentation

6.9.4.1 buffer_

CircularBuffer< **Frame**, **CAMERA_BUFFER_SIZE**> Camera::buffer_ [protected]

A ring buffer to store images.

Definition at line 108 of file camera_base.h.

6.9.4.2 daemon_thread_id_

pthread_t Camera::daemon_thread_id_ [protected]

Daemon thread id.

Definition at line 106 of file camera_base.h.

6.9.4.3 serial_number_

std::string Camera::serial_number_ [protected]

Serial (p. 136) number.

Definition at line 104 of file camera_base.h.

6.9.4.4 stop_daemon_thread_flag_

bool Camera::stop_daemon_thread_flag_ [protected]

Flag to stop daemon thread.

Definition at line 107 of file camera_base.h.

6.9.4.5 stream_running_

```
bool Camera::stream_running_  [protected]
```

Stream running flag.

Definition at line 105 of file camera_base.h.

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

- modules/camera-base/ **camera_base.h**

6.10 CameraFactory Class Reference

Singleton camera factory.

```
#include <camera_factory.h>
```

Public Member Functions

- **CameraFactory** (const **CameraFactory** &)=delete
- **CameraFactory** & **operator=** (const **CameraFactory** &)=delete
- void **RegisterCamera** (const std::string &camera_type_name, **CameraRegistryBase** *registry)
  Register a camera type.
- **Camera** * **CreateCamera** (const std::string &camera_type_name)
  Create a camera whose type is registered to factory.

Static Public Member Functions

- static **CameraFactory** & **Instance** ()
  Get the only instance_ of camera factory.

6.10.1 Detailed Description

Singleton camera factory.

For Singleton pattern, refer to https://en.wikipedia.org/wiki/Singleton_pattern.
For Factory pattern, refer to https://en.wikipedia.org/wiki/Factory_method_pattern.

Warning

Camera (p. 30) factory will not check whether CameraType is really subclass of **Camera** (p. 30) base class. (Thus, you should ensure that all callings of **CameraRegistry** (p. 41) constructor are completely under control.)

Definition at line 41 of file camera_factory.h.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 CameraFactory()

```
CameraFactory::CameraFactory (
            const  CameraFactory &  )  [delete]
```

6.10.3 Member Function Documentation

6.10.3.1 CreateCamera()

```
Camera* CameraFactory::CreateCamera (
            const std::string & camera_type_name )  [inline]
```

Create a camera whose type is registered to factory.

Parameters

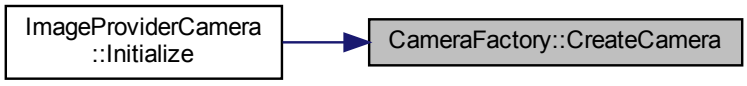
in	*camera_type_name*	Type name of camera.

Returns

A pointer to crated camera.

Definition at line 71 of file camera_factory.h.

Here is the caller graph for this function:



6.10.3.2 Instance()

```
static CameraFactory& CameraFactory::Instance ( ) [inline], [static]
```

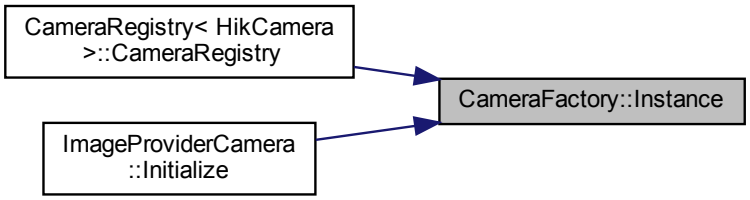
Get the only instance_ of camera factory.

Returns

A camera factory object.

Definition at line 51 of file camera_factory.h.

Here is the caller graph for this function:



6.10.3.3 operator=()

```
CameraFactory& CameraFactory::operator= (
            const CameraFactory & ) [delete]
```

6.10.3.4 RegisterCamera()

```
void CameraFactory::RegisterCamera (
            const std::string & camera_type_name,
             CameraRegistryBase * registry ) [inline]
```

Register a camera type.

Parameters

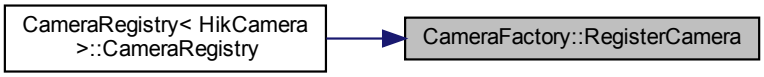
in	*camera_type_name*	Type name of camera.
in	*registry*	A registry object of camera.

Warning

You may call this function only when you're programming for a new type of camera.

Definition at line 62 of file camera_factory.h.

Here is the caller graph for this function:



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

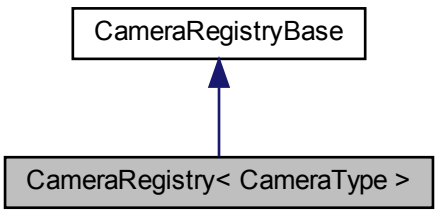
- modules/camera-base/ **camera_factory.h**

6.11 CameraRegistry< CameraType > Class Template Reference

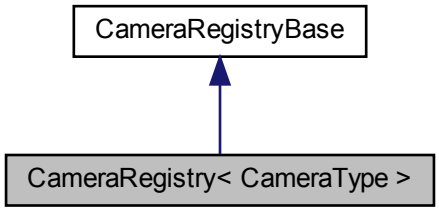
Templated camera registry class.

`#include <camera_factory.h>`

Inheritance diagram for CameraRegistry< CameraType >:



Collaboration diagram for CameraRegistry< CameraType >:



Public Member Functions

- **CameraRegistry** (const std::string &camera_type_name)
  Constructor of camera registry.
- **Camera** * **CreateCamera** () final
  Create a camera of this type.

Additional Inherited Members

6.11.1 Detailed Description

template<class CameraType>
class CameraRegistry< CameraType >

Templated camera registry class.

Template Parameters

CameraType	**Camera** (p. 30) type inherited from base class **Camera** (p. 30).

Attention

Once object is constructed, this type of camera will immediately be registered to camera factory.
This means the constructed object is useless and should not appear in any other place.
(Thus, template class though this is, it's better to be treated as a function.)

Warning

Camera (p. 30) factory will not check whether CameraType is really subclass of **Camera** (p. 30) base class.
(Thus, you should ensure that all callings of **CameraRegistry** (p. 41) constructor are completely under control.)

Definition at line 100 of file camera_factory.h.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 CameraRegistry()

```
template<class CameraType >
CameraRegistry< CameraType >::  CameraRegistry (
            const std::string & camera_type_name )  [inline], [explicit]
```

Constructor of camera registry.

Parameters

in	*camera_type_name*	Type name of camera.

Definition at line 106 of file camera_factory.h.

6.11.3 Member Function Documentation

6.11.3.1 CreateCamera()

```
template<class CameraType >
Camera* CameraRegistry< CameraType >::CreateCamera ( )  [inline], [final], [virtual]
```

Create a camera of this type.

Returns

A camera pointer.

Warning

NEVER directly call this function. Instead, it should be called by camera factory.

Implements **CameraRegistryBase** (p. 45).

Definition at line 115 of file camera_factory.h.

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

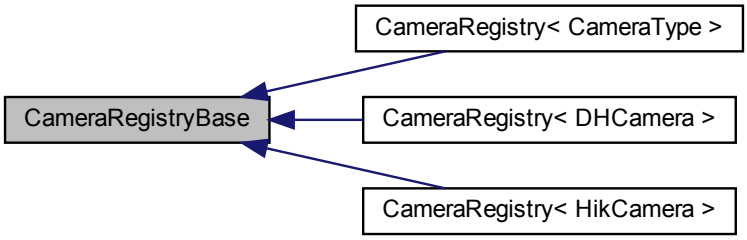
- modules/camera-base/ **camera_factory.h**

6.12 CameraRegistryBase Class Reference

Base (p. 25) class of camera registry.

`#include <camera_factory.h>`

Inheritance diagram for CameraRegistryBase:



Public Member Functions

- virtual **Camera** * **CreateCamera** ()=0
- **CameraRegistryBase** (const **CameraRegistryBase** &)=delete
- **CameraRegistryBase** & **operator=** (const **CameraRegistryBase** &)=delete

Protected Member Functions

- **CameraRegistryBase** ()=default
- virtual ~**CameraRegistryBase** ()=default

6.12.1 Detailed Description

Base (p. 25) class of camera registry.

Camera (p. 30) factory header.

Author

trantuan-20048607

Date

2022.1.28

Include this file to create camera objects.

Warning

You should use its subclass **CameraRegistry** (p. 41) instead of this base class.

Definition at line 18 of file camera_factory.h.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 CameraRegistryBase() [1/2]

```
CameraRegistryBase::CameraRegistryBase (
            const  CameraRegistryBase &  )  [delete]
```

6.12.2.2 CameraRegistryBase() [2/2]

```
CameraRegistryBase::CameraRegistryBase ( )  [protected], [default]
```

6.12.2.3 ~CameraRegistryBase()

```
virtual CameraRegistryBase::~CameraRegistryBase ( )  [protected], [virtual], [default]
```

6.12.3 Member Function Documentation

6.12.3.1 CreateCamera()

```
virtual  Camera* CameraRegistryBase::CreateCamera ( )  [pure virtual]
```

Implemented in **CameraRegistry< CameraType >** (p. 43), **CameraRegistry< DHCamera >** (p. 43), and **CameraRegistry< HikCamera >** (p. 43).

6.12.3.2 operator=()

```
CameraRegistryBase& CameraRegistryBase::operator= (
            const  CameraRegistryBase &  )  [delete]
```

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

- modules/camera-base/ **camera_factory.h**

6.13 CircularBuffer< Type, size > Class Template Reference

Circular buffer with mutex.

`#include <buffer.h>`

Public Member Functions

- **CircularBuffer** ()
- ~**CircularBuffer** ()=default
- unsigned int **Size** () const
- bool **Empty** () const

  Is this buffer empty?
- void **Push** (const Type &obj)

  Push an element.
- bool **Pop** (Type &obj)

  Pop an element.
- const Type & **operator[ ]** (unsigned int id)

6.13.1 Detailed Description

template<typename Type, unsigned int size>
class CircularBuffer< Type, size >

Circular buffer with mutex.

Circular buffer model header.

Author

anonymity, trantuan-20048607

Date

2022.1.28

Refer to `https://en.wikipedia.org/wiki/Circular_buffer`.

Template Parameters

Type	Type of elements in this buffer.
size	Max size of this buffer.

Attention

Size must be 2^N.

Definition at line 20 of file buffer.h.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 CircularBuffer()

```
template<typename Type , unsigned int size>
CircularBuffer< Type, size >::  CircularBuffer ( )  [inline]
```

Definition at line 30 of file buffer.h.

6.13.2.2 ~CircularBuffer()

```
template<typename Type , unsigned int size>
CircularBuffer< Type, size >::~ CircularBuffer ( )  [default]
```

6.13.3 Member Function Documentation

6.13.3.1 Empty()

```
template<typename Type , unsigned int size>
bool  CircularBuffer< Type, size >::Empty ( ) const  [inline]
```

Is this buffer empty?

Returns

Whether buffer is empty.

Definition at line 43 of file buffer.h.

6.13.3.2 operator[]()

```
template<typename Type , unsigned int size>
const Type&  CircularBuffer< Type, size >::operator[] (
             unsigned int id )  [inline]
```

Definition at line 78 of file buffer.h.

6.13.3.3 Pop()

```
template<typename Type , unsigned int size>
bool  CircularBuffer< Type, size >::Pop (
             Type & obj )  [inline]
```

Pop an element.

Parameters

out	*obj*	Output element.

Returns

Whether buffer is not empty.

Definition at line 67 of file buffer.h.

6.13.3.4 Push()

```
template<typename Type , unsigned int size>
void  CircularBuffer< Type, size >::Push (
             const Type & obj )  [inline]
```

Push an element.

Parameters

in	*obj*	Input element.

Definition at line 49 of file buffer.h.

6.13.3.5 Size()

```
template<typename Type , unsigned int size>
unsigned int  CircularBuffer< Type, size >::Size ( ) const  [inline]
```

Returns

Size of this buffer, which is specified when it is constructed.

Definition at line 37 of file buffer.h.

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

- modules/data-structure/ **buffer.h**

6.14 CmdlineArgParser Class Reference

Global command line argument parser.

```
#include <cmdline-arg-parser.h>
```

Public Member Functions

- **CmdlineArgParser** ()
- void **Parse** (int argc, char ∗argv[])

Static Public Member Functions

- static **CmdlineArgParser** & **Instance** ()

6.14.1 Detailed Description

Global command line argument parser.

Note

Use singleton pattern to make it global, refer to `https://en.wikipedia.org/wiki/Singleton_pattern`.

Definition at line 24 of file cmdline-arg-parser.h.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 CmdlineArgParser()

`CmdlineArgParser::CmdlineArgParser ( ) [inline]`

Definition at line 32 of file cmdline-arg-parser.h.

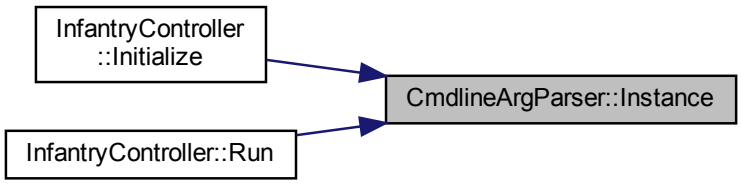
6.14.3 Member Function Documentation

6.14.3.1 Instance()

`static` **`CmdlineArgParser`**`& CmdlineArgParser::Instance ( ) [inline], [static]`

Definition at line 36 of file cmdline-arg-parser.h.

Here is the caller graph for this function:



6.14.3.2 Parse()

```
void CmdlineArgParser::Parse (
            int argc,
            char * argv[] )
```

Definition at line 14 of file cmdline-arg-parser.cpp.

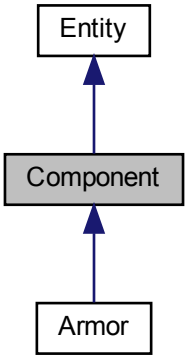
The documentation for this class was generated from the following files:

- modules/cmdline-arg-parser/ **cmdline-arg-parser.h**
- modules/cmdline-arg-parser/ **cmdline-arg-parser.cpp**

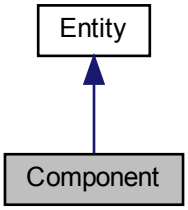
6.15 Component Class Reference

```
#include <component.h>
```

Inheritance diagram for Component:



Collaboration diagram for Component:



Public Types

- enum **ComponentType** { **kArmor** = 0, **SIZE** = 1 }

Public Member Functions

- **Component** (**Colors** color, **ComponentType** type)

Protected Attributes

- **ComponentType type_**

6.15.1 Detailed Description

Component (p. 50) definition header.

Author

trantuan-20048607

Date

2022.1.28

Attention

It's recommended to include **battlefield.h** (p. 168) for complete function.

Definition at line 13 of file component.h.

6.15.2 Member Enumeration Documentation

6.15.2.1 ComponentType

enum **Component::ComponentType**

Enumerator

kArmor	
SIZE	

Definition at line 15 of file component.h.

6.15.3 Constructor & Destructor Documentation

6.15.3.1 Component()

```
Component::Component (
            Colors color,
            ComponentType type )  [inline]
```

Definition at line 22 of file component.h.

6.15.4 Member Data Documentation

6.15.4.1 type_

```
ComponentType Component::type_  [protected]
```

Definition at line 28 of file component.h.

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

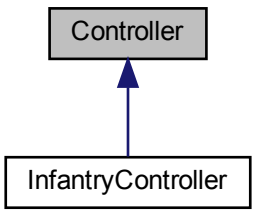
- modules/digital-twin/ **component.h**

6.16 Controller Class Reference

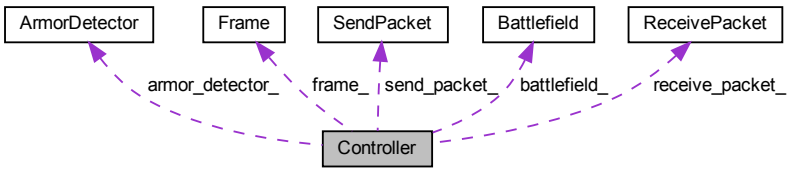
Controller (p. 52) base class.

```
#include <controller_base.h>
```

Inheritance diagram for Controller:



Collaboration diagram for Controller:



Public Member Functions

- **Controller** ()
- **Controller** (const **Controller** &)=delete
- **Controller** & **operator=** (const **Controller** &)=delete
- virtual bool **Initialize** ()=0
- virtual void **Run** ()=0

Protected Member Functions

- void **BboxToArmor** ()

  Convert boxes to armors.

Protected Attributes

- std::unique_ptr< **ImageProvider** > **image_provider_**

  Image provider handler.
- **Frame** **frame_**
- std::unique_ptr< **Serial** > **serial_**

  Serial (p. *136*) *communication handler.*
- **SendPacket** **send_packet_**
- **ReceivePacket** **receive_packet_**
- **ArmorDetector** **armor_detector_**
- std::vector< **bbox_t** > **boxes_**

  Store boxes here to speed up.
- std::vector< **Armor** > **armors_**

  Store armors here to speed up.
- **Battlefield** **battlefield_**

Static Protected Attributes

- static bool **exit_signal_**

  Global normal exit signal.

Friends

- void **SignalHandler** (int signal)
  Catch ctrl+c and exit safely.

6.16.1 Detailed Description

Controller (p. 52) base class.

Controller (p. 52) bass class header.

Author

trantuan-20048607, lzy20020320

Date

2022.1.28

Include this file only to declare or use pointers of controller.

Note

You cannot directly construct objects.
Instead, find controller types in subclass documents, include **controller_factory.h** (p. 154) and use **ControllerFactory::Instance()** (p. 60)::CreateController(controller_type_name).

Definition at line 23 of file controller_base.h.

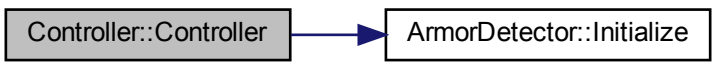
6.16.2 Constructor & Destructor Documentation

6.16.2.1 Controller() [1/2]

```
Controller::Controller ( )  [inline]
```

Definition at line 28 of file controller_base.h.

Here is the call graph for this function:



6.16.2.2 Controller() [2/2]

```
Controller::Controller (
            const  Controller &  )  [delete]
```

6.16.3 Member Function Documentation

6.16.3.1 BboxToArmor()

```
void Controller::BboxToArmor ( )  [inline], [protected]
```

Convert boxes to armors.

Attention

Since std::vector is not threading safe, do not use it in different threads.

Definition at line 62 of file controller_base.h.

Here is the caller graph for this function:

6.16.3.2 Initialize()

```
virtual bool Controller::Initialize ( )  [pure virtual]
```

Implemented in **InfantryController** (p. 119).

6.16.3.3 operator=()

```
Controller& Controller::operator= (
            const  Controller &  )  [delete]
```

6.16.3.4 Run()

```
virtual void Controller::Run ( )  [pure virtual]
```

Implemented in **InfantryController** (p. 119).

6.16.4 Friends And Related Function Documentation

6.16.4.1 SignalHandler

```
void SignalHandler (
            int signal )  [friend]
```

Catch ctrl+c and exit safely.

6.16.5 Member Data Documentation

6.16.5.1 armor_detector_

```
ArmorDetector Controller::armor_detector_  [protected]
```

Definition at line 51 of file controller_base.h.

6.16.5.2 armors_

```
std::vector< Armor> Controller::armors_  [protected]
```

Store armors here to speed up.

Definition at line 53 of file controller_base.h.

6.16.5.3 battlefield_

```
Battlefield Controller::battlefield_  [protected]
```

Definition at line 54 of file controller_base.h.

6.16.5.4 boxes_

```
std::vector< bbox_t> Controller::boxes_  [protected]
```

Store boxes here to speed up.

Definition at line 52 of file controller_base.h.

6.16.5.5 exit_signal_

```
bool Controller::exit_signal_  [static], [protected]
```

Global normal exit signal.

Definition at line 56 of file controller_base.h.

6.16.5.6 frame_

```
Frame Controller::frame_  [protected]
```

Definition at line 47 of file controller_base.h.

6.16.5.7 image_provider_

```
std::unique_ptr< ImageProvider> Controller::image_provider_  [protected]
```

Image provider handler.

Definition at line 46 of file controller_base.h.

6.16.5.8 receive_packet_

```
ReceivePacket Controller::receive_packet_  [protected]
```

Definition at line 50 of file controller_base.h.

6.16.5.9 send_packet_

SendPacket Controller::send_packet_ [protected]

Definition at line 49 of file controller_base.h.

6.16.5.10 serial_

std::unique_ptr< **Serial**> Controller::serial_ [protected]

Serial (p. 136) communication handler.

Definition at line 48 of file controller_base.h.

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

- modules/controller-base/ **controller_base.h**

6.17 ControllerFactory Class Reference

Singleton controller factory.

#include <controller_factory.h>

Public Member Functions

- **ControllerFactory** (const **ControllerFactory** &)=delete
- **ControllerFactory** & **operator=** (const **ControllerFactory** &)=delete
- void **RegisterController** (const std::string &controller_type_name, **ControllerRegistryBase** *registry)

  Register a controller type.
- **Controller** * **CreateController** (const std::string &controller_type_name)

  Create a controller whose type is registered to factory.

Static Public Member Functions

- static **ControllerFactory** & **Instance** ()

  Get the only instance_ of controller factory.

6.17.1 Detailed Description

Singleton controller factory.

For Singleton pattern, refer to https://en.wikipedia.org/wiki/Singleton_pattern.
For Factory pattern, refer to https://en.wikipedia.org/wiki/Factory_method_pattern.

Warning

Controller (p. 52) factory will not check whether ControllerType is really subclass of **Controller** (p. 52) base class.
(Thus, you should ensure that all callings of **ControllerRegistry** (p. 61) constructor are completely under control.)

Definition at line 45 of file controller_factory.h.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 ControllerFactory()

```
ControllerFactory::ControllerFactory (
            const  ControllerFactory &  )  [delete]
```

6.17.3 Member Function Documentation

6.17.3.1 CreateController()

```
Controller* ControllerFactory::CreateController (
            const std::string & controller_type_name )  [inline]
```

Create a controller whose type is registered to factory.

Parameters

in	*controller_type_name*	Type name of controller.

Returns

A pointer to crated controller.

Note

You may use macro **CREATE_CONTROLLER**(controller_type_name) instead of call this function.

Definition at line 76 of file controller_factory.h.

6.17.3.2 Instance()

```
static  ControllerFactory& ControllerFactory::Instance ( )  [inline], [static]
```

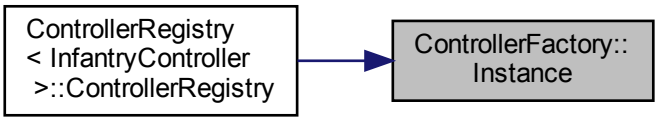
Get the only instance_ of controller factory.

Returns

A controller factory object.

Definition at line 55 of file controller_factory.h.

Here is the caller graph for this function:



6.17.3.3 operator=()

```
ControllerFactory& ControllerFactory::operator= (
            const  ControllerFactory &  )  [delete]
```

6.17.3.4 RegisterController()

```
void ControllerFactory::RegisterController (
            const std::string & controller_type_name,
             ControllerRegistryBase * registry )  [inline]
```

Register a controller type.

Parameters

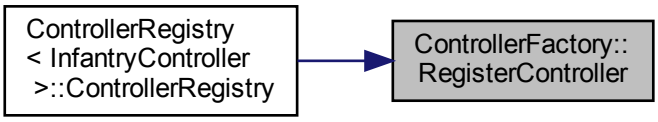
in	*controller_type_name*	Type name of controller.
in	*registry*	A registry object of controller.

Warning

You may call this function only when you're programming for a new type of controller.

Definition at line 66 of file controller_factory.h.

Here is the caller graph for this function:



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

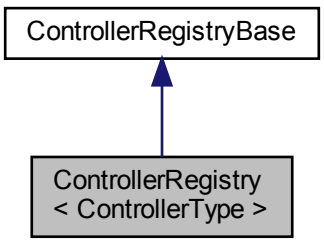
- modules/controller-base/ **controller_factory.h**

6.18 ControllerRegistry< ControllerType > Class Template Reference

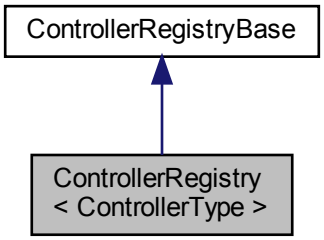
Templated controller registry class.

`#include <controller_factory.h>`

Inheritance diagram for ControllerRegistry< ControllerType >:



Collaboration diagram for ControllerRegistry< ControllerType >:



Public Member Functions

- **ControllerRegistry** (const std::string &controller_type_name)

  Constructor of controller registry.
- **Controller** * **CreateController** () final

  Create a controller of this type.

Additional Inherited Members

6.18.1 Detailed Description

template<class ControllerType>
class ControllerRegistry< ControllerType >

Templated controller registry class.

Template Parameters

ControllerType	**Controller** (p. 52) type inherited from base class **Controller** (p. 52).

Attention

Once object is constructed, this type of controller will immediately be registered to controller factory.
This means the constructed object is useless and should not appear in any other place.
(Thus, template class though this is, it's better to be treated as a function.)

Warning

Controller (p. 52) factory will not check whether ControllerType is really subclass of **Controller** (p. 52) base class.
(Thus, you should ensure that all callings of **ControllerRegistry** (p. 61) constructor are completely under control.)

Definition at line 105 of file controller_factory.h.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 ControllerRegistry()

```
template<class ControllerType >
ControllerRegistry< ControllerType >::  ControllerRegistry (
            const std::string & controller_type_name )  [inline], [explicit]
```

Constructor of controller registry.

Parameters

in	*controller_type_name*	Type name of controller.

Definition at line 111 of file controller_factory.h.

6.18.3 Member Function Documentation

6.18.3.1 CreateController()

```
template<class ControllerType >
Controller* ControllerRegistry< ControllerType >::CreateController ( )  [inline], [final],
[virtual]
```

Create a controller of this type.

Returns

A controller pointer.

Warning

NEVER directly call this function. Instead, it should be called by controller factory.

Implements **ControllerRegistryBase** (p. 65).

Definition at line 120 of file controller_factory.h.

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

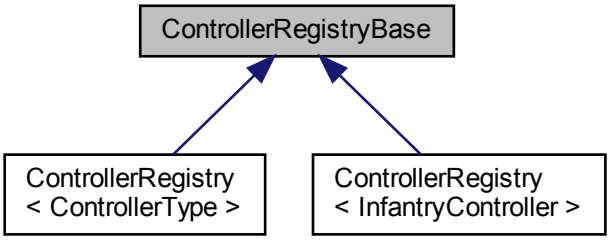
- modules/controller-base/ **controller_factory.h**

6.19 ControllerRegistryBase Class Reference

Base (p. 25) class of controller registry.

`#include <controller_factory.h>`

Inheritance diagram for ControllerRegistryBase:



Public Member Functions

- virtual **Controller** * **CreateController** ()=0
- **ControllerRegistryBase** (const **ControllerRegistryBase** &)=delete
- **ControllerRegistryBase** & **operator=** (const **ControllerRegistryBase** &)=delete

Protected Member Functions

- **ControllerRegistryBase** ()=default
- virtual ~**ControllerRegistryBase** ()=default

6.19.1 Detailed Description

Base (p. 25) class of controller registry.

Warning

You should use its subclass **ControllerRegistry** (p. 61) instead of this base class.

Definition at line 22 of file controller_factory.h.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 ControllerRegistryBase() [1/2]

```
ControllerRegistryBase::ControllerRegistryBase (
            const  ControllerRegistryBase &  )  [delete]
```

6.19.2.2 ControllerRegistryBase() [2/2]

```
ControllerRegistryBase::ControllerRegistryBase ( )  [protected], [default]
```

6.19.2.3 ∼ControllerRegistryBase()

```
virtual ControllerRegistryBase::∼ControllerRegistryBase ( )  [protected], [virtual], [default]
```

6.19.3 Member Function Documentation

6.19.3.1 CreateController()

```
virtual  Controller* ControllerRegistryBase::CreateController ( )  [pure virtual]
```

Implemented in **ControllerRegistry**< **ControllerType** > (p. 63), and **ControllerRegistry**< **InfantryController** > (p. 63).

6.19.3.2 operator=()

```
ControllerRegistryBase& ControllerRegistryBase::operator= (
            const  ControllerRegistryBase &  )  [delete]
```

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

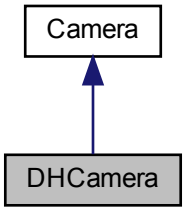
- modules/controller-base/ **controller_factory.h**

6.20 DHCamera Class Reference

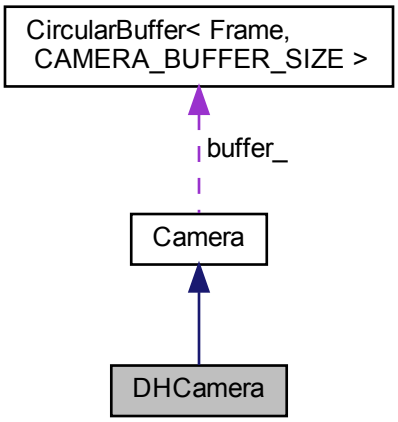
DaHeng camera class implementation.

`#include <camera_dh.h>`

Inheritance diagram for DHCamera:



Collaboration diagram for DHCamera:



Public Member Functions

- **DHCamera** ()
- **DHCamera** (const **DHCamera** &)=delete
- **DHCamera** & **operator=** (const **DHCamera** &)=delete
- ~**DHCamera** () final=default
- bool **OpenCamera** (const std::string &serial_number, const std::string &config_file) final

  Open a camera.

- bool **CloseCamera** () final

  Close the opened camera.
- bool **StartStream** () final

  Run the stream.
- bool **StopStream** () final

  Stop the stream.
- bool **IsConnected** () final

  Check if current device is connected.
- bool **GetFrame** (**Frame** &frame) final

  Get a frame with image and time stamp from internal image buffer.
- bool **ExportConfigurationFile** (const std::string &file_path) final

  Export current config to specified file.
- bool **ImportConfigurationFile** (const std::string &file_path) final

  Import current config to specified file.
- bool **SetExposureTime** (uint32_t exposure_time) final

  Set exposure time.
- bool **SetGainValue** (float gain_value) final

  Set gain value.

Additional Inherited Members

6.20.1 Detailed Description

DaHeng camera class implementation.

Warning

NEVER directly use this class to create camera!
Turn to **CameraFactory** (p. 38) class for correct method.

Definition at line 66 of file camera_dh.h.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 DHCamera() [1/2]

```
DHCamera::DHCamera ( )  [inline]
```

Definition at line 68 of file camera_dh.h.

6.20.2.2 DHCamera() [2/2]

```
DHCamera::DHCamera (
            const  DHCamera &  )  [delete]
```

6.20.2.3 ~DHCamera()

```
DHCamera::~DHCamera ( )  [final], [default]
```

6.20.3 Member Function Documentation

6.20.3.1 CloseCamera()

```
bool DHCamera::CloseCamera ( )  [final], [virtual]
```

Close the opened camera.

Returns

Whether the camera is closed normally.

Attention

No matter what is returned, the camera handle will be unreachable.

Implements **Camera** (p. 32).

Definition at line 118 of file camera_dh.cpp.

6.20.3.2 ExportConfigurationFile()

```
bool DHCamera::ExportConfigurationFile (
            const std::string & file_path )  [inline], [final], [virtual]
```

Export current config to specified file.

Parameters

in	*file_path*	File path.

Returns

Whether config file is saved.

Implements **Camera** (p. 32).

Definition at line 92 of file camera_dh.h.

Here is the caller graph for this function:

6.20.3.3 GetFrame()

```
bool DHCamera::GetFrame (
              Frame & frame )  [inline], [final], [virtual]
```

Get a frame with image and time stamp from internal image buffer.

Parameters

out	*frame*	Acquired frame will be stored here.

Returns

Whether buffer is not empty, or if you can successfully get an frame.

Implements **Camera** (p. 33).

Definition at line 90 of file camera_dh.h.

6.20.3.4 ImportConfigurationFile()

```
bool DHCamera::ImportConfigurationFile (
             const std::string & file_path )  [inline], [final], [virtual]
```

Import current config to specified file.

Parameters

in	*file_path*	File path.

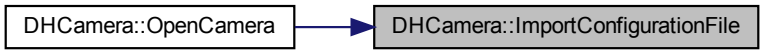
Returns

Whether config file is imported.

Implements **Camera** (p. 33).

Definition at line 99 of file camera_dh.h.

Here is the caller graph for this function:



6.20.3.5 IsConnected()

```
bool DHCamera::IsConnected ( )  [final], [virtual]
```

Check if current device is connected.

Returns

Whether current device is connected.

Implements **Camera** (p. 34).

Definition at line 214 of file camera_dh.cpp.

6.20.3.6 OpenCamera()

```
bool DHCamera::OpenCamera (
            const std::string & serial_number,
            const std::string & config_file )  [final], [virtual]
```

Open a camera.

Parameters

in	*serial_number*	**Serial** (p. 136) number of the camera you wanna open.
in	*config_file*	Will load config from this file.

Returns

Whether the camera is opened.

Note

Another try after failures is allowed.

Implements **Camera** (p. 34).

Definition at line 17 of file camera_dh.cpp.

Here is the call graph for this function:

6.20.3.7 operator=()

```
DHCamera& DHCamera::operator= (
            const  DHCamera &  )  [delete]
```

6.20.3.8 SetExposureTime()

```
bool DHCamera::SetExposureTime (
            uint32_t exposure_time )  [inline], [final], [virtual]
```

Set exposure time.

Parameters

exposure_time	Exposure time, automatically converted to corresponding data type.

Returns

Whether exposure time is set.

Implements **Camera** (p. 35).

Definition at line 106 of file camera_dh.h.

6.20.3.9 SetGainValue()

```
bool DHCamera::SetGainValue (
            float gain )  [inline], [final], [virtual]
```

Set gain value.

Parameters

gain	Gain value, automatically converted to corresponding data type.

Returns

Whether gain value is set.

Implements **Camera** (p. 35).

Definition at line 118 of file camera_dh.h.

6.20.3.10 StartStream()

```
bool DHCamera::StartStream ( )  [final], [virtual]
```

Run the stream.

Returns

Whether stream is started normally.

Attention

This function will return false when stream is already started or camera is not opened.

Implements **Camera** (p. 36).

Definition at line 170 of file camera_dh.cpp.

Here is the call graph for this function:

6.20.3.11 StopStream()

```
bool DHCamera::StopStream ( )  [final], [virtual]
```

Stop the stream.

Returns

Whether stream is stopped normally.

Attention

This function will return false when stream is not started or camera is not opened.

Implements **Camera** (p. 36).

Definition at line 190 of file camera_dh.cpp.

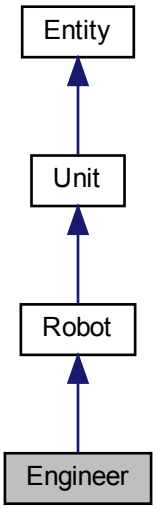
The documentation for this class was generated from the following files:

- modules/camera-dh/ **camera_dh.h**
- modules/camera-dh/ **camera_dh.cpp**

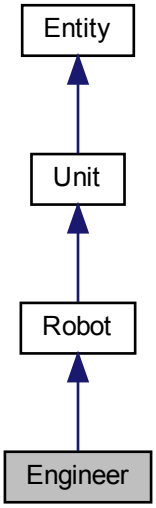
6.21 Engineer Class Reference

```
#include <engineer.h>
```

Inheritance diagram for Engineer:



Collaboration diagram for Engineer:



Public Member Functions

- **Engineer** (**Colors** color, double health, **RobotTypes** type= **kEngineer**)

Additional Inherited Members

6.21.1 Detailed Description

Engineer (p. 73) robot header.

Author

trantuan-20048607

Date

2022.1.28

Attention

It's recommended to include **battlefield.h** (p. 168) for complete function.

Definition at line 13 of file engineer.h.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 Engineer()

```
Engineer::Engineer (
              Colors color,
             double health,
              RobotTypes type =  kEngineer )  [inline]
```

Definition at line 15 of file engineer.h.

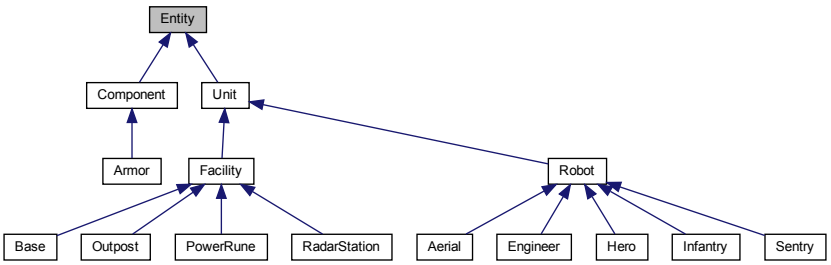
The documentation for this class was generated from the following file:

- modules/digital-twin/robots/ **engineer.h**

6.22 Entity Class Reference

```
#include <entity.h>
```

Inheritance diagram for Entity:



Public Types

- enum **Colors** {
  kBlue = 0, **kRed** = 1, **kPurple** = 2, **kGrey** = 3,
  SIZE = 4 }

Public Member Functions

- **Entity** (**Colors** color)
- **Entity** ()=delete

Protected Attributes

- **Colors color_**

6.22.1 Detailed Description

Entity (p. 75) definition header.

Author

trantuan-20048607

Date

2022.1.28

Attention

It's recommended to include **battlefield.h** (p. 168) for complete function.

Definition at line 14 of file entity.h.

6.22.2 Member Enumeration Documentation

6.22.2.1 Colors

`enum` **Entity::Colors**

Enumerator

kBlue	
kRed	
kPurple	
kGrey	
SIZE	

Definition at line 16 of file entity.h.

6.22.3 Constructor & Destructor Documentation

6.22.3.1 Entity() [1/2]

```
Entity::Entity (
            Colors color )  [inline], [explicit]
```

Definition at line 26 of file entity.h.

6.22.3.2 Entity() [2/2]

```
Entity::Entity ( )  [delete]
```

6.22.4 Member Data Documentation

6.22.4.1 color_

```
Colors Entity::color_  [protected]
```

Definition at line 32 of file entity.h.

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

- modules/digital-twin/ **entity.h**

6.23 ExtendedKalmanFilter< DataType, Nx, Ny > Class Template Reference

```
#include <predictor_ekf.h>
```

Public Types

- typedef Eigen::Matrix< DataType, Nx, Nx > **MatrixNxNx**
- typedef Eigen::Matrix< DataType, Ny, Nx > **MatrixNyNx**
- typedef Eigen::Matrix< DataType, Nx, Ny > **MatrixNxNy**
- typedef Eigen::Matrix< DataType, Ny, Ny > **MatrixNyNy**
- typedef Eigen::Matrix< DataType, Nx, 1 > **VectorNx**
- typedef Eigen::Matrix< DataType, Ny, 1 > **VectorNy**

Public Member Functions

- **ExtendedKalmanFilter** ()
- void **Initialize** (const **VectorNx** &x=VectorNx::Zero())
- template<typename Function >
  VectorNx Predict (Function &&function)
- template<typename Function >
  VectorNx Update (Function &&function, const **VectorNy** &y)

Public Attributes

- **VectorNx x_estimate_**
  Estimated status var. [Xe].
- **VectorNx x_predict_**
  Predicted status var. [Xp].
- **MatrixNxNx estimate_jacobi_**
  Prediction jacobi matrix. [F].
- **MatrixNyNx measure_jacobi_**
  Measurement jacobi matrix. [H].
- **MatrixNxNx status_cov_**
  Status covariance matrix. [P].
- **MatrixNxNx predict_cov_**
  Prediction covariance matrix. [Q].
- **MatrixNyNy measure_cov_**
  Measurement covariance matrix. [R].
- **MatrixNxNy kalman_gain_**
  Kalman gain. [K].
- **VectorNy y_predict_**
  Predicted measuring var. [Yp].

6.23.1 Detailed Description

template<typename DataType, unsigned int Nx, unsigned int Ny>
class ExtendedKalmanFilter< DataType, Nx, Ny >

Extended Kalman Filter templated class header.

Author

lzy20020320

Date

2022.1.30

Note

This file is only for internal use of implementation. To use predictor in other modules, include the corresponding headers.

Definition at line 16 of file predictor_ekf.h.

6.23.2 Member Typedef Documentation

6.23.2.1 MatrixNxNx

```
template<typename DataType , unsigned int Nx, unsigned int Ny>
typedef Eigen::Matrix<DataType, Nx, Nx>  ExtendedKalmanFilter< DataType, Nx, Ny >::  MatrixNxNx
```

Definition at line 18 of file predictor_ekf.h.

6.23.2.2 MatrixNxNy

```
template<typename DataType , unsigned int Nx, unsigned int Ny>
typedef Eigen::Matrix<DataType, Nx, Ny>  ExtendedKalmanFilter< DataType, Nx, Ny >::  MatrixNxNy
```

Definition at line 20 of file predictor_ekf.h.

6.23.2.3 MatrixNyNx

```
template<typename DataType , unsigned int Nx, unsigned int Ny>
typedef Eigen::Matrix<DataType, Ny, Nx>  ExtendedKalmanFilter< DataType, Nx, Ny >::  MatrixNyNx
```

Definition at line 19 of file predictor_ekf.h.

6.23.2.4 MatrixNyNy

```
template<typename DataType , unsigned int Nx, unsigned int Ny>
typedef Eigen::Matrix<DataType, Ny, Ny>  ExtendedKalmanFilter< DataType, Nx, Ny >::  MatrixNyNy
```

Definition at line 21 of file predictor_ekf.h.

6.23.2.5 VectorNx

```
template<typename DataType , unsigned int Nx, unsigned int Ny>
typedef Eigen::Matrix<DataType, Nx, 1>  ExtendedKalmanFilter< DataType, Nx, Ny >::  VectorNx
```

Definition at line 22 of file predictor_ekf.h.

6.23.2.6 VectorNy

```
template<typename DataType , unsigned int Nx, unsigned int Ny>
typedef Eigen::Matrix<DataType, Ny, 1>  ExtendedKalmanFilter< DataType, Nx, Ny >::  VectorNy
```

Definition at line 23 of file predictor_ekf.h.

6.23.3 Constructor & Destructor Documentation

6.23.3.1 ExtendedKalmanFilter()

```
template<typename DataType , unsigned int Nx, unsigned int Ny>
ExtendedKalmanFilter< DataType, Nx, Ny >::  ExtendedKalmanFilter ( )  [inline]
```

Definition at line 25 of file predictor_ekf.h.

6.23.4 Member Function Documentation

6.23.4.1 Initialize()

```
template<typename DataType , unsigned int Nx, unsigned int Ny>
void  ExtendedKalmanFilter< DataType, Nx, Ny >::Initialize (
            const  VectorNx & x = VectorNx::Zero() )  [inline]
```

Definition at line 40 of file predictor_ekf.h.

6.23.4.2 Predict()

```
template<typename DataType , unsigned int Nx, unsigned int Ny>
template<typename Function >
VectorNx  ExtendedKalmanFilter< DataType, Nx, Ny >::Predict (
            Function && function )  [inline]
```

Definition at line 43 of file predictor_ekf.h.

6.23.4.3 Update()

```
template<typename DataType , unsigned int Nx, unsigned int Ny>
template<typename Function >
VectorNx  ExtendedKalmanFilter< DataType, Nx, Ny >::Update (
            Function && function,
            const  VectorNy & y )  [inline]
```

Definition at line 62 of file predictor_ekf.h.

6.23.5 Member Data Documentation

6.23.5.1 estimate_jacobi_

```
template<typename DataType , unsigned int Nx, unsigned int Ny>
MatrixNxNx  ExtendedKalmanFilter< DataType, Nx, Ny >::estimate_jacobi_
```

Prediction jacobi matrix. [F].

Definition at line 88 of file predictor_ekf.h.

6.23.5.2 kalman_gain_

```
template<typename DataType , unsigned int Nx, unsigned int Ny>
MatrixNxNy  ExtendedKalmanFilter< DataType, Nx, Ny >::kalman_gain_
```

Kalman gain. [K].

Definition at line 93 of file predictor_ekf.h.

6.23.5.3 measure_cov_

```
template<typename DataType , unsigned int Nx, unsigned int Ny>
MatrixNyNy  ExtendedKalmanFilter< DataType, Nx, Ny >::measure_cov_
```

Measurement covariance matrix. [R].

Definition at line 92 of file predictor_ekf.h.

6.23.5.4 measure_jacobi_

```
template<typename DataType , unsigned int Nx, unsigned int Ny>
MatrixNyNx ExtendedKalmanFilter< DataType, Nx, Ny >::measure_jacobi_
```

Measurement jacobi matrix. [H].

Definition at line 89 of file predictor_ekf.h.

6.23.5.5 predict_cov_

```
template<typename DataType , unsigned int Nx, unsigned int Ny>
MatrixNxNx ExtendedKalmanFilter< DataType, Nx, Ny >::predict_cov_
```

Prediction covariance matrix. [Q].

Definition at line 91 of file predictor_ekf.h.

6.23.5.6 status_cov_

```
template<typename DataType , unsigned int Nx, unsigned int Ny>
MatrixNxNx ExtendedKalmanFilter< DataType, Nx, Ny >::status_cov_
```

Status covariance matrix. [P].

Definition at line 90 of file predictor_ekf.h.

6.23.5.7 x_estimate_

```
template<typename DataType , unsigned int Nx, unsigned int Ny>
VectorNx ExtendedKalmanFilter< DataType, Nx, Ny >::x_estimate_
```

Estimated status var. [Xe].

Definition at line 86 of file predictor_ekf.h.

6.23.5.8 x_predict_

```
template<typename DataType , unsigned int Nx, unsigned int Ny>
VectorNx ExtendedKalmanFilter< DataType, Nx, Ny >::x_predict_
```

Predicted status var. [Xp].

Definition at line 87 of file predictor_ekf.h.

6.23.5.9 y_predict_

```
template<typename DataType , unsigned int Nx, unsigned int Ny>
VectorNy  ExtendedKalmanFilter< DataType, Nx, Ny >::y_predict_
```

Predicted measuring var. [Yp].

Definition at line 94 of file predictor_ekf.h.

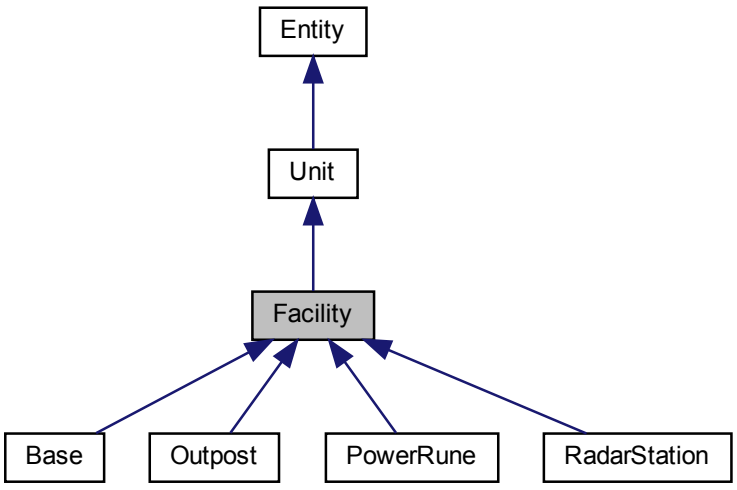
The documentation for this class was generated from the following file:

- modules/predictor-base/ **predictor_ekf.h**

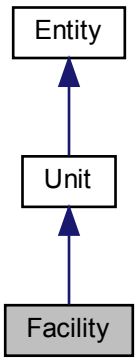
6.24 Facility Class Reference

```
#include <facility.h>
```

Inheritance diagram for Facility:



Collaboration diagram for Facility:



Public Types

- enum **FacilityTypes** {
  kBase = 0, **kOutpost** = 1, **kRadarStation** = 2, **kPowerRune** = 3,
  SIZE = 4 }

Public Member Functions

- **Facility** (**Colors** color, double health, **FacilityTypes** type)
- void **AddBottomArmor** (const **Armor** &armor)
- void **AddTopArmor** (const **Armor** &armor)

Protected Attributes

- **FacilityTypes type_**
- std::vector< **Armor** > **bottom_armors_**
- std::vector< **Armor** > **top_armors_**

6.24.1 Detailed Description

Facility (p. 83) definition header.

Author

trantuan-20048607

Date

2022.1.28

Attention

It's recommended to include **battlefield.h** (p. 168) for complete function.

Definition at line 14 of file facility.h.

6.24.2 Member Enumeration Documentation

6.24.2.1 FacilityTypes

```
enum Facility::FacilityTypes
```

Enumerator

kBase	
kOutpost	
kRadarStation	
kPowerRune	
SIZE	

Definition at line 16 of file facility.h.

6.24.3 Constructor & Destructor Documentation

6.24.3.1 Facility()

```
Facility::Facility (
            Colors color,
           double health,
            FacilityTypes type )  [inline]
```

Definition at line 30 of file facility.h.

6.24.4 Member Function Documentation

6.24.4.1 AddBottomArmor()

```
void Facility::AddBottomArmor (
            const  Armor & armor )  [inline]
```

Definition at line 36 of file facility.h.

6.24.4.2 AddTopArmor()

```
void Facility::AddTopArmor (
            const  Armor & armor )  [inline]
```

Definition at line 38 of file facility.h.

6.24.5 Member Data Documentation

6.24.5.1 bottom_armors_

```
std::vector< Armor> Facility::bottom_armors_  [protected]
```

Definition at line 43 of file facility.h.

6.24.5.2 top_armors_

```
std::vector< Armor> Facility::top_armors_  [protected]
```

Definition at line 44 of file facility.h.

6.24.5.3 type_

```
FacilityTypes Facility::type_  [protected]
```

Definition at line 41 of file facility.h.

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

- modules/digital-twin/ **facility.h**

6.25 Frame Struct Reference

Single frame structure.

`#include <frame.h>`

Public Member Functions

- **Frame** (cv::Mat &_image, uint64_t _time_stamp)
- **Frame** ()

Public Attributes

- cv::Mat **image**

  OpenCV style image matrix.
- uint64_t **time_stamp**

  Time stamp in DEC nanoseconds.

6.25.1 Detailed Description

Single frame structure.

Frame (p. 86) data structure header.

Author

trantuan-20048607

Date

2022.1.28

2 ways of initializing method provided:
(Default) Directly use **Frame()** (p. 87) to initialize an empty and useless frame.
(Manual) Use **Frame(_image, _time_stamp)** (p. 86) to initialize a complete frame.

Definition at line 18 of file frame.h.

6.25.2 Constructor & Destructor Documentation

6.25.2.1 Frame() [1/2]

```
Frame::Frame (
            cv::Mat & _image,
            uint64_t _time_stamp )  [inline]
```

Definition at line 22 of file frame.h.

6.25.2.2 Frame() [2/2]

```
Frame::Frame ( )  [inline]
```

Definition at line 27 of file frame.h.

6.25.3 Member Data Documentation

6.25.3.1 image

`cv::Mat Frame::image`

OpenCV style image matrix.

Definition at line 19 of file frame.h.

6.25.3.2 time_stamp

`uint64_t Frame::time_stamp`

Time stamp in DEC nanoseconds.

Definition at line 20 of file frame.h.

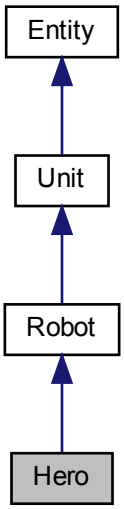
The documentation for this struct was generated from the following file:

- modules/data-structure/ **frame.h**

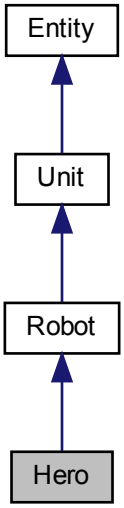
6.26 Hero Class Reference

`#include <hero.h>`

Inheritance diagram for Hero:



Collaboration diagram for Hero:



Public Member Functions

- **Hero** (**Colors** color, double health, **RobotTypes** type= **kHero**)

Additional Inherited Members

6.26.1 Detailed Description

Hero (p. 88) robot header.

Author

trantuan-20048607

Date

2022.1.28

Attention

It's recommended to include **battlefield.h** (p. 168) for complete function.

Definition at line 13 of file hero.h.

6.26.2 Constructor & Destructor Documentation

6.26.2.1 Hero()

```
Hero::Hero (
              Colors color,
             double health,
              RobotTypes type =  kHero )  [inline]
```

Definition at line 15 of file hero.h.

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

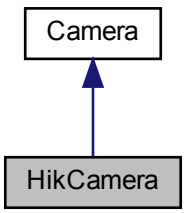
- modules/digital-twin/robots/ **hero.h**

6.27 HikCamera Class Reference

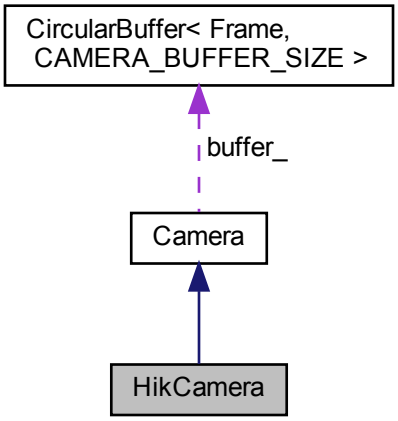
HikRobot camera class implementation.

```
#include <camera_hik.h>
```

Inheritance diagram for HikCamera:



Collaboration diagram for HikCamera:



Public Member Functions

- **HikCamera** ()
- **HikCamera** (const **HikCamera** &)=delete
- **HikCamera** & **operator=** (const **HikCamera** &)=delete
- ~**HikCamera** () final=default
- bool **OpenCamera** (const std::string &, const std::string &) final

  Open a camera.
- bool **StartStream** () final

  Run the stream.
- bool **GetFrame** (**Frame** &frame) final

  Get a frame with image and time stamp from internal image buffer.
- bool **StopStream** () final

  Stop the stream.
- bool **CloseCamera** () final

  Close the opened camera.
- bool **IsConnected** () final

  Check if current device is connected.
- bool **ExportConfigurationFile** (const std::string &file_path) final

  Export current config to specified file.
- bool **ImportConfigurationFile** (const std::string &file_path) final

  Import current config to specified file.
- bool **SetExposureTime** (uint32_t exposure_time) final

  Set exposure time.
- bool **SetGainValue** (float gain) final

  Set gain value.

Additional Inherited Members

6.27.1 Detailed Description

HikRobot camera class implementation.

Hik camera header.

Author

trantuan-20048607

Date

2022.1.28

Warning

NEVER include this file except in ./camera_hik.cpp.

NEVER directly use this class to create camera!
Turn to **CameraFactory** (p. 38) class for correct method.

Definition at line 18 of file camera_hik.h.

6.27.2 Constructor & Destructor Documentation

6.27.2.1 HikCamera() [1/2]

```
HikCamera::HikCamera ( )  [inline]
```

Definition at line 20 of file camera_hik.h.

6.27.2.2 HikCamera() [2/2]

```
HikCamera::HikCamera (
            const  HikCamera &  )  [delete]
```

6.27.2.3 ~HikCamera()

```
HikCamera::~HikCamera ( )  [final], [default]
```

6.27.3 Member Function Documentation

6.27.3.1 CloseCamera()

```
bool HikCamera::CloseCamera ( )  [final], [virtual]
```

Close the opened camera.

Returns

Whether the camera is closed normally.

Attention

No matter what is returned, the camera handle will be unreachable.

Implements **Camera** (p. 32).

Definition at line 201 of file camera_hik.cpp.

6.27.3.2 ExportConfigurationFile()

```
bool HikCamera::ExportConfigurationFile (
            const std::string & file_path )  [inline], [final], [virtual]
```

Export current config to specified file.

Parameters

in	*file_path*	File path.

Returns

Whether config file is saved.

Implements **Camera** (p. 32).

Definition at line 43 of file camera_hik.h.

Here is the caller graph for this function:

6.27.3.3 GetFrame()

```
bool HikCamera::GetFrame (
            Frame & frame )  [inline], [final], [virtual]
```

Get a frame with image and time stamp from internal image buffer.

Parameters

out	*frame*	Acquired frame will be stored here.

Returns

Whether buffer is not empty, or if you can successfully get an frame.

Implements **Camera** (p. 33).

Definition at line 32 of file camera_hik.h.

6.27.3.4 ImportConfigurationFile()

```
bool HikCamera::ImportConfigurationFile (
            const std::string & file_path )  [inline], [final], [virtual]
```

Import current config to specified file.

Parameters

in	*file_path*	File path.

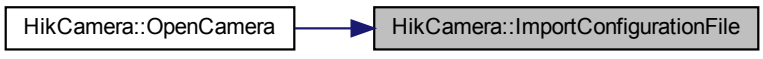
Returns

Whether config file is imported.

Implements **Camera** (p. 33).

Definition at line 54 of file camera_hik.h.

Here is the caller graph for this function:



6.27.3.5 IsConnected()

```
bool HikCamera::IsConnected ( )  [inline], [final], [virtual]
```

Check if current device is connected.

Returns

Whether current device is connected.

Implements **Camera** (p. 34).

Definition at line 38 of file camera_hik.h.

6.27.3.6 OpenCamera()

```
bool HikCamera::OpenCamera (
            const std::string & serial_number,
            const std::string & config_file )  [final], [virtual]
```

Open a camera.

Parameters

in	*serial_number*	**Serial** (p. 136) number of the camera you wanna open.
in	*config_file*	Will load config from this file.

Returns

Whether the camera is opened.

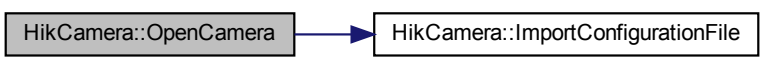
Note

Another try after failures is allowed.

Implements **Camera** (p. 34).

Definition at line 14 of file camera_hik.cpp.

Here is the call graph for this function:



6.27.3.7 operator=()

```
HikCamera& HikCamera::operator= (
            const  HikCamera &  )  [delete]
```

6.27.3.8 SetExposureTime()

```
bool HikCamera::SetExposureTime (
            uint32_t exposure_time )  [inline], [final], [virtual]
```

Set exposure time.

Parameters

exposure_time	Exposure time, automatically converted to corresponding data type.

Returns

Whether exposure time is set.

Implements **Camera** (p. 35).

Definition at line 65 of file camera_hik.h.

6.27.3.9 SetGainValue()

```
bool HikCamera::SetGainValue (
            float gain )  [inline], [final], [virtual]
```

Set gain value.

Parameters

gain	Gain value, automatically converted to corresponding data type.

Returns

Whether gain value is set.

Implements **Camera** (p. 35).

Definition at line 69 of file camera_hik.h.

6.27.3.10 StartStream()

```
bool HikCamera::StartStream ( )  [final], [virtual]
```

Run the stream.

Returns

Whether stream is started normally.

Attention

This function will return false when stream is already started or camera is not opened.

Implements **Camera** (p. 36).

Definition at line 145 of file camera_hik.cpp.

Here is the call graph for this function:

6.27.3.11 StopStream()

```
bool HikCamera::StopStream ( )  [final], [virtual]
```

Stop the stream.

Returns

Whether stream is stopped normally.

Attention

This function will return false when stream is not started or camera is not opened.

Implements **Camera** (p. 36).

Definition at line 184 of file camera_hik.cpp.

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

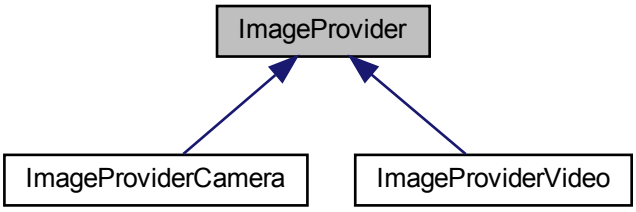
- modules/camera-hik/ **camera_hik.h**
- modules/camera-hik/ **camera_hik.cpp**

6.28 ImageProvider Class Reference

Image provider base class.

`#include <image_provider_base.h>`

Inheritance diagram for ImageProvider:



Public Member Functions

- **ImageProvider** ()=default
- virtual ~**ImageProvider** ()=default
- **ImageProvider** (const **ImageProvider** &)=delete
- **ImageProvider** & **operator=** (const **ImageProvider** &)=delete
- virtual bool **Initialize** (const std::string &file_path)=0

  Initialize by specified configuration file.
- virtual bool **GetFrame** (**Frame** &frame)=0

  Get a frame.

Protected Attributes

- cv::Mat **intrinsic_matrix_**

  Intrinsic matrix for solving PnP.
- cv::Mat **distortion_matrix_**

  Distortion matrix for solving PnP.

6.28.1 Detailed Description

Image provider base class.

Image provider bass class header.

Author

trantuan-20048607

Date

2022.1.28

Include this file only to declare or use pointers of image provider.

Note

You cannot directly construct objects.
Instead, find camera types in subclass documents, include **image_provider_factory.h** (p. 190) and use **CREATE_IMAGE_PROVIDER** macro.

Definition at line 20 of file image_provider_base.h.

6.28.2 Constructor & Destructor Documentation

6.28.2.1 ImageProvider() [1/2]

```
ImageProvider::ImageProvider ( )  [default]
```

6.28.2.2 ~ImageProvider()

```
virtual ImageProvider::~ImageProvider ( )  [virtual], [default]
```

All workings of release will be done here.

6.28.2.3 ImageProvider() [2/2]

```
ImageProvider::ImageProvider (
            const  ImageProvider &  )  [delete]
```

6.28.3 Member Function Documentation

6.28.3.1 GetFrame()

```
virtual bool ImageProvider::GetFrame (
              Frame & frame )  [pure virtual]
```

Get a frame.

Parameters

out	*frame*	OpenCV image reference.

Returns

Whether frame is complete.

Implemented in **ImageProviderCamera** (p. 103), and **ImageProviderVideo** (p. 114).

6.28.3.2 Initialize()

```
virtual bool ImageProvider::Initialize (
            const std::string & file_path )  [pure virtual]
```

Initialize by specified configuration file.

Parameters

in	*file_path*	Configuration file path.

Returns

Whether initialization succeeded.

Implemented in **ImageProviderCamera** (p. 104), and **ImageProviderVideo** (p. 115).

6.28.3.3 operator=()

```
ImageProvider& ImageProvider::operator= (
            const  ImageProvider &  )  [delete]
```

6.28.4 Member Data Documentation

6.28.4.1 distortion_matrix_

```
cv::Mat ImageProvider::distortion_matrix_  [protected]
```

Distortion matrix for solving PnP.

Definition at line 51 of file image_provider_base.h.

6.28.4.2 intrinsic_matrix_

```
cv::Mat ImageProvider::intrinsic_matrix_  [protected]
```

Intrinsic matrix for solving PnP.

Definition at line 50 of file image_provider_base.h.

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

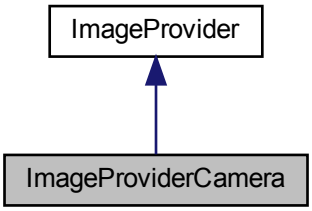
- modules/image-provider-base/ **image_provider_base.h**

6.29 ImageProviderCamera Class Reference

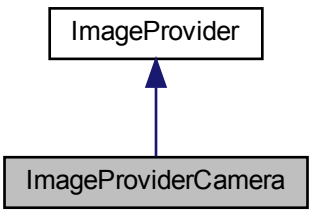
Camera (p. 30) image provider class implementation.

```
#include <image_provider_camera.h>
```

Inheritance diagram for ImageProviderCamera:



Collaboration diagram for ImageProviderCamera:



Public Member Functions

- **ImageProviderCamera** ()
- ~**ImageProviderCamera** () final
- bool **Initialize** (const std::string &) final

  Initialize by specified configuration file.
- bool **GetFrame** (**Frame** &frame) final

  Get a frame.

Additional Inherited Members

6.29.1 Detailed Description

Camera (p. 30) image provider class implementation.

Image provider camera header.

Author

trantuan-20048607

Date

2022.1.28

Warning

NEVER include this file except in ./image_provider_camera.cpp.

NEVER directly use this class to create image provider!
Instead, turn to **ImageProviderFactory** (p. 105) class and use **CREATE_IMAGE_PROVIDER**("IPCamera").

Definition at line 18 of file image_provider_camera.h.

6.29.2 Constructor & Destructor Documentation

6.29.2.1 ImageProviderCamera()

```
ImageProviderCamera::ImageProviderCamera ( )  [inline]
```

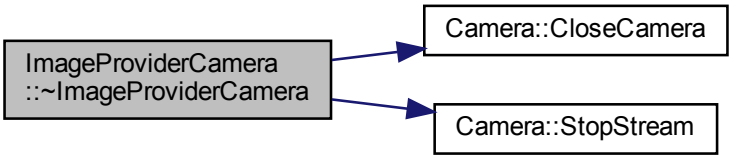
Definition at line 20 of file image_provider_camera.h.

6.29.2.2 ~ImageProviderCamera()

```
ImageProviderCamera::~ImageProviderCamera ( )  [final]
```

Definition at line 13 of file image_provider_camera.cpp.

Here is the call graph for this function:



6.29.3 Member Function Documentation

6.29.3.1 GetFrame()

```
bool ImageProviderCamera::GetFrame (
            Frame & frame )  [inline], [final], [virtual]
```

Get a frame.

Parameters

out	*frame*	OpenCV image reference.

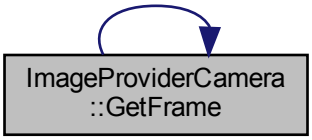
Returns

Whether frame is complete.

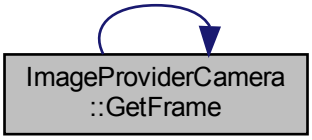
Implements **ImageProvider** (p. 99).

Definition at line 26 of file image_provider_camera.h.

Here is the call graph for this function:



Here is the caller graph for this function:



6.29.3.2 Initialize()

```
bool ImageProviderCamera::Initialize (
            const std::string & file_path )  [final], [virtual]
```

Initialize by specified configuration file.

Parameters

in	*file_path*	Configuration file path.

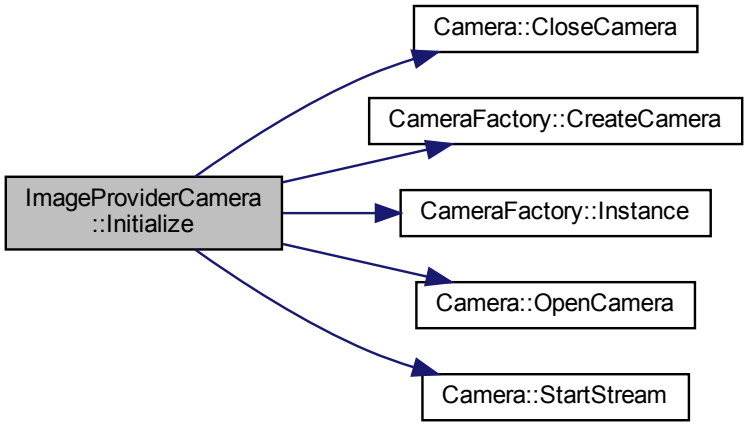
Returns

Whether initialization succeeded.

Implements **ImageProvider** (p. 100).

Definition at line 23 of file image_provider_camera.cpp.

Here is the call graph for this function:



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

- modules/image-provider-camera/ **image_provider_camera.h**
- modules/image-provider-camera/ **image_provider_camera.cpp**

6.30 ImageProviderFactory Class Reference

Singleton image provider factory.

`#include <image_provider_factory.h>`

Public Member Functions

- **ImageProviderFactory** (const **ImageProviderFactory** &)=delete
- **ImageProviderFactory** & **operator=** (const **ImageProviderFactory** &)=delete
- void **RegisterImageProvider** (const std::string &ip_type_name, **ImageProviderRegistryBase** ∗registry)

  Register an image provider type.
- **ImageProvider** ∗ **CreateImageProvider** (const std::string &ip_type_name)

  Create an image provider whose type is registered to factory.

Static Public Member Functions

- static **ImageProviderFactory** & **Instance** ()

  Get the only instance of image provider factory.

6.30.1 Detailed Description

Singleton image provider factory.

For Singleton pattern, refer to https://en.wikipedia.org/wiki/Singleton_pattern.
For Factory pattern, refer to https://en.wikipedia.org/wiki/Factory_method_pattern.

Warning

Image provider factory will not check whether IPType is really subclass of **ImageProvider** (p. 98) base class. (Thus, you should ensure that all callings of **ImageProviderRegistry** (p. 108) constructor are completely under control.)

Definition at line 45 of file image_provider_factory.h.

6.30.2 Constructor & Destructor Documentation

6.30.2.1 ImageProviderFactory()

```
ImageProviderFactory::ImageProviderFactory (
            const  ImageProviderFactory &  )  [delete]
```

6.30.3 Member Function Documentation

6.30.3.1 CreateImageProvider()

```
ImageProvider* ImageProviderFactory::CreateImageProvider (
            const std::string & ip_type_name )  [inline]
```

Create an image provider whose type is registered to factory.

Parameters

in	*ip_type_name*	Type name of image provider.

Returns

A pointer to crated image provider.

Note

You may use macro **CREATE_IMAGE_PROVIDER**(ip_type_name) instead of call this function.

Definition at line 77 of file image_provider_factory.h.

6.30.3.2 Instance()

```
static  ImageProviderFactory& ImageProviderFactory::Instance ( )  [inline], [static]
```

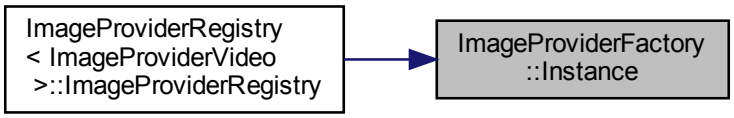
Get the only instance of image provider factory.

Returns

An image provider factory object.

Definition at line 55 of file image_provider_factory.h.

Here is the caller graph for this function:



6.30.3.3 operator=()

```
ImageProviderFactory& ImageProviderFactory::operator= (
            const  ImageProviderFactory &  )  [delete]
```

6.30.3.4 RegisterImageProvider()

```
void ImageProviderFactory::RegisterImageProvider (
            const std::string & ip_type_name,
             ImageProviderRegistryBase * registry )  [inline]
```

Register an image provider type.

Parameters

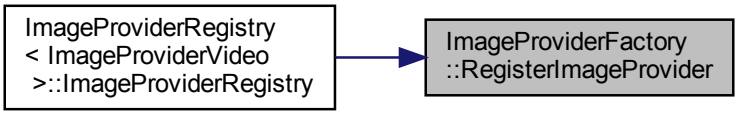
in	*ip_type_name*	Type name of image provider.
in	*registry*	A registry object of image provider.

Warning

You may call this function only when you're programming for a new type of image provider.

Definition at line 66 of file image_provider_factory.h.

Here is the caller graph for this function:



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

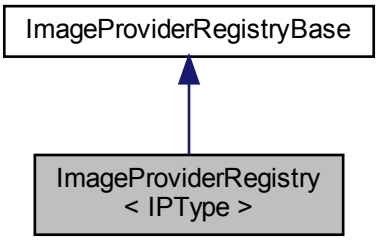
- modules/image-provider-base/ **image_provider_factory.h**

6.31 ImageProviderRegistry< IPType > Class Template Reference

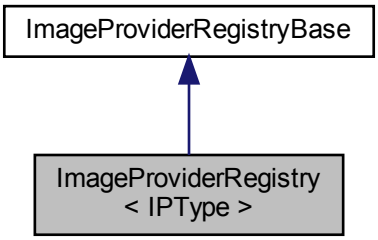
Templated image provider registry class.

`#include <image_provider_factory.h>`

Inheritance diagram for ImageProviderRegistry< IPType >:



Collaboration diagram for ImageProviderRegistry< IPType >:



Public Member Functions

- **ImageProviderRegistry** (const std::string &ip_type_name)

  Constructor of image provider registry.
- **ImageProvider** * **CreateImageProvider** () final

  Create an image provider of this type.

Additional Inherited Members

6.31.1 Detailed Description

template<class IPType>
class ImageProviderRegistry< IPType >

Templated image provider registry class.

Template Parameters

IPType	Image provider type inherited from base class **ImageProvider** (p. 98).

Attention

Once object is constructed, this type of image provider will immediately be registered to image provider factory.
This means the constructed object is useless and should not appear in any other place.
(Thus, template class though this is, it's better to be treated as a function.)

Warning

Image provider factory will not check whether IPType is really subclass of **ImageProvider** (p. 98) base class. (Thus, you should ensure that all callings of **ImageProviderRegistry** (p. 108) constructor are completely under control.)

Definition at line 107 of file image_provider_factory.h.

6.31.2 Constructor & Destructor Documentation

6.31.2.1 ImageProviderRegistry()

```
template<class IPType >
ImageProviderRegistry< IPType >::  ImageProviderRegistry (
            const std::string & ip_type_name )  [inline], [explicit]
```

Constructor of image provider registry.

Parameters

in	*ip_type_name*	Type name of image provider.

Definition at line 113 of file image_provider_factory.h.

6.31.3 Member Function Documentation

6.31.3.1 CreateImageProvider()

```
template<class IPType >
ImageProvider* ImageProviderRegistry< IPType >::CreateImageProvider ( )  [inline], [final],
[virtual]
```

Create an image provider of this type.

Returns

An image provider pointer.

Warning

NEVER directly call this function. Instead, it should be called by image provider factory.

Implements **ImageProviderRegistryBase** (p. 112).

Definition at line 122 of file image_provider_factory.h.

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

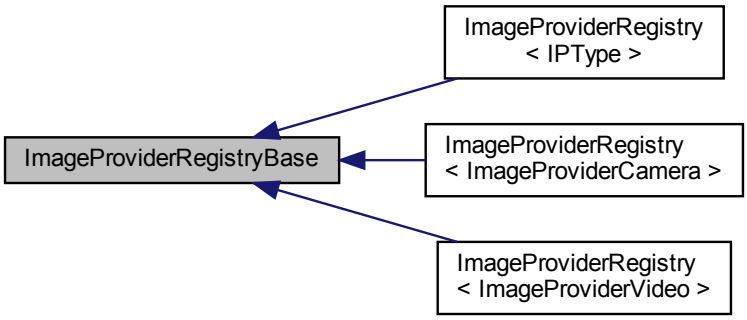
- modules/image-provider-base/ **image_provider_factory.h**

6.32 ImageProviderRegistryBase Class Reference

Base (p. 25) class of image provider registry.

`#include <image_provider_factory.h>`

Inheritance diagram for ImageProviderRegistryBase:



Public Member Functions

- virtual **ImageProvider** * **CreateImageProvider** ()=0
- **ImageProviderRegistryBase** (const **ImageProviderRegistryBase** &)=delete
- **ImageProviderRegistryBase** & **operator=** (const **ImageProviderRegistryBase** &)=delete

Protected Member Functions

- **ImageProviderRegistryBase** ()=default
- virtual ~**ImageProviderRegistryBase** ()=default

6.32.1 Detailed Description

Base (p. 25) class of image provider registry.

Warning

You should use its subclass **ImageProviderRegistry** (p. 108) instead of this base class.

Definition at line 22 of file image_provider_factory.h.

6.32.2 Constructor & Destructor Documentation

6.32.2.1 ImageProviderRegistryBase() [1/2]

```
ImageProviderRegistryBase::ImageProviderRegistryBase (
            const  ImageProviderRegistryBase &  )  [delete]
```

6.32.2.2 ImageProviderRegistryBase() [2/2]

```
ImageProviderRegistryBase::ImageProviderRegistryBase ( )  [protected], [default]
```

6.32.2.3 ~ImageProviderRegistryBase()

```
virtual ImageProviderRegistryBase::~ImageProviderRegistryBase ( )  [protected], [virtual],
[default]
```

6.32.3 Member Function Documentation

6.32.3.1 CreateImageProvider()

```
virtual  ImageProvider* ImageProviderRegistryBase::CreateImageProvider ( )  [pure virtual]
```

Implemented in **ImageProviderRegistry< IPType >** (p. 110), **ImageProviderRegistry< ImageProviderCamera >** (p. 110), and **ImageProviderRegistry< ImageProviderVideo >** (p. 110).

6.32.3.2 operator=()

```
ImageProviderRegistryBase& ImageProviderRegistryBase::operator= (
            const  ImageProviderRegistryBase &  )  [delete]
```

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

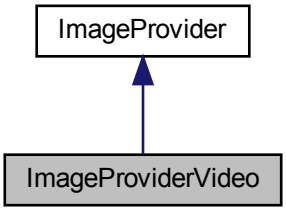
- modules/image-provider-base/ **image_provider_factory.h**

6.33 ImageProviderVideo Class Reference

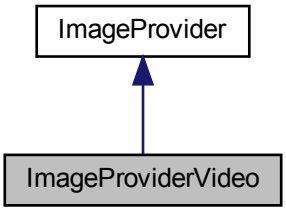
Video image provider class implementation.

`#include <image_provider_video.h>`

Inheritance diagram for ImageProviderVideo:



Collaboration diagram for ImageProviderVideo:



Public Member Functions

- **ImageProviderVideo** ()=default
- ~**ImageProviderVideo** () final
- bool **Initialize** (const std::string &) final

  Initialize by specified configuration file.
- bool **GetFrame** (**Frame** &frame) final

  Get a frame.

Additional Inherited Members

6.33.1 Detailed Description

Video image provider class implementation.

Image provider video header.

Author

trantuan-20048607

Date

2022.1.28

Warning

NEVER include this file except in ./image_provider_video.cpp.

NEVER directly use this class to create image provider!
Instead, turn to **ImageProviderFactory** (p. 105) class and use **CREATE_IMAGE_PROVIDER**("IPVideo").

Definition at line 18 of file image_provider_video.h.

6.33.2 Constructor & Destructor Documentation

6.33.2.1 ImageProviderVideo()

```
ImageProviderVideo::ImageProviderVideo ( )  [default]
```

6.33.2.2 ~ImageProviderVideo()

```
ImageProviderVideo::~ImageProviderVideo ( )  [final]
```

Definition at line 13 of file image_provider_video.cpp.

6.33.3 Member Function Documentation

6.33.3.1 GetFrame()

```
bool ImageProviderVideo::GetFrame (
            Frame & frame )  [inline], [final], [virtual]
```

Get a frame.

Parameters

out	*frame*	OpenCV image reference.

Returns

Whether frame is complete.

Implements **ImageProvider** (p. 99).

Definition at line 26 of file image_provider_video.h.

6.33.3.2 Initialize()

```
bool ImageProviderVideo::Initialize (
            const std::string & file_path )  [final], [virtual]
```

Initialize by specified configuration file.

Parameters

in	*file_path*	Configuration file path.

Returns

Whether initialization succeeded.

Implements **ImageProvider** (p. 100).

Definition at line 21 of file image_provider_video.cpp.

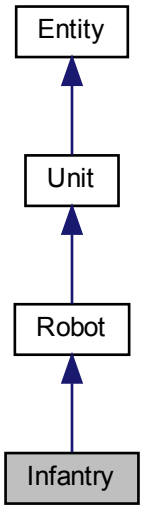
The documentation for this class was generated from the following files:

- modules/image-provider-video/ **image_provider_video.h**
- modules/image-provider-video/ **image_provider_video.cpp**

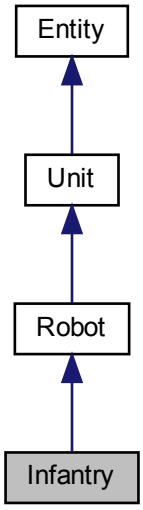
6.34 Infantry Class Reference

```
#include <infantry.h>
```

Inheritance diagram for Infantry:



Collaboration diagram for Infantry:



Public Member Functions

- **Infantry** (**Colors** color, double health, **RobotTypes** type)

Additional Inherited Members

6.34.1 Detailed Description

Infantry (p. 115) robot header.

Author

trantuan-20048607

Date

2022.1.28

Attention

It's recommended to include **battlefield.h** (p. 168) for complete function.

Definition at line 13 of file infantry.h.

6.34.2 Constructor & Destructor Documentation

6.34.2.1 Infantry()

```
Infantry::Infantry (
             Colors color,
            double health,
             RobotTypes type )  [inline]
```

Definition at line 15 of file infantry.h.

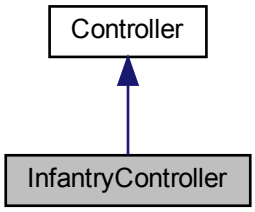
The documentation for this class was generated from the following file:

- modules/digital-twin/robots/ **infantry.h**

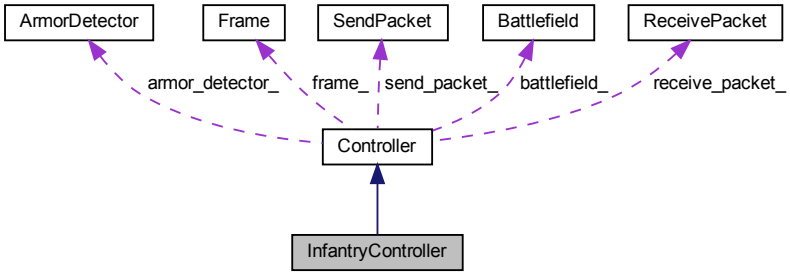
6.35 InfantryController Class Reference

`#include <controller_infantry.h>`

Inheritance diagram for InfantryController:



Collaboration diagram for InfantryController:



Public Member Functions

- bool **Initialize** () final
- void **Run** () final

Additional Inherited Members

6.35.1 Detailed Description

Infantry (p. 115) controller header.

Author

trantuan-20048607, screw-44

Date

2022.1.28

Warning

NEVER include this file except in ./controller_infantry.cpp.

Definition at line 13 of file controller_infantry.h.

6.35.2 Member Function Documentation

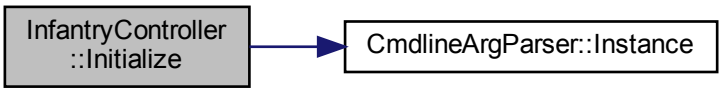
6.35.2.1 Initialize()

```
bool InfantryController::Initialize ( )  [final], [virtual]
```

Implements **Controller** (p. 55).

Definition at line 16 of file controller_infantry.cpp.

Here is the call graph for this function:



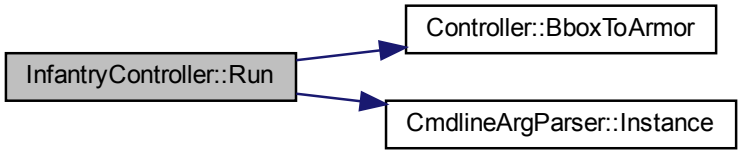
6.35.2.2 Run()

```
void InfantryController::Run ( )  [final], [virtual]
```

Implements **Controller** (p. 55).

Definition at line 44 of file controller_infantry.cpp.

Here is the call graph for this function:



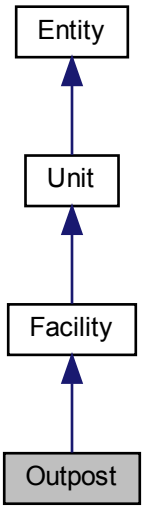
The documentation for this class was generated from the following files:

- modules/controller-infantry/ **controller_infantry.h**
- modules/controller-infantry/ **controller_infantry.cpp**

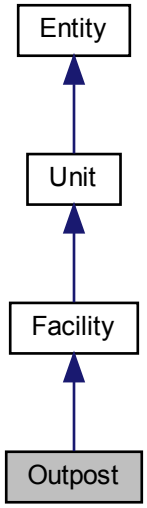
6.36 Outpost Class Reference

`#include <outpost.h>`

Inheritance diagram for Outpost:



Collaboration diagram for Outpost:



Public Member Functions

- **Outpost** (**Colors** color, double health, **FacilityTypes** type= **kOutpost**)

Additional Inherited Members

6.36.1 Detailed Description

Outpost (p. 120) definition header.

Author

trantuan-20048607

Date

2022.1.28

Attention

It's recommended to include **battlefield.h** (p. 168) for complete function.

Definition at line 13 of file outpost.h.

6.36.2 Constructor & Destructor Documentation

6.36.2.1 Outpost()

```
Outpost::Outpost (
              Colors color,
             double health,
              FacilityTypes type =  kOutpost )  [inline]
```

Definition at line 15 of file outpost.h.

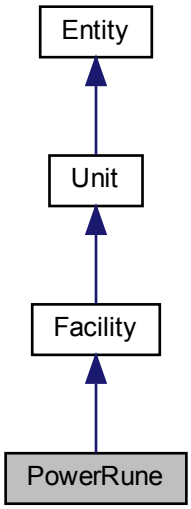
The documentation for this class was generated from the following file:

- modules/digital-twin/facilities/ **outpost.h**

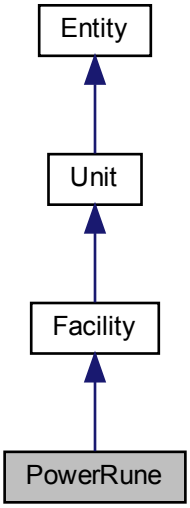
6.37 PowerRune Class Reference

```
#include <power_rune.h>
```

Inheritance diagram for PowerRune:



Collaboration diagram for PowerRune:



Public Member Functions

- **PowerRune** (**Colors** color, double health=0, **FacilityTypes** type= **kPowerRune**)

Additional Inherited Members

6.37.1 Detailed Description

Power rune definition header.

Author

trantuan-20048607

Date

2022.1.28

Attention

It's recommended to include **battlefield.h** (p. 168) for complete function.

Definition at line 13 of file power_rune.h.

6.37.2 Constructor & Destructor Documentation

6.37.2.1 PowerRune()

```
PowerRune::PowerRune (
              Colors color,
             double health = 0,
              FacilityTypes type =  kPowerRune )  [inline], [explicit]
```

Definition at line 15 of file power_rune.h.

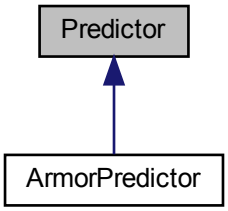
The documentation for this class was generated from the following file:

- modules/digital-twin/facilities/ **power_rune.h**

6.38 Predictor Class Reference

```
#include <predictor_base.h>
```

Inheritance diagram for Predictor:



Public Member Functions

- **Predictor** ()=default
- **Predictor** (const **Predictor** &)=delete
- **Predictor** & **operator=** (const **Predictor** &)=delete
- virtual bool **Initialize** ()=0
- virtual **SendPacket** **Predict** (**Battlefield** battlefield)=0

6.38.1 Detailed Description

Predictor (p. 124) base class header.

Author

screw-44

Date

2022.1.30

Definition at line 13 of file predictor_base.h.

6.38.2 Constructor & Destructor Documentation

6.38.2.1 Predictor() [1/2]

```
Predictor::Predictor ( )  [default]
```

6.38.2.2 Predictor() [2/2]

```
Predictor::Predictor (
            const  Predictor &  )  [delete]
```

6.38.3 Member Function Documentation

6.38.3.1 Initialize()

```
virtual bool Predictor::Initialize ( )  [pure virtual]
```

6.38.3.2 operator=()

```
Predictor& Predictor::operator= (
            const  Predictor &  )  [delete]
```

6.38.3.3 Predict()

```
virtual  SendPacket Predictor::Predict (
             Battlefield battlefield )  [pure virtual]
```

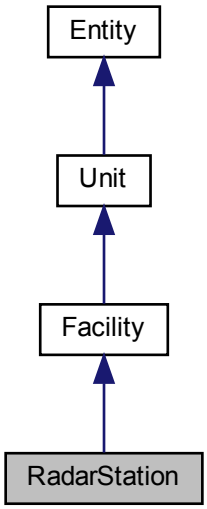
The documentation for this class was generated from the following file:

- modules/predictor-base/ **predictor_base.h**

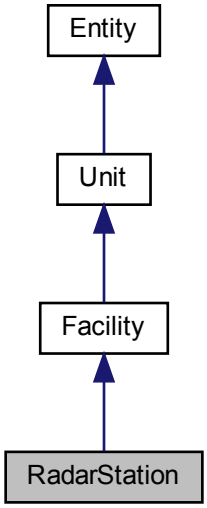
6.39 RadarStation Class Reference

```
#include <radar_station.h>
```

Inheritance diagram for RadarStation:



Collaboration diagram for RadarStation:



Public Member Functions

- **RadarStation** (**Colors** color, double health=0, **FacilityTypes** type= **kRadarStation**)

Additional Inherited Members

6.39.1 Detailed Description

Radar station definition header.

Author

trantuan-20048607

Date

2022.1.28

Attention

It's recommended to include **battlefield.h** (p. 168) for complete function.

Definition at line 13 of file radar_station.h.

6.39.2 Constructor & Destructor Documentation

6.39.2.1 RadarStation()

```
RadarStation::RadarStation (
             Colors color,
            double health = 0,
             FacilityTypes type =  kRadarStation )  [inline], [explicit]
```

Definition at line 15 of file radar_station.h.

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

- modules/digital-twin/facilities/ **radar_station.h**

6.40 ReceivePacket Struct Reference

Packet format received.

`#include <serial.h>`

Public Attributes

- int **mode**
- int **armor_kind**
- int **prior_enemy**
- int **color**
- float **bullet_speed**
- float **quaternion_0**
- float **quaternion_1**
- float **quaternion_2**
- float **quaternion_3**

6.40.1 Detailed Description

Packet format received.

Definition at line 21 of file serial.h.

6.40.2 Member Data Documentation

6.40.2.1 armor_kind

```
int ReceivePacket::armor_kind
```

Definition at line 23 of file serial.h.

6.40.2.2 bullet_speed

```
float ReceivePacket::bullet_speed
```

Definition at line 26 of file serial.h.

6.40.2.3 color

```
int ReceivePacket::color
```

Definition at line 25 of file serial.h.

6.40.2.4 mode

```
int ReceivePacket::mode
```

Definition at line 22 of file serial.h.

6.40.2.5 prior_enemy

```
int ReceivePacket::prior_enemy
```

Definition at line 24 of file serial.h.

6.40.2.6 quaternion_0

```
float ReceivePacket::quaternion_0
```

Definition at line 27 of file serial.h.

6.40.2.7 quaternion_1

```
float ReceivePacket::quaternion_1
```

Definition at line 28 of file serial.h.

6.40.2.8 quaternion_2

```
float ReceivePacket::quaternion_2
```

Definition at line 29 of file serial.h.

6.40.2.9 quaternion_3

```
float ReceivePacket::quaternion_3
```

Definition at line 30 of file serial.h.

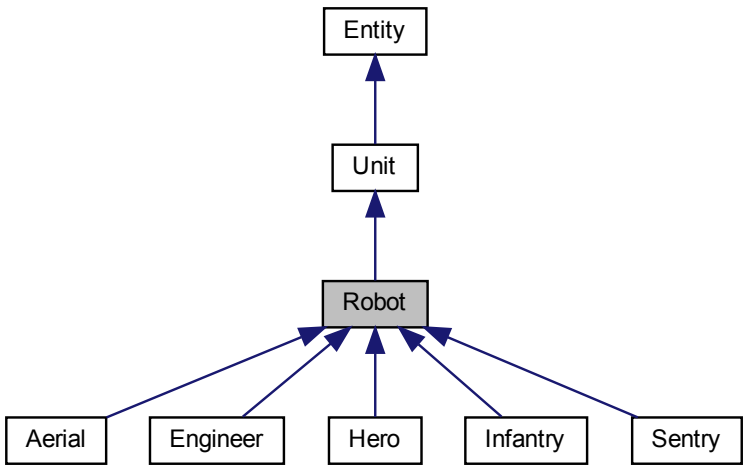
The documentation for this struct was generated from the following file:

- modules/data-structure/ **serial.h**

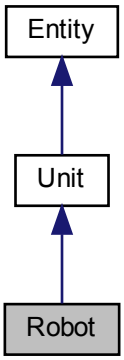
6.41 Robot Class Reference

```
#include <robot.h>
```

Inheritance diagram for Robot:



Collaboration diagram for Robot:



Public Types

- enum **RobotTypes** {
  kSentry = 0, **kHero** = 1, **kEngineer** = 2, **kInfantry3** = 3,
  kInfantry4 = 4, **kInfantry5** = 5, **kAerial** = 6, **SIZE** = 7 }

Public Member Functions

- **Robot** (**Colors** color, double health, **RobotTypes** type)
- void **AddArmor** (const **Armor** &armor)

Protected Attributes

- **RobotTypes type_**
- std::vector< **Armor** > **armors_**

6.41.1 Detailed Description

Robot (p. 130) definition header.

Author

trantuan-20048607

Date

2022.1.28

Attention

It's recommended to include **battlefield.h** (p. 168) for complete function.

Definition at line 14 of file robot.h.

6.41.2 Member Enumeration Documentation

6.41.2.1 RobotTypes

```
enum Robot::RobotTypes
```

Enumerator

kSentry	
kHero	
kEngineer	
kInfantry3	
kInfantry4	
kInfantry5	
kAerial	
SIZE	

Definition at line 16 of file robot.h.

6.41.3 Constructor & Destructor Documentation

6.41.3.1 Robot()

```
Robot::Robot (
             Colors color,
            double health,
             RobotTypes type )  [inline]
```

Definition at line 31 of file robot.h.

6.41.4 Member Function Documentation

6.41.4.1 AddArmor()

```
void Robot::AddArmor (
            const  Armor & armor )  [inline]
```

Definition at line 37 of file robot.h.

6.41.5 Member Data Documentation

6.41.5.1 armors_

```
std::vector< Armor> Robot::armors_  [protected]
```

Definition at line 42 of file robot.h.

6.41.5.2 type_

```
RobotTypes Robot::type_  [protected]
```

Definition at line 40 of file robot.h.

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

- modules/digital-twin/ **robot.h**

6.42 SendPacket Struct Reference

Packet format to send.

```
#include <serial.h>
```

Public Attributes

- float **yaw**
- float **pitch**
- float **delay**
- float **check_sum**

6.42.1 Detailed Description

Packet format to send.

Serial (p. 136) communication packet models header.

Author

anonymity, screw-44

Date

2022.1.28

Definition at line 13 of file serial.h.

6.42.2 Member Data Documentation

6.42.2.1 check_sum

```
float SendPacket::check_sum
```

Definition at line 17 of file serial.h.

6.42.2.2 delay

```
float SendPacket::delay
```

Definition at line 16 of file serial.h.

6.42.2.3 pitch

```
float SendPacket::pitch
```

Definition at line 15 of file serial.h.

6.42.2.4 yaw

```
float SendPacket::yaw
```

Definition at line 14 of file serial.h.

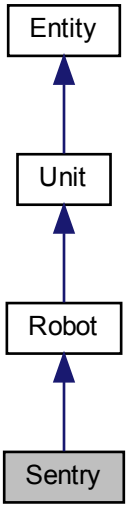
The documentation for this struct was generated from the following file:

- modules/data-structure/ **serial.h**

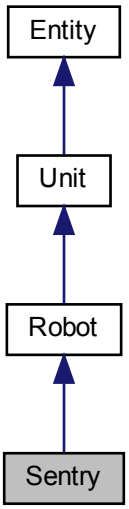
6.43 Sentry Class Reference

`#include <sentry.h>`

Inheritance diagram for Sentry:



Collaboration diagram for Sentry:



Public Member Functions

- **Sentry** (**Colors** color, double health, **RobotTypes** type= **kSentry**)

Additional Inherited Members

6.43.1 Detailed Description

Sentry (p. 135) robot header.

Author

trantuan-20048607

Date

2022.1.28

Attention

It's recommended to include **battlefield.h** (p. 168) for complete function.

Definition at line 13 of file sentry.h.

6.43.2 Constructor & Destructor Documentation

6.43.2.1 Sentry()

```
Sentry::Sentry (
             Colors color,
            double health,
             RobotTypes type =  kSentry )  [inline]
```

Definition at line 15 of file sentry.h.

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

- modules/digital-twin/robots/ **sentry.h**

6.44 Serial Class Reference

Serial (p. 136) manager class.

```
#include <serial.h>
```

Public Member Functions

- **Serial** ()
- **Serial** (const **Serial** &)=delete
- **Serial** & **operator=** (const **Serial** &)=delete
- ~**Serial** ()
- bool **IsOpened** () const

  Is serial port connected?
- bool **StartCommunication** ()

  Start serial communication.
- bool **StopCommunication** ()

  Stop serial communication.
- template<typename Rep , typename Period >
  bool **SendData** (**SendPacket** &data, const std::chrono::duration< Rep, Period > &duration)

  Send a data packet.
- template<typename Rep , typename Period >
  bool **GetData** (**ReceivePacket** &data, const std::chrono::duration< Rep, Period > &duration)

  Get a data packet.

6.44.1 Detailed Description

Serial (p. 136) manager class.

Definition at line 19 of file serial.h.

6.44.2 Constructor & Destructor Documentation

6.44.2.1 Serial() [1/2]

```
Serial::Serial ( )  [inline]
```

Definition at line 21 of file serial.h.

6.44.2.2 Serial() [2/2]

```
Serial::Serial (
            const  Serial &  )  [delete]
```

6.44.2.3 ~Serial()

```
Serial::~Serial ( )
```

Definition at line 324 of file serial.cpp.

Here is the call graph for this function:

6.44.3 Member Function Documentation

6.44.3.1 GetData()

```
template<typename Rep , typename Period >
bool Serial::GetData (
              ReceivePacket & data,
             const std::chrono::duration< Rep, Period > & duration )  [inline]
```

Get a data packet.

Parameters

out	*data*	A data packet to receive.
in	*duration*	Time duration.

Returns

Whether data packet is successfully read.

Definition at line 82 of file serial.h.

6.44.3.2 IsOpened()

```
bool Serial::IsOpened ( ) const  [inline]
```

Is serial port connected?

Returns

Whether serial port is opened.

Definition at line 40 of file serial.h.

6.44.3.3 operator=()

```
Serial& Serial::operator= (
            const  Serial &  )  [delete]
```

6.44.3.4 SendData()

```
template<typename Rep , typename Period >
bool Serial::SendData (
             SendPacket & data,
            const std::chrono::duration< Rep, Period > & duration )  [inline]
```

Send a data packet.

Parameters

in	*data*	A data packet to send.
in	*duration*	Time duration.

Returns

Whether data packet is successfully sent.

Definition at line 61 of file serial.h.

6.44.3.5 StartCommunication()

```
bool Serial::StartCommunication ( )
```

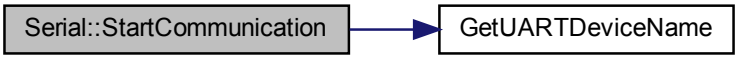
Start serial communication.

Returns

Whether communication is successfully established.

Definition at line 36 of file serial.cpp.

Here is the call graph for this function:



6.44.3.6 StopCommunication()

```
bool Serial::StopCommunication ( )
```

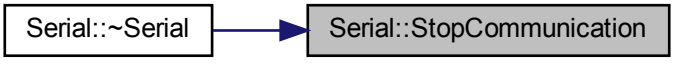
Stop serial communication.

Returns

Whether communication is successfully stopped.

Definition at line 63 of file serial.cpp.

Here is the caller graph for this function:



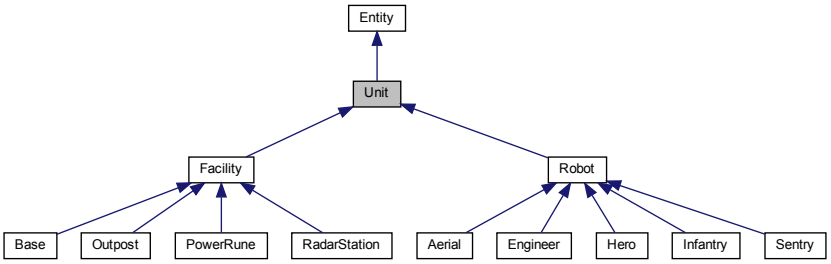
The documentation for this class was generated from the following files:

- modules/serial/ **serial.h**
- modules/serial/ **serial.cpp**

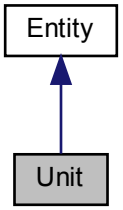
6.45 Unit Class Reference

`#include <unit.h>`

Inheritance diagram for Unit:



Collaboration diagram for Unit:



Public Member Functions

- **Unit** (**Colors** color, double health)

Protected Attributes

- double **health_**

Additional Inherited Members

6.45.1 Detailed Description

Unit (p. 141) definition header.

Author

trantuan-20048607

Date

2022.1.28

Attention

It's recommended to include **battlefield.h** (p. 168) for complete function.

Definition at line 13 of file unit.h.

6.45.2 Constructor & Destructor Documentation

6.45.2.1 Unit()

```
Unit::Unit (
              Colors color,
             double health )  [inline]
```

Definition at line 17 of file unit.h.

6.45.3 Member Data Documentation

6.45.3.1 health_

```
double Unit::health_  [protected]
```

Definition at line 23 of file unit.h.

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

- modules/digital-twin/ **unit.h**

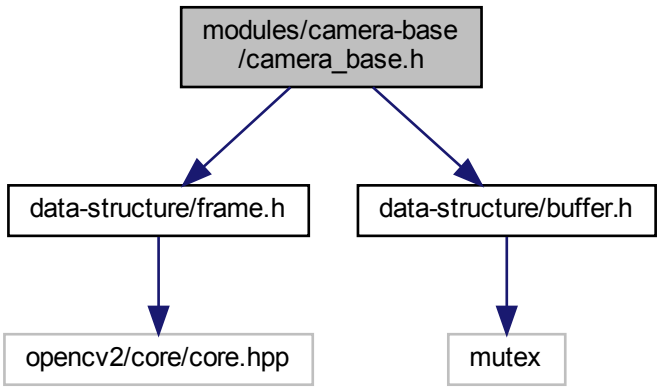
Chapter 7

File Documentation

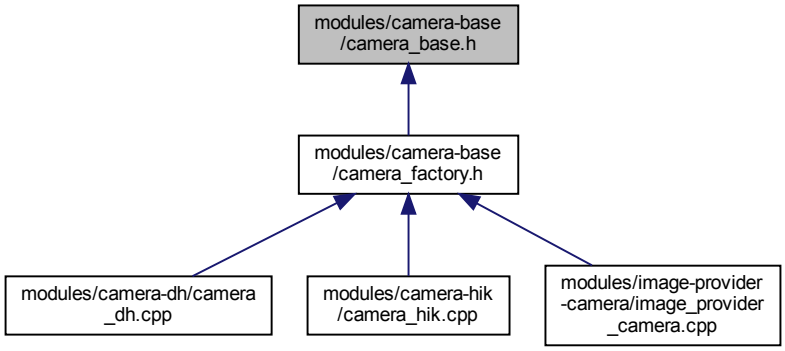
7.1 modules/camera-base/camera_base.h File Reference

```
#include "data-structure/frame.h"
#include "data-structure/buffer.h"
```

Include dependency graph for camera_base.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **Camera**

  Camera *(p. 30) base class.*

Macros

- #define **CAMERA_BUFFER_SIZE** 4

  Buffer size for image provider and camera.

7.1.1 Macro Definition Documentation

7.1.1.1 CAMERA_BUFFER_SIZE

```
#define CAMERA_BUFFER_SIZE 4
```

Buffer size for image provider and camera.

Camera (p. 30) bass class header.

Author

trantuan-20048607

Date

2022.1.28

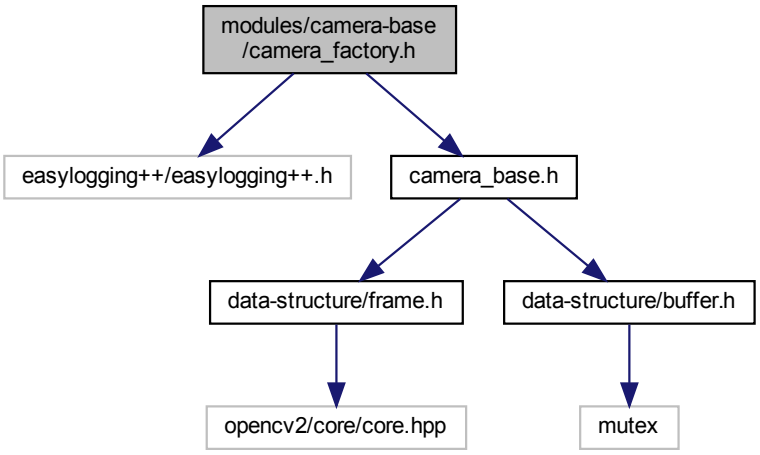
Include this file only to declare or use pointers of camera.

Definition at line 15 of file camera_base.h.

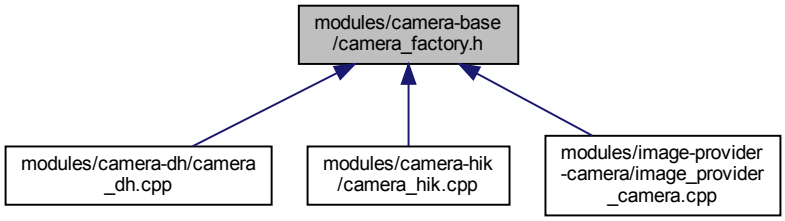
7.2 modules/camera-base/camera_factory.h File Reference

```
#include "easylogging++/easylogging++.h"
#include "camera_base.h"
```

Include dependency graph for camera_factory.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **CameraRegistryBase**

  Base *(p. 25) class of camera registry.*
- class **CameraFactory**

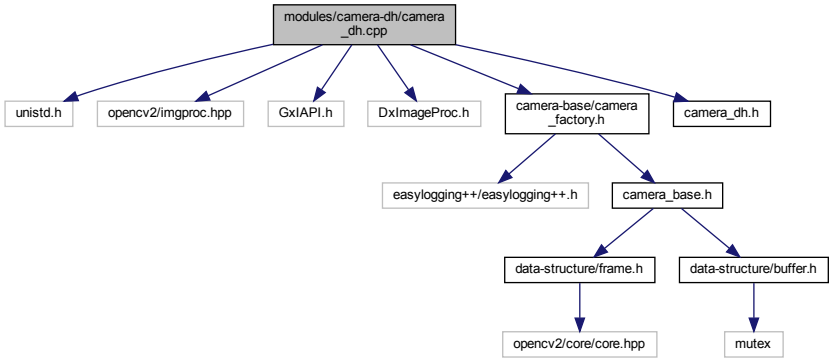
  Singleton camera factory.
- class **CameraRegistry**< **CameraType** >

  Templated camera registry class.

7.3 modules/camera-dh/camera_dh.cpp File Reference

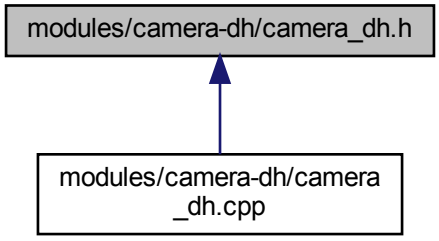
```
#include <unistd.h>
#include <opencv2/imgproc.hpp>
#include <GxIAPI.h>
#include <DxImageProc.h>
#include "camera-base/camera_factory.h"
#include "camera_dh.h"
```

Include dependency graph for camera_dh.cpp:



7.4 modules/camera-dh/camera_dh.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class **DHCamera**

  DaHeng camera class implementation.

Macros

- #define **GX_OPEN_CAMERA_CHECK_STATUS**(status_code)

  This macro is used to check if the device is successfully initialized.
- #define **GX_CHECK_STATUS**(status_code)

  This macro is used to check if parameters are successfully modified or set.
- #define **GX_START_STOP_STREAM_CHECK_STATUS_**(status_code)

  This macro is used to check if the stream is successfully opened or closed.

7.4.1 Macro Definition Documentation

7.4.1.1 GX_CHECK_STATUS

```
#define GX_CHECK_STATUS(
              status_code )
```

Value:

```
    if ((status_code) != GX_STATUS_SUCCESS) {     \
        LOG(ERROR) « GetErrorInfo(status_code);  \
        return false;                             \
    }
```

This macro is used to check if parameters are successfully modified or set.

Warning

Do NOT use this macro in other place!

Definition at line 37 of file camera_dh.h.

7.4.1.2 GX_OPEN_CAMERA_CHECK_STATUS

```
#define GX_OPEN_CAMERA_CHECK_STATUS(
              status_code )
```

Value:

```
    if ((status_code) != GX_STATUS_SUCCESS) {             \
        LOG(ERROR) « GetErrorInfo(status_code);          \
        (status_code) = GXCloseDevice(device_);           \
        if ((status_code) != GX_STATUS_SUCCESS)           \
            LOG(ERROR) « GetErrorInfo(status_code);      \
        device_ = nullptr;                                \
        serial_number_ = "";                              \
        if (!camera_count_) {                             \
            (status_code) = GXCloseLib();                 \
            if ((status_code) != GX_STATUS_SUCCESS)       \
                LOG(ERROR) « GetErrorInfo(status_code);  \
        }                                                 \
        return false;                                     \
    }
```

This macro is used to check if the device is successfully initialized.

DaHeng camera header.

Author

trantuan-20048607

Date

2022.1.28

Warning

NEVER include this file except in ./camera_dh.cpp.

Do NOT use this macro in other place!

Definition at line 17 of file camera_dh.h.

7.4.1.3 GX_START_STOP_STREAM_CHECK_STATUS_

```
#define GX_START_STOP_STREAM_CHECK_STATUS_(
              status_code )
```

Value:

```
    if ((status_code) != GX_STATUS_SUCCESS) {            \
        if (raw_16_to_8_cache_ != nullptr) {             \
            delete[] raw_16_to_8_cache_;                 \
            raw_16_to_8_cache_ = nullptr;                \
        }                                                \
        if (raw_8_to_rgb_24_cache_ != nullptr) {         \
            delete[] raw_8_to_rgb_24_cache_;             \
            raw_8_to_rgb_24_cache_ = nullptr;            \
        }                                                \
        LOG(ERROR) « GetErrorInfo(status_code);         \
        return false;                                    \
    }
```

This macro is used to check if the stream is successfully opened or closed.

Warning

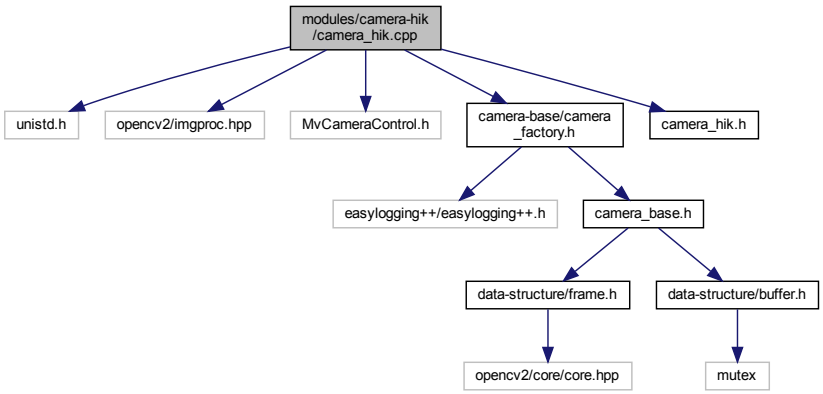
Do NOT use this macro in other place!

Definition at line 47 of file camera_dh.h.

7.5 modules/camera-hik/camera_hik.cpp File Reference

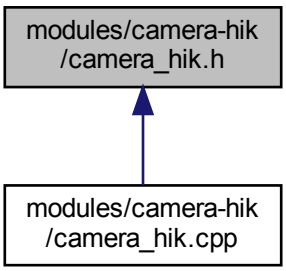
```
#include <unistd.h>
#include <opencv2/imgproc.hpp>
#include <MvCameraControl.h>
#include "camera-base/camera_factory.h"
#include "camera_hik.h"
```

Include dependency graph for camera_hik.cpp:



7.6 modules/camera-hik/camera_hik.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

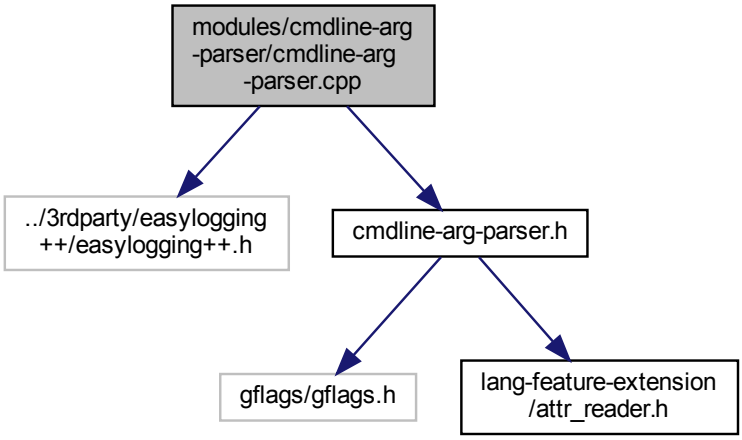
- class **HikCamera**

  HikRobot camera class implementation.

7.7 modules/cmdline-arg-parser/cmdline-arg-parser.cpp File Reference

```
#include "../3rdparty/easylogging++/easylogging++.h"
#include "cmdline-arg-parser.h"
```

Include dependency graph for cmdline-arg-parser.cpp:



Functions

- **DEFINE_string** (type, "", "controller type")
- **DEFINE_bool** (camera, false, "run with camera")
- **DEFINE_bool** (serial, false, "run with serial")

7.7.1 Function Documentation

7.7.1.1 DEFINE_bool() [1/2]

```
DEFINE_bool (
            camera ,
            false ,
            "run with camera"  )
```

7.7.1.2 DEFINE_bool() [2/2]

```
DEFINE_bool (
            serial ,
            false ,
            "run with serial"  )
```

7.7.1.3 DEFINE_string()

```
DEFINE_string (
             type ,
             "" ,
             "controller type"  )
```

Note

DEFINE flags declared in .h file here. Refer to `https://gflags.github.io/gflags/`.

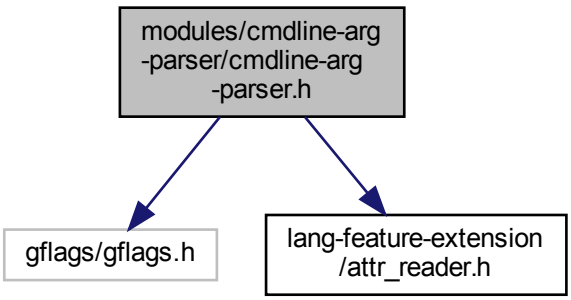
Warning

Flags will be initialized before the program entering the main function! This means any error occurring here will not be caught unless you're using debugger. (Thus, do not use this variable in any other place and you should not modify it.)

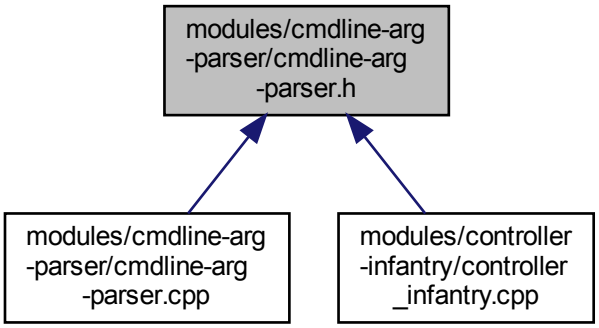
7.8 modules/cmdline-arg-parser/cmdline-arg-parser.h File Reference

```
#include <gflags/gflags.h>
#include "lang-feature-extension/attr_reader.h"
```

Include dependency graph for cmdline-arg-parser.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **CmdlineArgParser**

  Global command line argument parser.

Functions

- **DECLARE_string** (type)
- **DECLARE_bool** (camera)
- **DECLARE_bool** (serial)

7.8.1 Function Documentation

7.8.1.1 DECLARE_bool() [1/2]

```
DECLARE_bool (
             camera  )
```

7.8.1.2 DECLARE_bool() [2/2]

```
DECLARE_bool (
             serial  )
```

7.8.1.3 DECLARE_string()

```
DECLARE_string (
            type  )
```

Command line parser header.

Author

screw-44, trantuan

Date

2021.1.28

All arguments will be defined and parsed in this class.

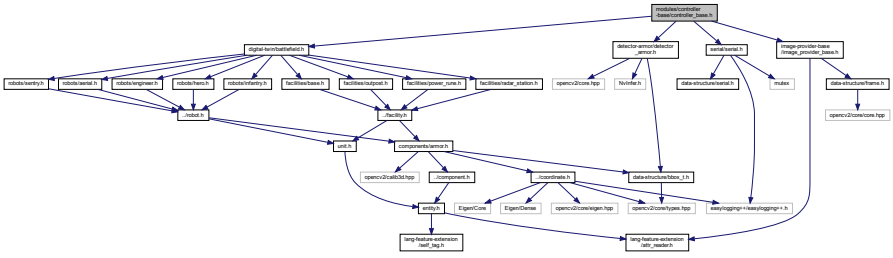
Attention

Only DECLARE args here. Args should be DEFINED in .cpp file.

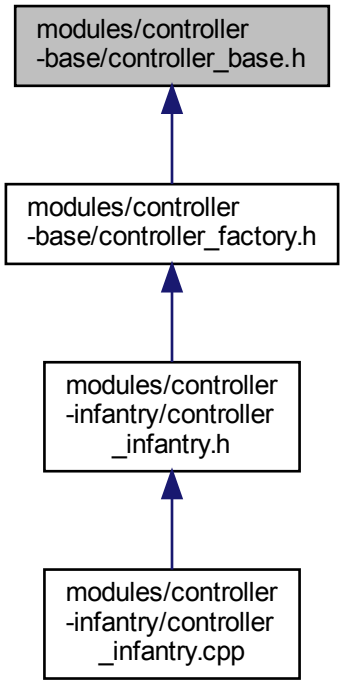
7.9 modules/controller-base/controller_base.h File Reference

```
#include "digital-twin/battlefield.h"
#include "serial/serial.h"
#include "image-provider-base/image_provider_base.h"
#include "detector-armor/detector_armor.h"
```

Include dependency graph for controller_base.h:



This graph shows which files directly or indirectly include this file:



Classes

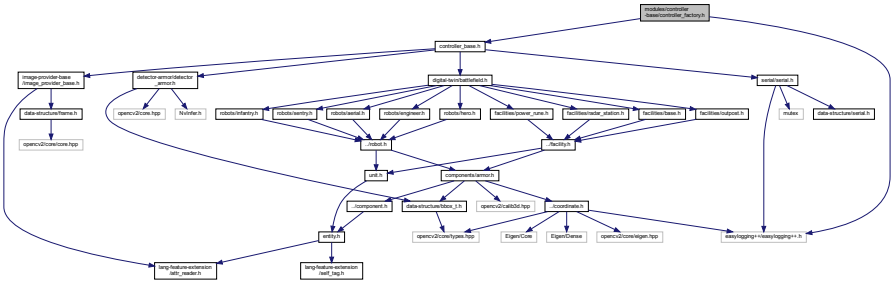
- class **Controller**

  Controller (p. *52) base class.*

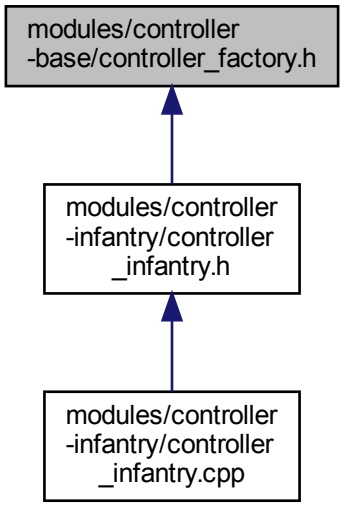
7.10 modules/controller-base/controller_factory.h File Reference

```
#include "easylogging++/easylogging++.h"
#include "controller_base.h"
```

Include dependency graph for controller_factory.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **ControllerRegistryBase**

  Base *(p. 25) class of controller registry.*
- class **ControllerFactory**

  Singleton controller factory.
- class **ControllerRegistry**< **ControllerType** >

  Templated controller registry class.

Macros

- #define **__CREATE_CONTROLLER__**(controller_type_name) **ControllerFactory::Instance**().Create↩ Controller(controller_type_name)

  A macro to create a controller of specified type name. Turn to class ***ControllerFactory*** *(p. 58) for details.*

7.10.1 Macro Definition Documentation

7.10.1.1 __CREATE_CONTROLLER__

```
#define __CREATE_CONTROLLER__(
              controller_type_name )  ControllerFactory::Instance().CreateController(controller↩
_type_name)
```

A macro to create a controller of specified type name. Turn to class **ControllerFactory** (p. 58) for details.

Controller (p. 52) factory header.

Author

trantuan-20048607

Date

2022.1.28

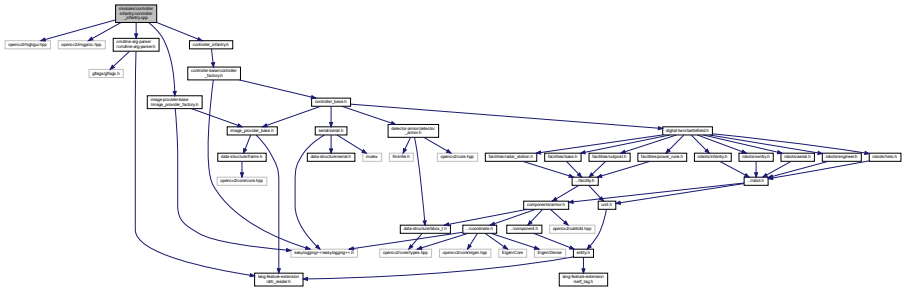
Include this file to create controller objects.

Definition at line 15 of file controller_factory.h.

7.11 modules/controller-infantry/controller_infantry.cpp File Reference

```
#include <opencv2/highgui.hpp>
#include <opencv2/imgproc.hpp>
#include "cmdline-arg-parser/cmdline-arg-parser.h"
#include "image-provider-base/image_provider_factory.h"
#include "controller_infantry.h"
```

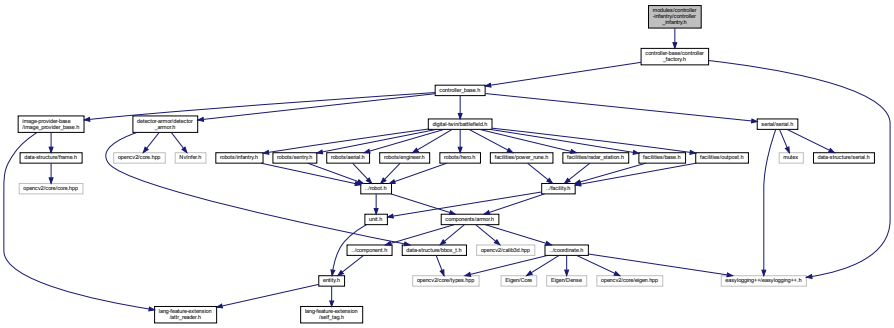
Include dependency graph for controller_infantry.cpp:



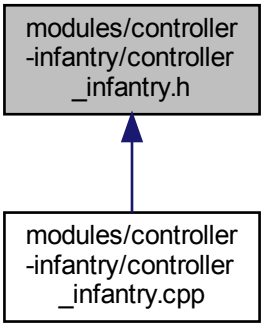
7.12 modules/controller-infantry/controller_infantry.h File Reference

```
#include "controller-base/controller_factory.h"
```

Include dependency graph for controller_infantry.h:



This graph shows which files directly or indirectly include this file:



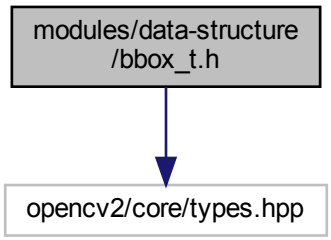
Classes

- class **InfantryController**

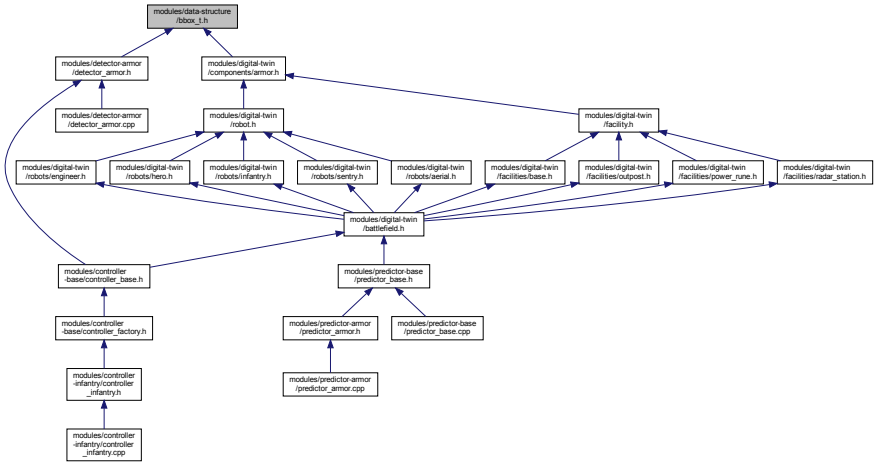
7.13 modules/data-structure/bbox_t.h File Reference

```
#include <opencv2/core/types.hpp>
```

Include dependency graph for bbox_t.h:



This graph shows which files directly or indirectly include this file:



Classes

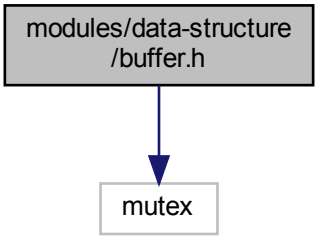
- struct **bbox_t**

  4-point model structure for detection result.

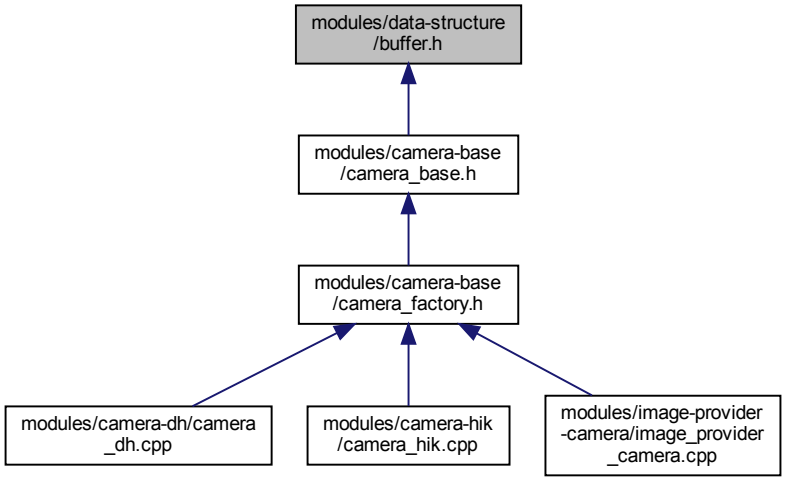
7.14 modules/data-structure/buffer.h File Reference

```
#include <mutex>
```

Include dependency graph for buffer.h:



This graph shows which files directly or indirectly include this file:



Classes

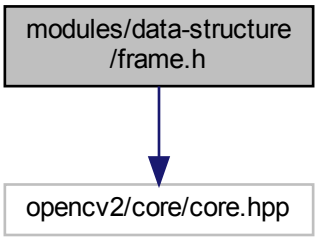
- class **CircularBuffer**< **Type, size** >

  Circular buffer with mutex.

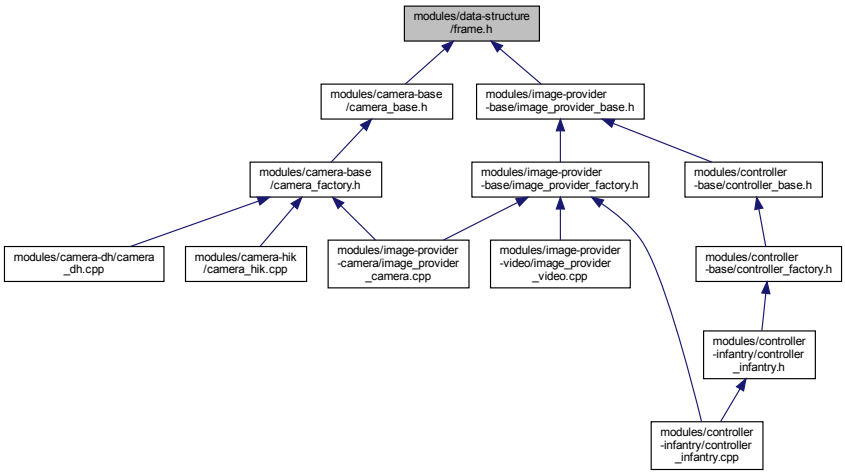
7.15 modules/data-structure/frame.h File Reference

```
#include <opencv2/core/core.hpp>
```

Include dependency graph for frame.h:



This graph shows which files directly or indirectly include this file:



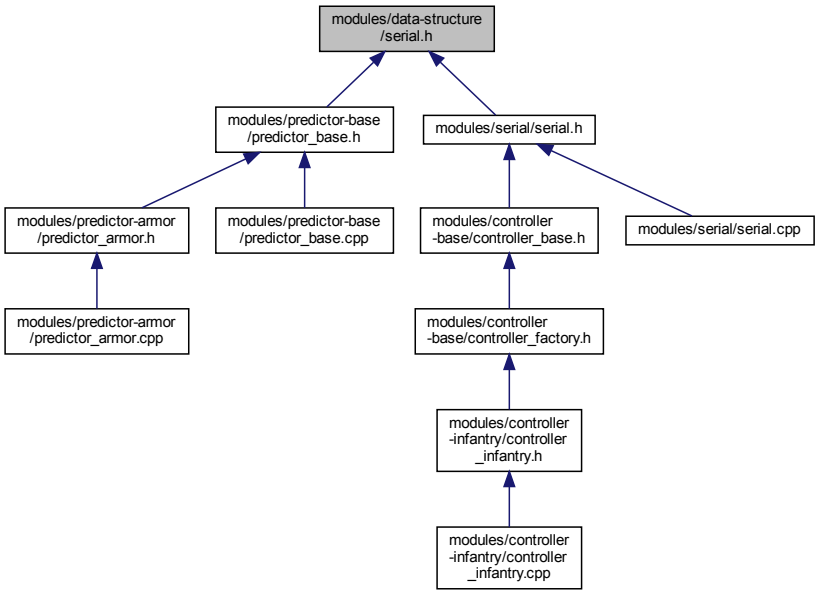
Classes

- struct **Frame**

  Single frame structure.

7.16 modules/data-structure/serial.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- struct **SendPacket**

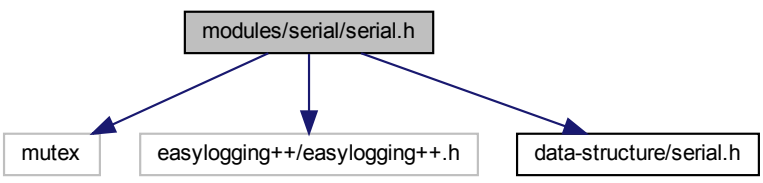
  Packet format to send.
- struct **ReceivePacket**

  Packet format received.

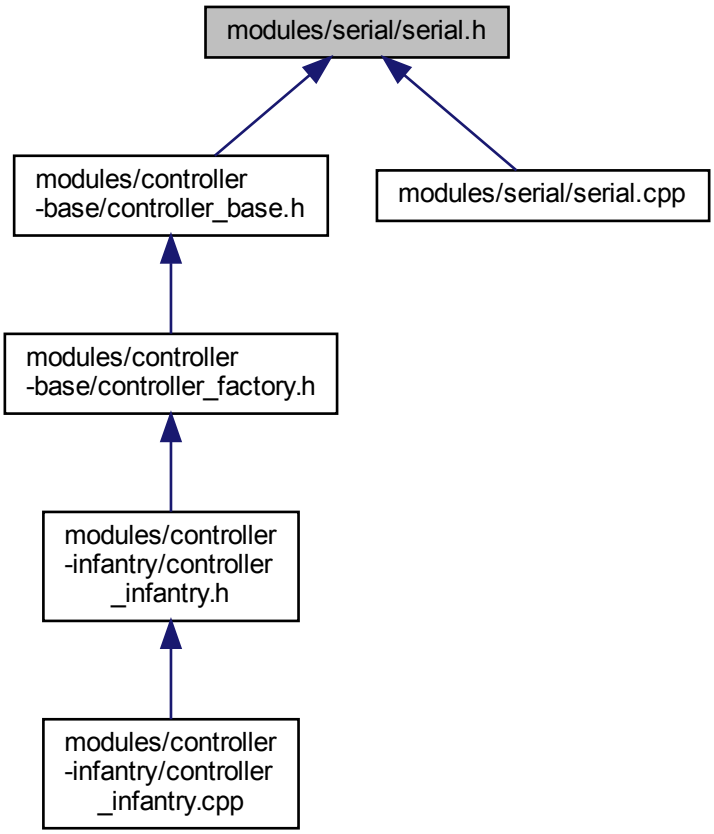
7.17 modules/serial/serial.h File Reference

```
#include <mutex>
#include "easylogging++/easylogging++.h"
#include "data-structure/serial.h"
```

Include dependency graph for serial.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **Serial**

  Serial *(p. 136) manager class.*

Macros

- #define **_SERIAL_RECEIVING_TIMEOUT_MS_** 10

  Receiving thread time out. (millisecond)
- #define **_SERIAL_SENDING_TIMEOUT_MS_** 10

  Sending thread time out. (millisecond)

7.17.1 Macro Definition Documentation

7.17.1.1 _SERIAL_RECEIVING_TIMEOUT_MS_

```
#define _SERIAL_RECEIVING_TIMEOUT_MS_ 10
```

Receiving thread time out. (millisecond)

Serial (p. 136) class header.

Author

trantuan-20048607, anonymity

Date

2022.1.28

Include this file to use serial communication.

Definition at line 15 of file serial.h.

7.17.1.2 _SERIAL_SENDING_TIMEOUT_MS_

```
#define _SERIAL_SENDING_TIMEOUT_MS_ 10
```

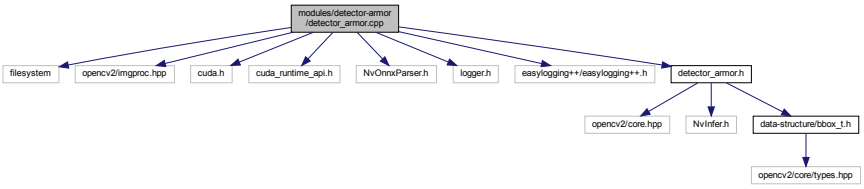
Sending thread time out. (millisecond)

Definition at line 16 of file serial.h.

7.18 modules/detector-armor/detector_armor.cpp File Reference

```
#include <filesystem>
#include <opencv2/imgproc.hpp>
#include <cuda.h>
#include <cuda_runtime_api.h>
#include <NvOnnxParser.h>
#include <logger.h>
#include "easylogging++/easylogging++.h"
#include "detector_armor.h"
```

Include dependency graph for detector_armor.cpp:



Macros

- #define **__TRT_ASSERT__**(expr)

Functions

- template<class F , class T , class ... Ts>
  T **reduce** (F &&func, T x, Ts... xs)
- template<class T , class ... Ts>
  T **reduce_max** (T x, Ts... xs)
- template<class T , class ... Ts>
  T **reduce_min** (T x, Ts... xs)
- constexpr float **inv_sigmoid** (float x)
- constexpr float **sigmoid** (float x)

7.18.1 Macro Definition Documentation

7.18.1.1 __TRT_ASSERT__

```
#define __TRT_ASSERT__(
              expr )
```

Value:

```
    if(!(expr)) {                                                   \
        LOG(ERROR) « "TensorRT assertion failed: " « #expr « ".";  \
        exit(-1);                                                   \
    }
```

Definition at line 10 of file detector_armor.cpp.

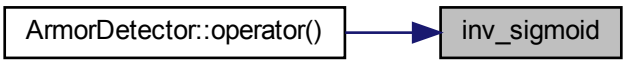
7.18.2 Function Documentation

7.18.2.1 inv_sigmoid()

```
constexpr float inv_sigmoid (
            float x )  [inline], [constexpr]
```

Definition at line 64 of file detector_armor.cpp.

Here is the caller graph for this function:

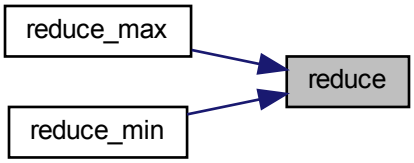


7.18.2.2 reduce()

```
template<class F , class T , class ...  Ts>
T reduce (
            F && func,
            T x,
            Ts...  xs )
```

Definition at line 23 of file detector_armor.cpp.

Here is the caller graph for this function:

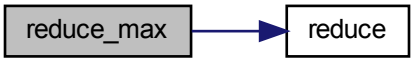


7.18.2.3 reduce_max()

```
template<class T , class ...  Ts>
T reduce_max (
            T x,
            Ts...  xs )
```

Definition at line 32 of file detector_armor.cpp.

Here is the call graph for this function:

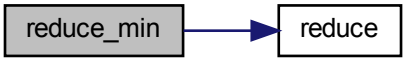


7.18.2.4 reduce_min()

```
template<class T , class ...  Ts>
T reduce_min (
            T x,
            Ts...  xs )
```

Definition at line 37 of file detector_armor.cpp.

Here is the call graph for this function:

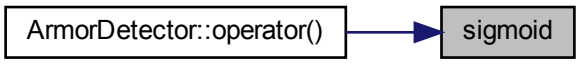


7.18.2.5 sigmoid()

```
constexpr float sigmoid (
            float x )  [inline], [constexpr]
```

Definition at line 68 of file detector_armor.cpp.

Here is the caller graph for this function:

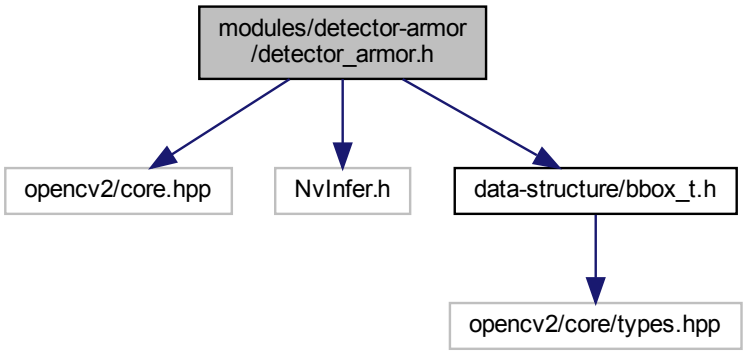


7.19 modules/detector-armor/detector_armor.h File Reference

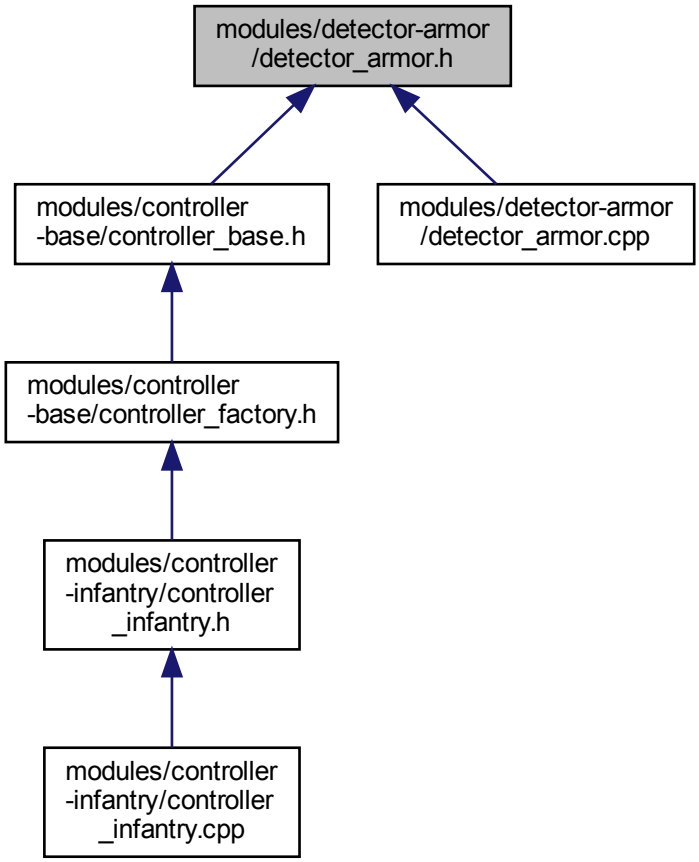
```
#include <opencv2/core.hpp>
#include <NvInfer.h>
```

```
#include "data-structure/bbox_t.h"
```

Include dependency graph for detector_armor.h:



This graph shows which files directly or indirectly include this file:



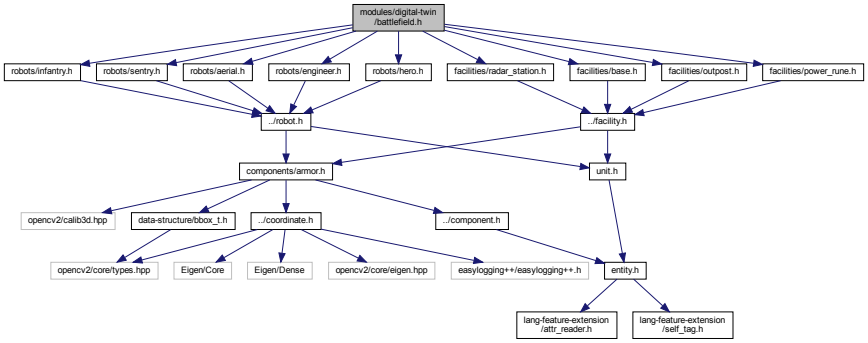
Classes

- class **ArmorDetector**

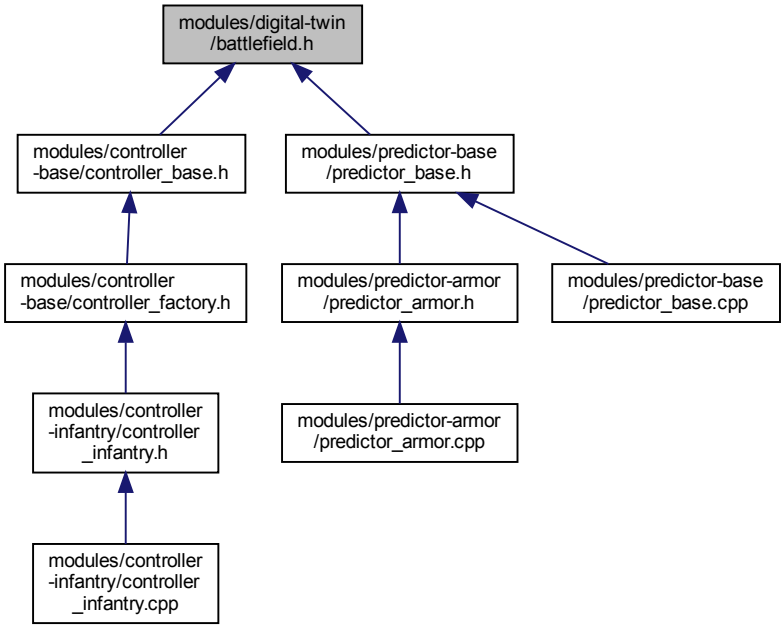
7.20 modules/digital-twin/battlefield.h File Reference

```
#include "robots/engineer.h"
#include "robots/hero.h"
#include "robots/infantry.h"
#include "robots/sentry.h"
#include "robots/aerial.h"
#include "facilities/base.h"
#include "facilities/outpost.h"
#include "facilities/power_rune.h"
#include "facilities/radar_station.h"
```

Include dependency graph for battlefield.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **Battlefield**

  Battlefield (p. *26) simulation data model.*

Macros

- #define **ARMOR_CONFIDENCE_HIGH_THRESH** 0.8
- #define **ARMOR_CONFIDENCE_LOW_THRESH** 0.6
- #define **ARMOR_SQUARED_CENTER_DISTANCE_THRESH** 144
- #define **ARMOR_SQUARED_TRANSLATION_VECTOR_WORLD_DISTANCE_THRESH** 0.125
- #define **ADD_ARMOR_TO_ROBOT**(_type, _class)
- #define **ADD_ARMOR_TO_FACILITY**(_type, _class)

7.20.1 Macro Definition Documentation

7.20.1.1 ADD_ARMOR_TO_FACILITY

```
#define ADD_ARMOR_TO_FACILITY(
              _type,
              _class )
```

Value:

```
    if (facilities_[armor.Color()][_type].get() == nullptr)
            \
        facilities_[armor.Color()][_type] = std::make_shared<_class>(armor.Color(), 0, _type);
            \
    if (armor.Confidence() > ARMOR_CONFIDENCE_HIGH_THRESH) {
            \
        facilities_[armor.Color()][_type].get()->AddBottomArmor(armor);
            \
    } else if (last_battlefield_data_.facilities_[armor.Color()][_type].get() != nullptr
            \
                && armor.Confidence() > ARMOR_CONFIDENCE_LOW_THRESH) {
            \
        for (auto &armor_last:
       last_battlefield_data_.facilities_[armor.Color()][_type].get()->BottomArmors()) {  \
            auto center_delta = armor.Center() - armor_last.Center();
            \
            auto tv_world_delta = armor.TranslationVectorWorld() -
            \
                                  armor_last.TranslationVectorWorld();
            \
            if (center_delta.dot(center_delta) < ARMOR_SQUARED_CENTER_DISTANCE_THRESH &&
            \
                tv_world_delta.norm() < ARMOR_SQUARED_TRANSLATION_VECTOR_WORLD_DISTANCE_THRESH) {
            \
                facilities_[armor.Color()][_type].get()->AddBottomArmor(armor);
            \
            }
            \
        }
            \
    }
            \
    break;
```

Definition at line 64 of file battlefield.h.

7.20.1.2 ADD_ARMOR_TO_ROBOT

```
#define ADD_ARMOR_TO_ROBOT(
              _type,
              _class )
```

Value:

```
    if (robots_[armor.Color()][_type].get() == nullptr)                                          \
        robots_[armor.Color()][_type] = std::make_shared<_class>(armor.Color(), 0, _type);       \
    if (armor.Confidence() > ARMOR_CONFIDENCE_HIGH_THRESH) {                                     \
        robots_[armor.Color()][_type].get()->AddArmor(armor);                                    \
    } else if (last_battlefield_data_.robots_[armor.Color()][_type].get() != nullptr             \
                && armor.Confidence() > ARMOR_CONFIDENCE_LOW_THRESH) {                           \
        for (auto &armor_last:                                                                   \
                last_battlefield_data_.robots_[armor.Color()][_type].get()->Armors()) {          \
            auto center_delta = armor.Center() - armor_last.Center();                            \
            auto tv_world_delta = armor.TranslationVectorWorld() -                               \
                                  armor_last.TranslationVectorWorld();                           \
            if (center_delta.dot(center_delta) < ARMOR_SQUARED_CENTER_DISTANCE_THRESH &&          \
                tv_world_delta.norm() < ARMOR_SQUARED_TRANSLATION_VECTOR_WORLD_DISTANCE_THRESH) {  \
                robots_[armor.Color()][_type].get()->AddArmor(armor);                            \
            }                                                                                    \
        }                                                                                        \
    }                                                                                            \
    break;
```

Definition at line 43 of file battlefield.h.

7.20.1.3 ARMOR_CONFIDENCE_HIGH_THRESH

```
#define ARMOR_CONFIDENCE_HIGH_THRESH 0.8
```

Digital twin main header.

Author

trantuan-20048607, screw-44

Date

2022.1.28

Include this file in digital-twin module for complete function.

Definition at line 21 of file battlefield.h.

7.20.1.4 ARMOR_CONFIDENCE_LOW_THRESH

```
#define ARMOR_CONFIDENCE_LOW_THRESH 0.6
```

Definition at line 22 of file battlefield.h.

7.20.1.5 ARMOR_SQUARED_CENTER_DISTANCE_THRESH

```
#define ARMOR_SQUARED_CENTER_DISTANCE_THRESH 144
```

Definition at line 23 of file battlefield.h.

7.20.1.6 ARMOR_SQUARED_TRANSLATION_VECTOR_WORLD_DISTANCE_THRESH

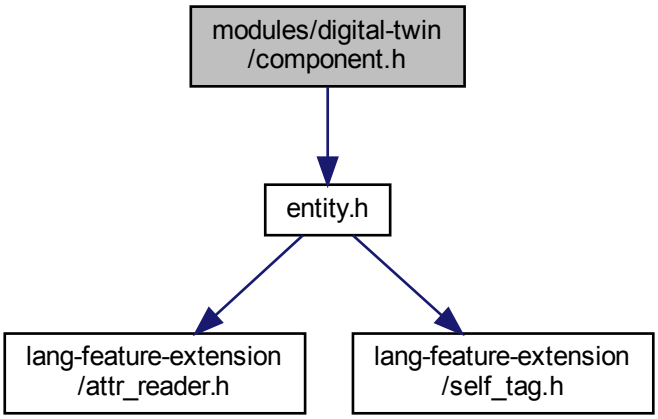
```
#define ARMOR_SQUARED_TRANSLATION_VECTOR_WORLD_DISTANCE_THRESH 0.125
```

Definition at line 24 of file battlefield.h.

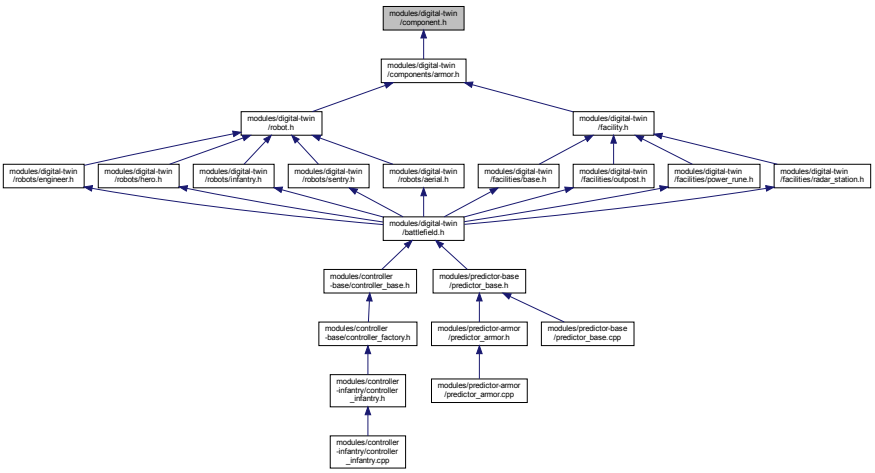
7.21 modules/digital-twin/component.h File Reference

```
#include "entity.h"
```

Include dependency graph for component.h:



This graph shows which files directly or indirectly include this file:



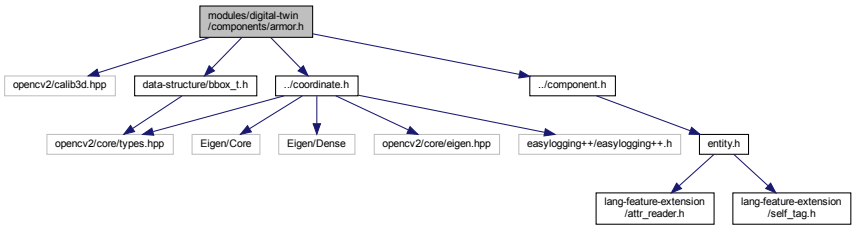
Classes

- class **Component**

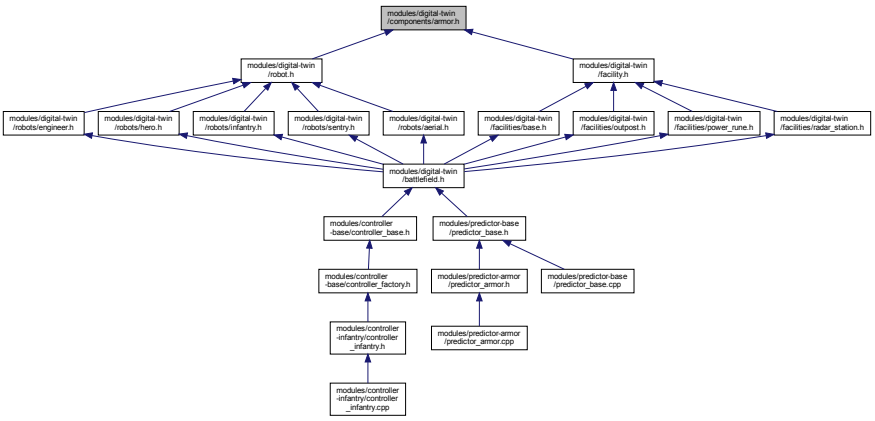
7.22 modules/digital-twin/components/armor.h File Reference

```
#include <opencv2/calib3d.hpp>
#include "data-structure/bbox_t.h"
#include "../coordinate.h"
#include "../component.h"
```

Include dependency graph for armor.h:



This graph shows which files directly or indirectly include this file:



Classes

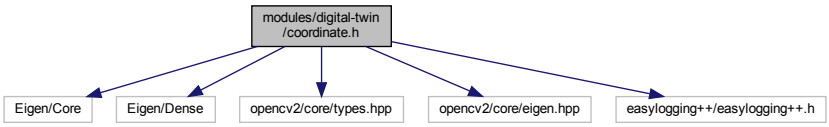
- class **Armor**

7.23 modules/digital-twin/coordinate.h File Reference

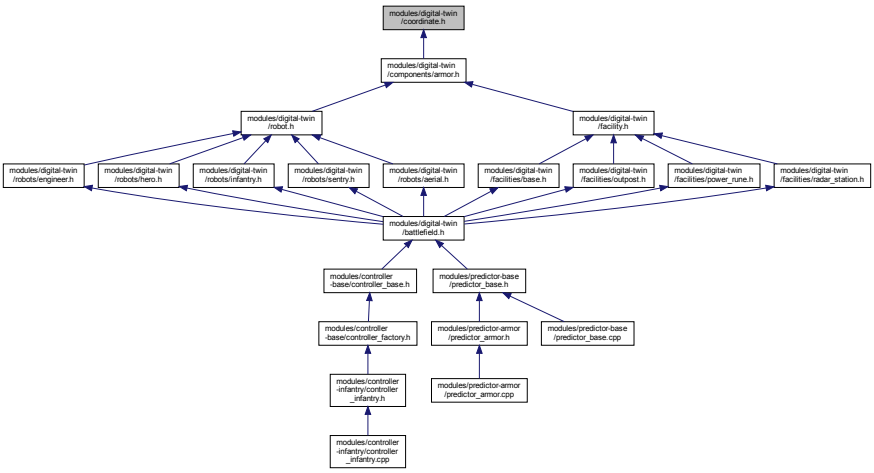
```
#include <Eigen/Core>
#include <Eigen/Dense>
#include <opencv2/core/types.hpp>
#include <opencv2/core/eigen.hpp>
```

```
#include "easylogging++/easylogging++.h"
```

Include dependency graph for coordinate.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- **coordinate**
- **coordinate::transform**

Macros

- #define **EXP_ACCELERATE_Q2R** true

Typedefs

- typedef Eigen::Vector3d **coordinate::TranslationVector**
- typedef Eigen::Vector3d **coordinate::RotationVector**
- typedef Eigen::Matrix< double, 3, 1 > **coordinate::TranslationMatrix**
- typedef Eigen::Matrix3d **coordinate::RotationMatrix**
- typedef Eigen::Quaternionf **coordinate::Quaternion**

Functions

- RotationMatrix **coordinate::transform::QuaternionToRotationMatrix** (const Quaternion &quaternion)
- TranslationVector **coordinate::transform::CameraToWorld** (const TranslationVector &tv_cam, const RotationMatrix &rm_imu, const TranslationMatrix &tm_cam_to_imu, const RotationMatrix &rm_cam_to↩ _imu)
- TranslationVector **coordinate::transform::WorldToCamera** (const TranslationVector &tv_world, const RotationMatrix &rm_imu_to_world, const TranslationMatrix &tm_cam_to_imu, const RotationMatrix &rm_↩ cam_to_imu)

7.23.1 Macro Definition Documentation

7.23.1.1 EXP_ACCELERATE_Q2R

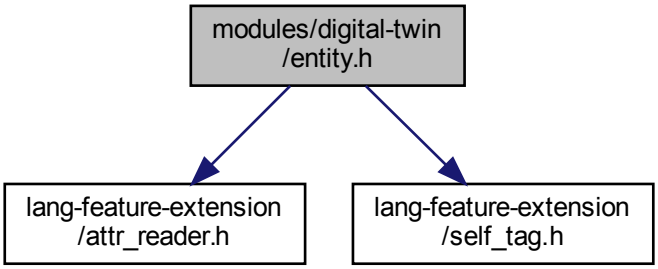
```
#define EXP_ACCELERATE_Q2R true
```

Definition at line 30 of file coordinate.h.

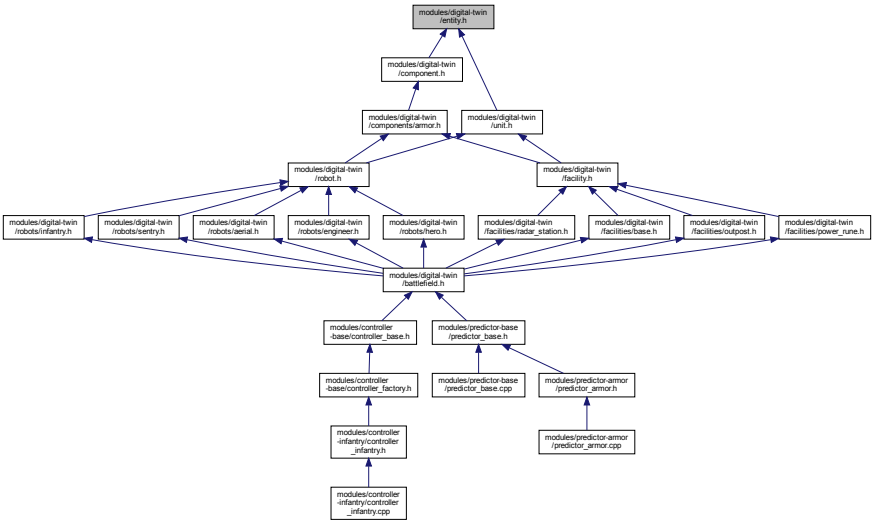
7.24 modules/digital-twin/entity.h File Reference

```
#include "lang-feature-extension/attr_reader.h"
#include "lang-feature-extension/self_tag.h"
```

Include dependency graph for entity.h:



This graph shows which files directly or indirectly include this file:



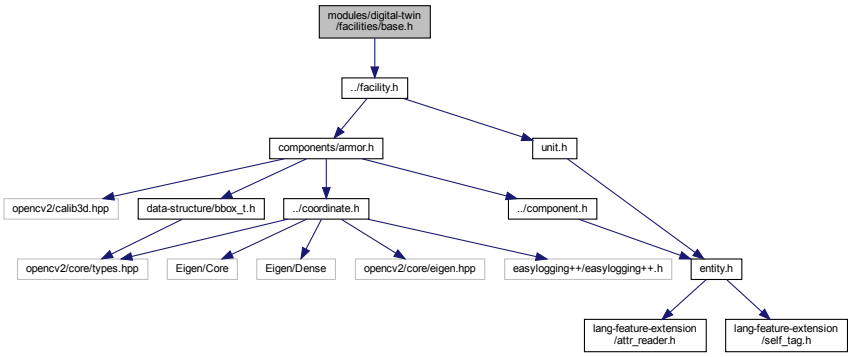
Classes

- class **Entity**

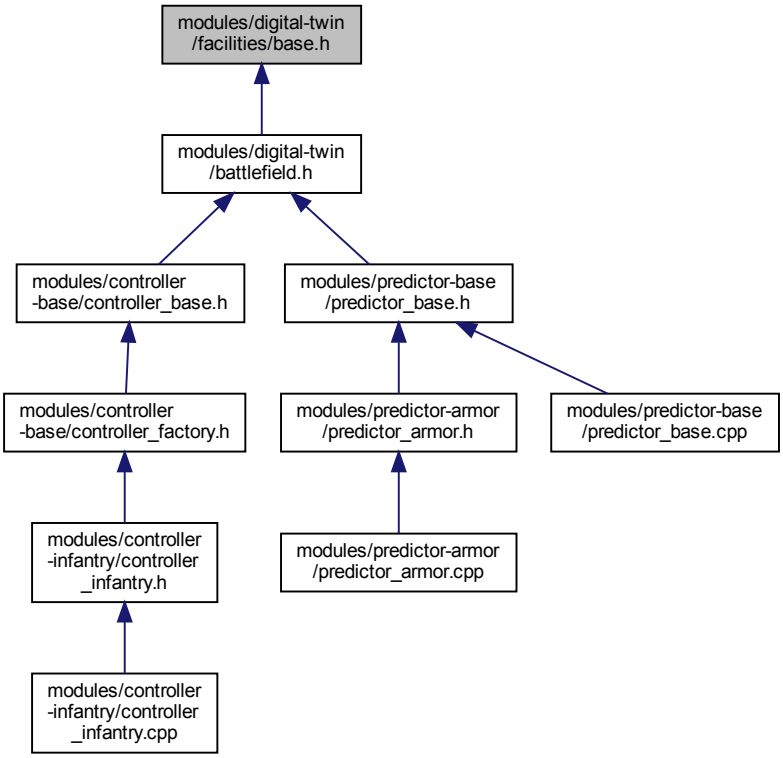
7.25 modules/digital-twin/facilities/base.h File Reference

```
#include "../facility.h"
```

Include dependency graph for base.h:



This graph shows which files directly or indirectly include this file:

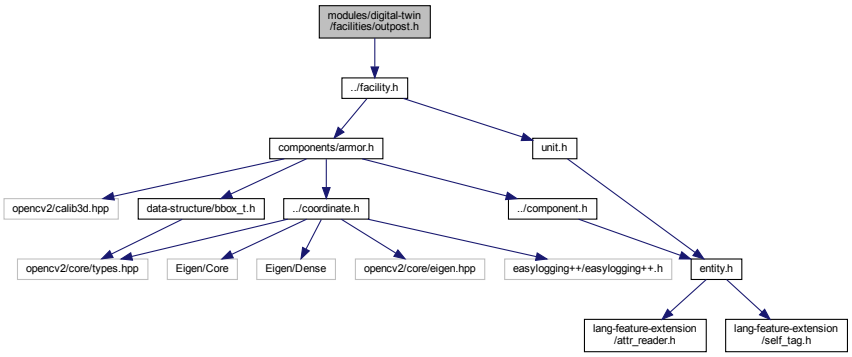


Classes

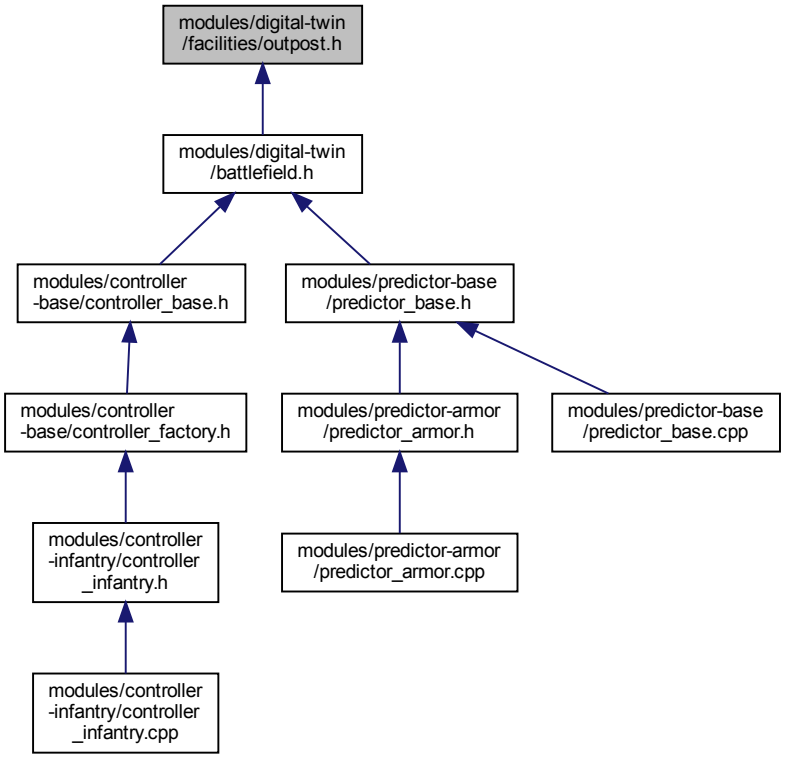
- class **Base**

7.26 modules/digital-twin/facilities/outpost.h File Reference

`#include "../facility.h"`
Include dependency graph for outpost.h:



This graph shows which files directly or indirectly include this file:



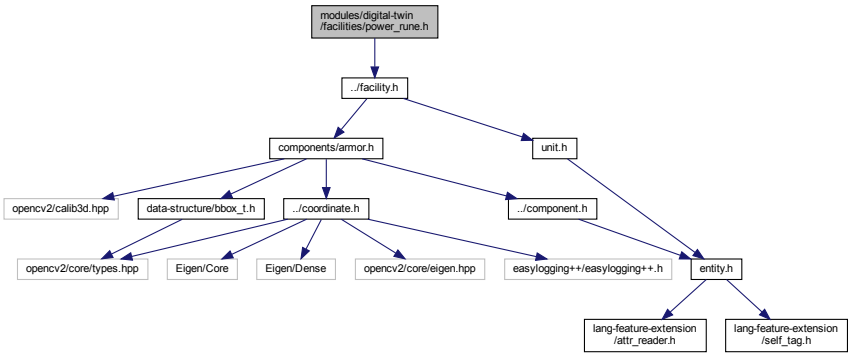
Classes

- class **Outpost**

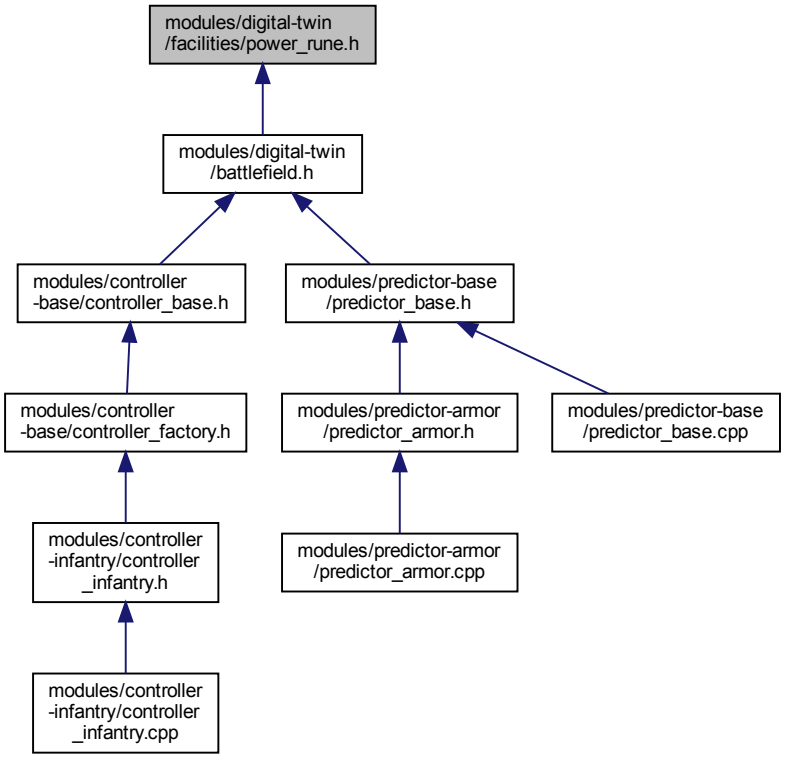
7.27 modules/digital-twin/facilities/power_rune.h File Reference

```
#include "../facility.h"
```

Include dependency graph for power_rune.h:



This graph shows which files directly or indirectly include this file:



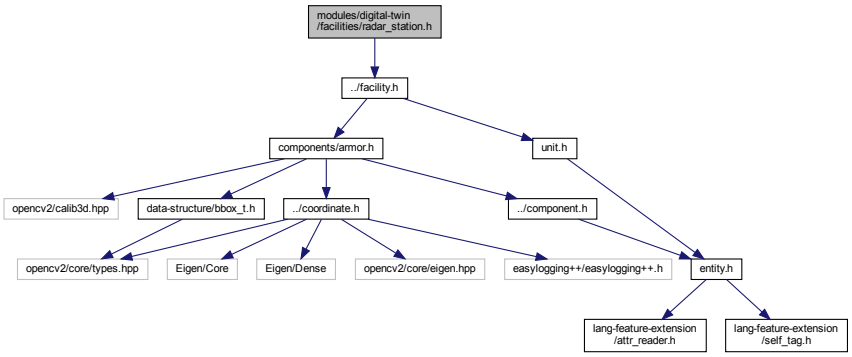
Classes

- class **PowerRune**

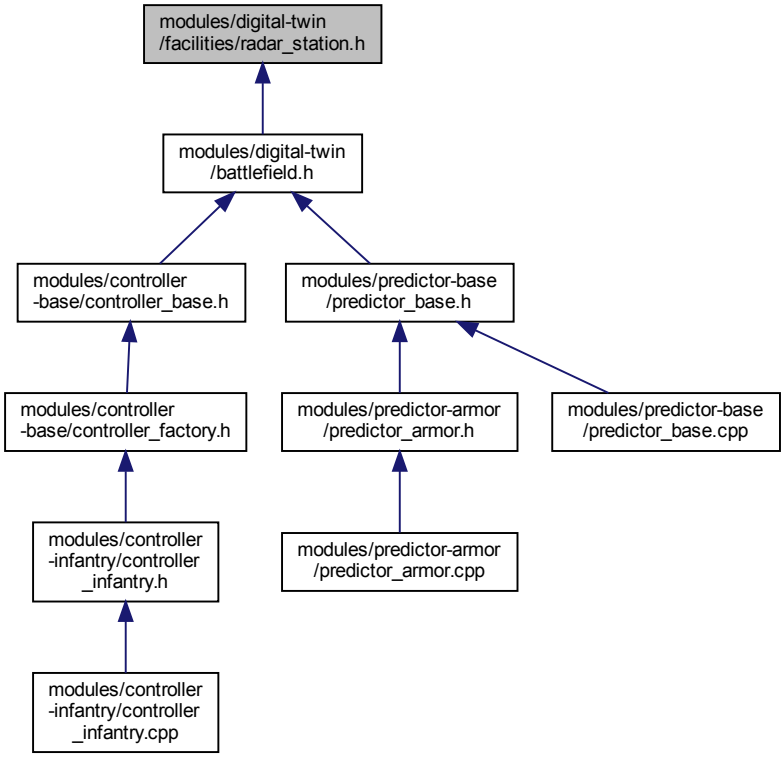
7.28 modules/digital-twin/facilities/radar_station.h File Reference

```
#include "../facility.h"
```

Include dependency graph for radar_station.h:



This graph shows which files directly or indirectly include this file:



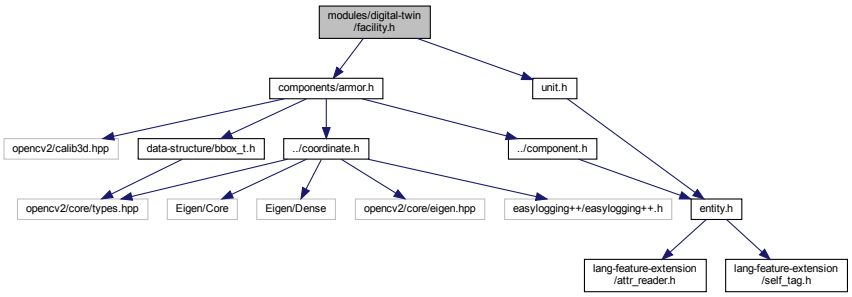
Classes

- class **RadarStation**

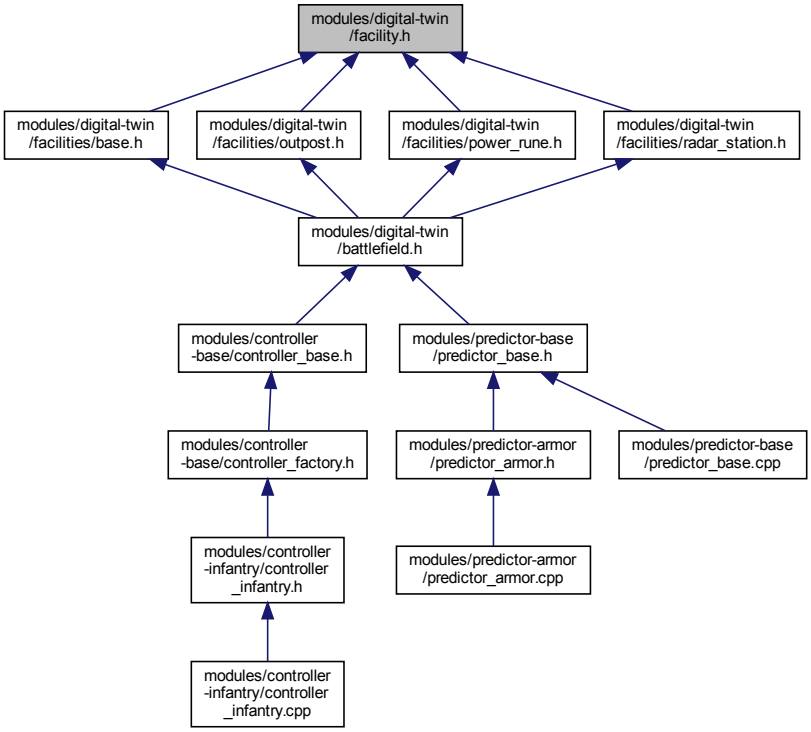
7.29 modules/digital-twin/facility.h File Reference

```
#include "components/armor.h"
#include "unit.h"
```

Include dependency graph for facility.h:



This graph shows which files directly or indirectly include this file:



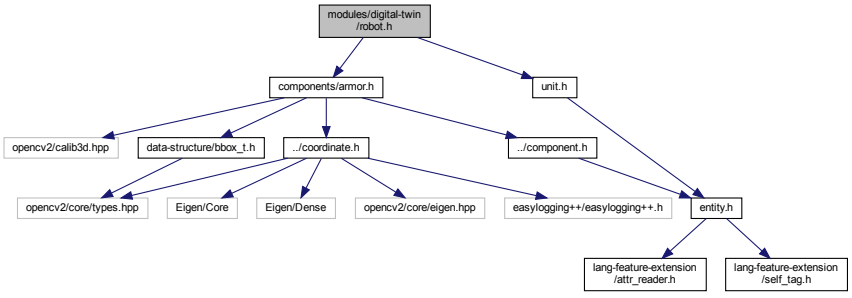
Classes

- class **Facility**

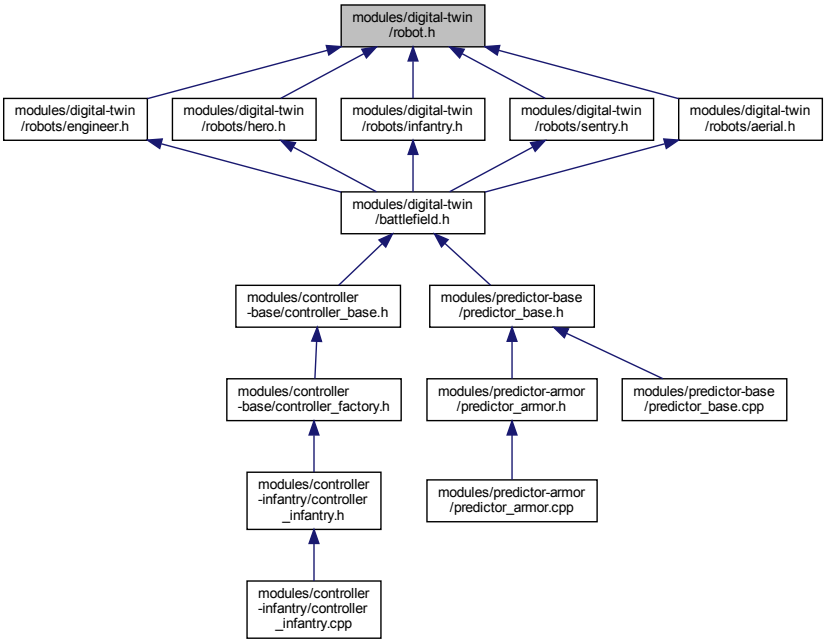
7.30 modules/digital-twin/robot.h File Reference

```
#include "components/armor.h"
#include "unit.h"
```

Include dependency graph for robot.h:



This graph shows which files directly or indirectly include this file:

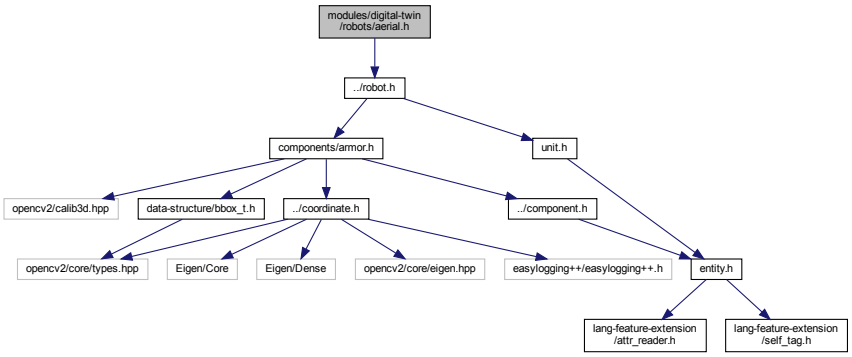


Classes

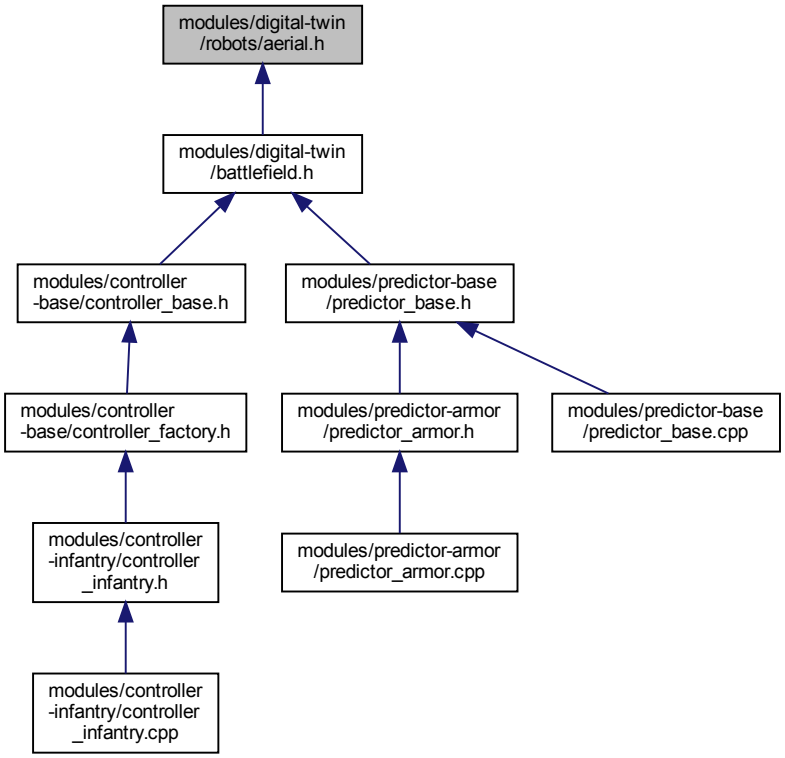
- class **Robot**

7.31 modules/digital-twin/robots/aerial.h File Reference

`#include "../robot.h"`
Include dependency graph for aerial.h:



This graph shows which files directly or indirectly include this file:

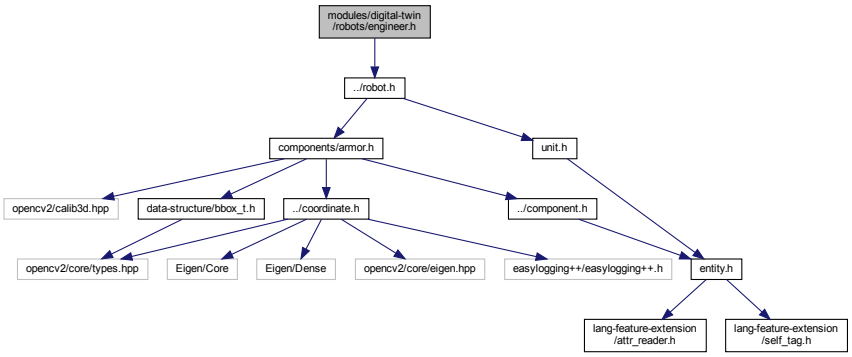


Classes

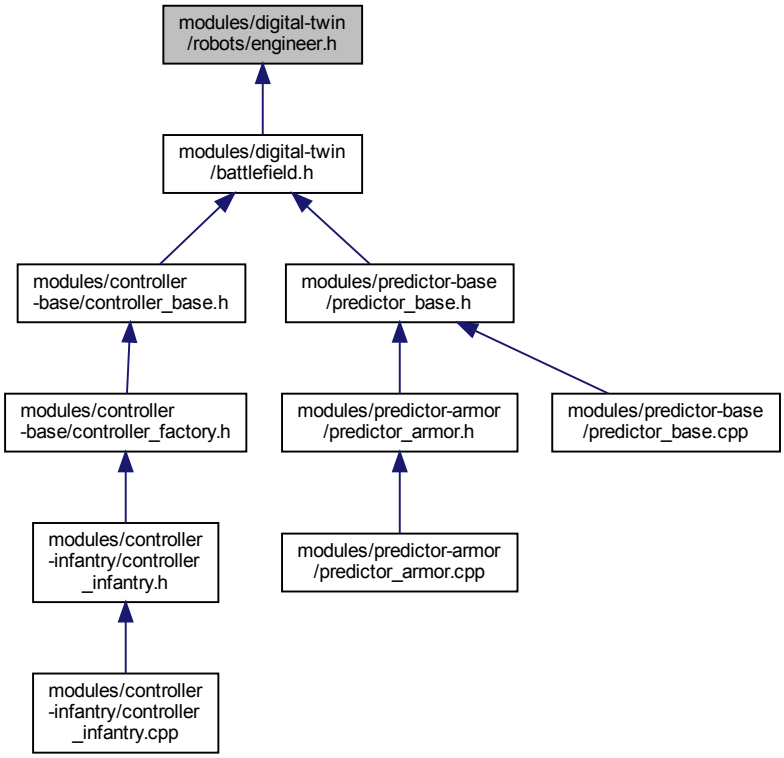
- class **Aerial**

7.32 modules/digital-twin/robots/engineer.h File Reference

`#include "../robot.h"`
Include dependency graph for engineer.h:



This graph shows which files directly or indirectly include this file:

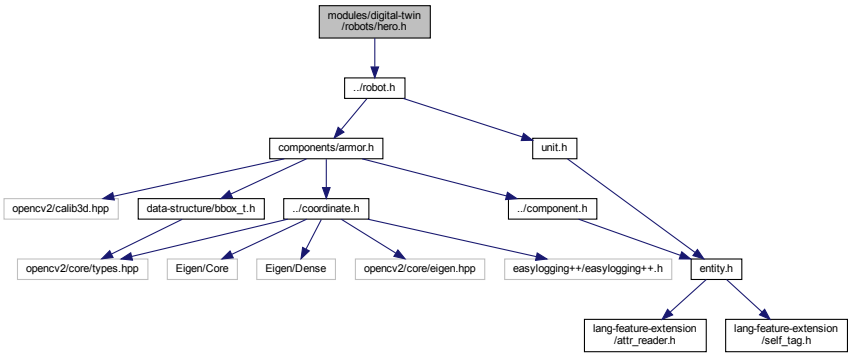


Classes

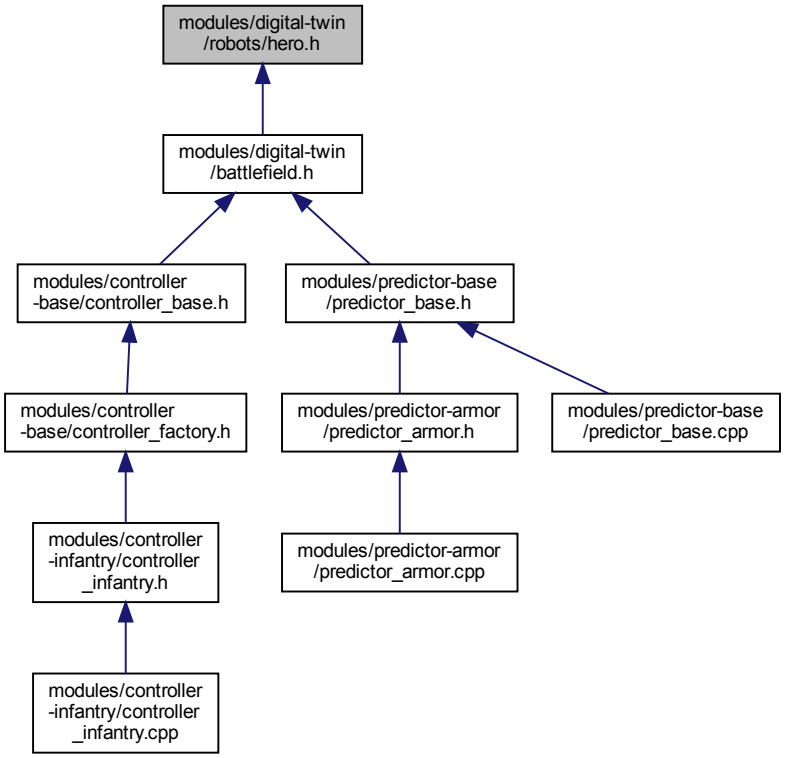
- class **Engineer**

7.33 modules/digital-twin/robots/hero.h File Reference

`#include "../robot.h"`
Include dependency graph for hero.h:



This graph shows which files directly or indirectly include this file:

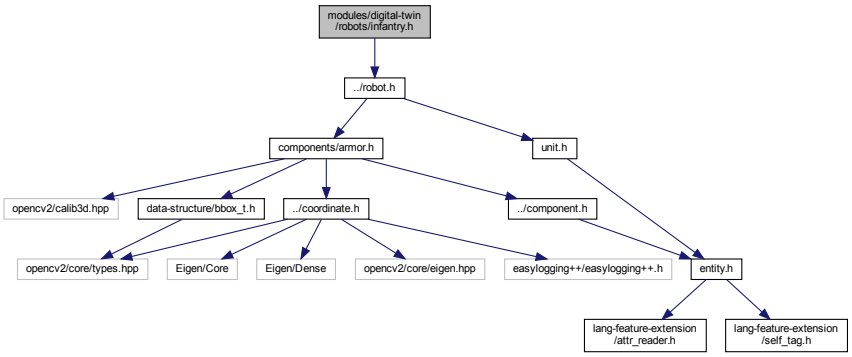


Classes

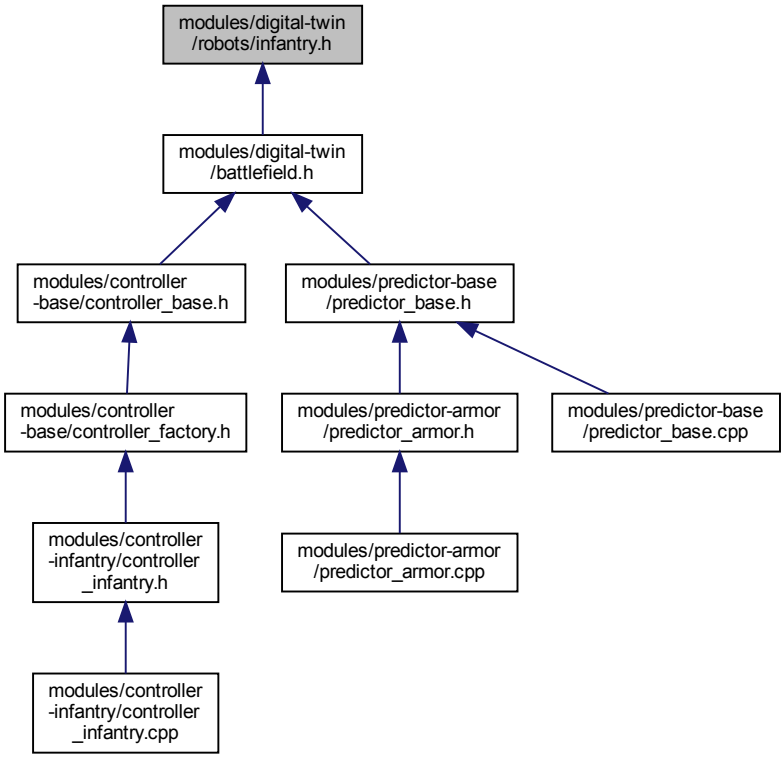
- class **Hero**

7.34 modules/digital-twin/robots/infantry.h File Reference

`#include "../robot.h"`
Include dependency graph for infantry.h:



This graph shows which files directly or indirectly include this file:

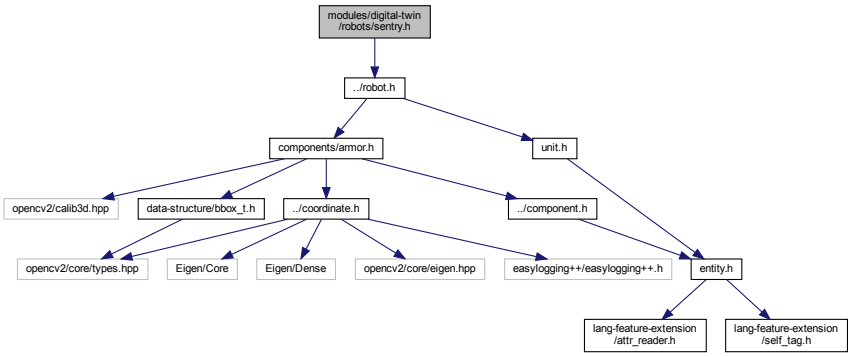


Classes

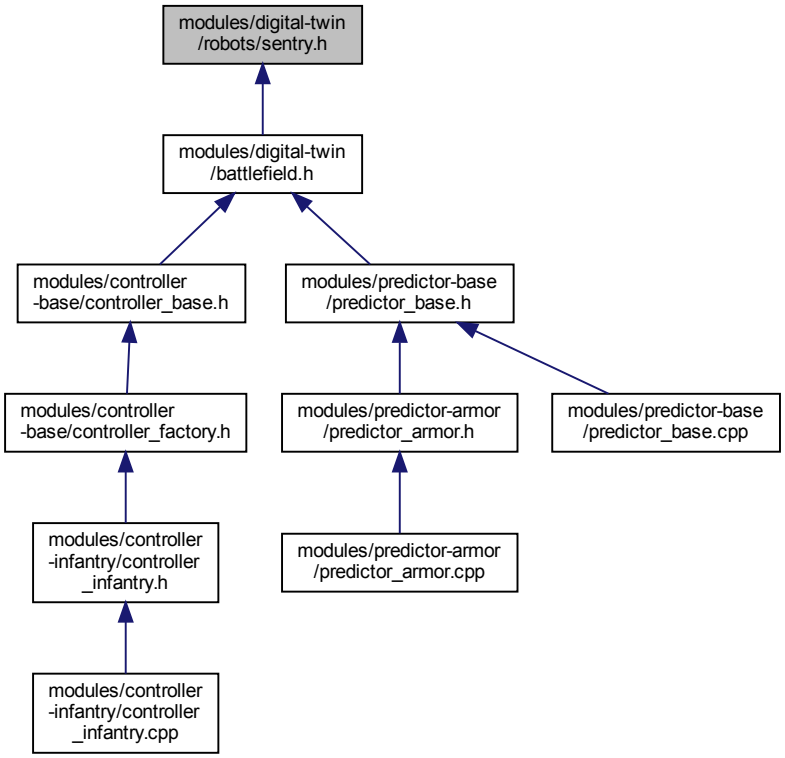
- class **Infantry**

7.35 modules/digital-twin/robots/sentry.h File Reference

`#include "../robot.h"`
Include dependency graph for sentry.h:



This graph shows which files directly or indirectly include this file:



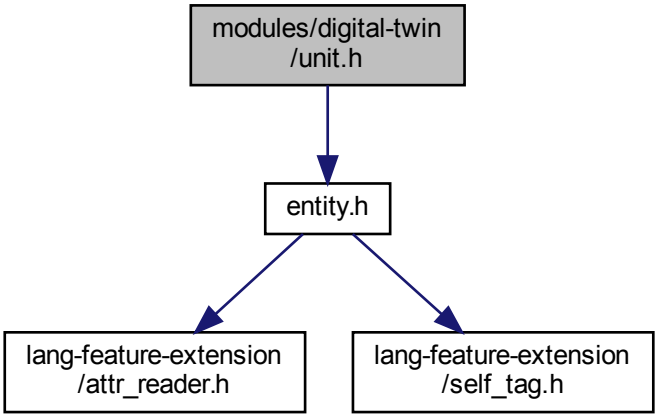
Classes

- class **Sentry**

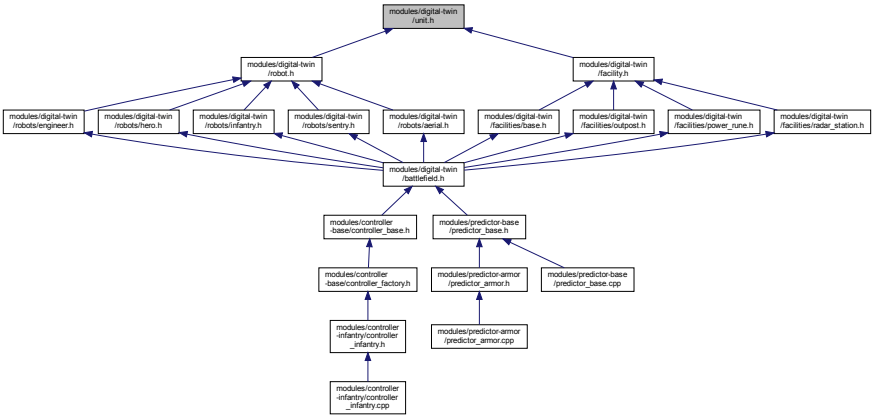
7.36 modules/digital-twin/unit.h File Reference

```
#include "entity.h"
```

Include dependency graph for unit.h:



This graph shows which files directly or indirectly include this file:



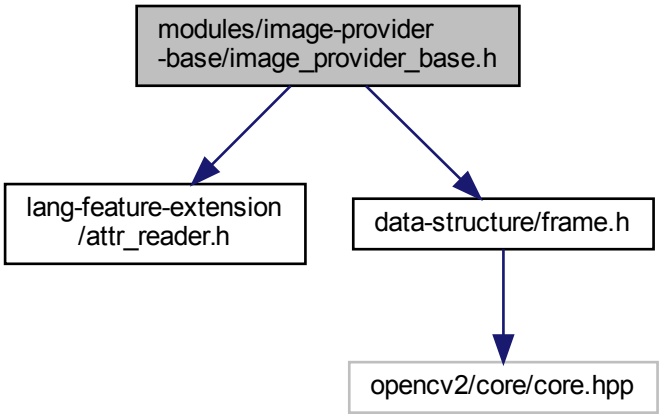
Classes

- class **Unit**

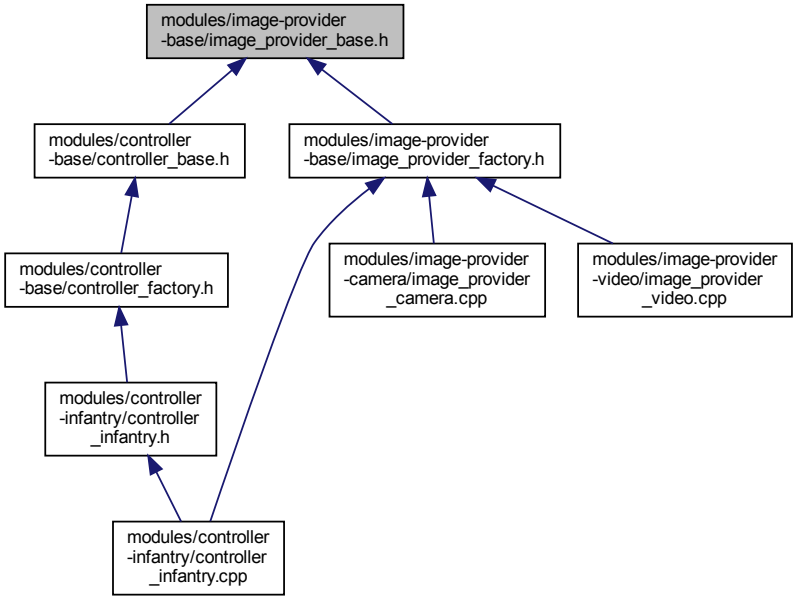
7.37 modules/image-provider-base/image_provider_base.h File Reference

```
#include "lang-feature-extension/attr_reader.h"
#include "data-structure/frame.h"
```

Include dependency graph for image_provider_base.h:



This graph shows which files directly or indirectly include this file:



Classes

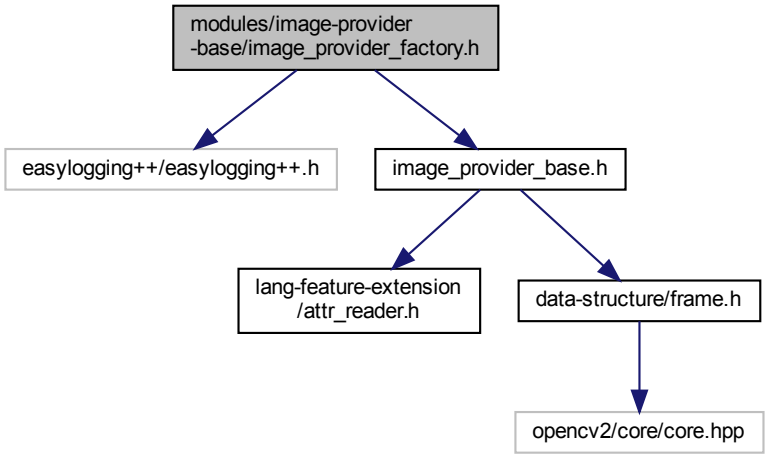
- class **ImageProvider**

  Image provider base class.

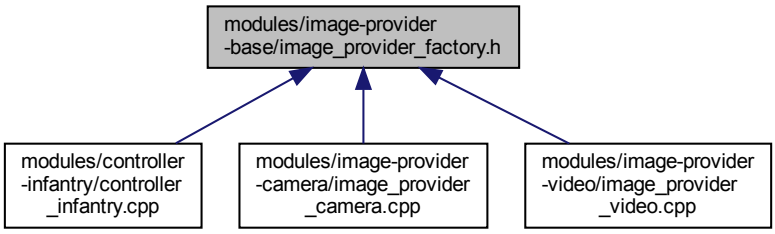
7.38 modules/image-provider-base/image_provider_factory.h File Reference

```
#include "easylogging++/easylogging++.h"
#include "image_provider_base.h"
```

Include dependency graph for image_provider_factory.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **ImageProviderRegistryBase**

  Base *(p. 25) class of image provider registry.*
- class **ImageProviderFactory**

  Singleton image provider factory.
- class **ImageProviderRegistry**< **IPType** >

  Templated image provider registry class.

Macros

- #define **__CREATE_IMAGE_PROVIDER__**(ip_type_name) **ImageProviderFactory::Instance**().Create↩ ImageProvider(ip_type_name)

  *A macro to create an image provider of specified type name. For details, turn to class **ImageProviderFactory** (p. 105).*

7.38.1 Macro Definition Documentation

7.38.1.1 __CREATE_IMAGE_PROVIDER__

```
#define __CREATE_IMAGE_PROVIDER__(
              ip_type_name )  ImageProviderFactory::Instance().CreateImageProvider(ip_type_↩
name)
```

A macro to create an image provider of specified type name. For details, turn to class **ImageProviderFactory** (p. 105).

Image provider factory header.

Author

trantuan-20048607

Date

2022.1.28

Include this file to create image provider objects.

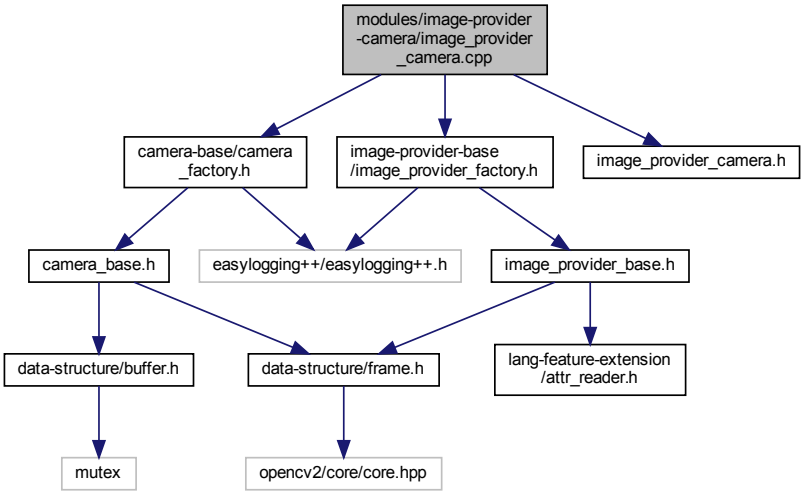
Definition at line 15 of file image_provider_factory.h.

7.39 modules/image-provider-camera/image_provider_camera.cpp File Reference

```
#include "camera-base/camera_factory.h"
#include "image-provider-base/image_provider_factory.h"
```

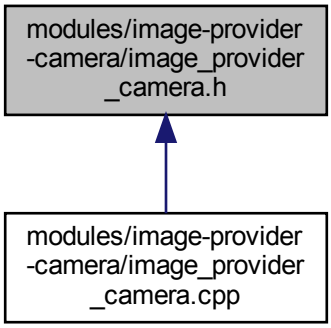
```
#include "image_provider_camera.h"
```

Include dependency graph for image_provider_camera.cpp:



7.40 modules/image-provider-camera/image_provider_camera.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

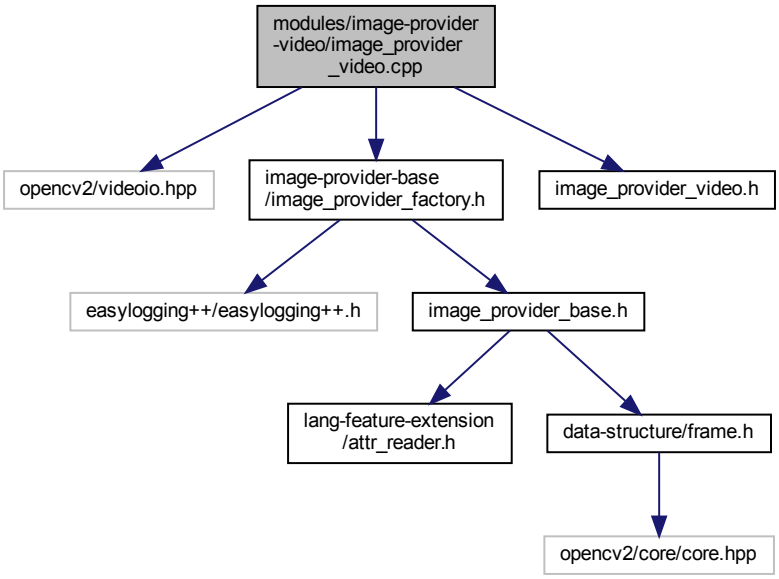
- class **ImageProviderCamera**

  Camera *(p. 30) image provider class implementation.*

7.41 modules/image-provider-video/image_provider_video.cpp File Reference

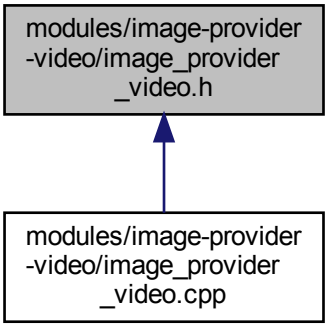
```
#include <opencv2/videoio.hpp>
#include "image-provider-base/image_provider_factory.h"
#include "image_provider_video.h"
```

Include dependency graph for image_provider_video.cpp:



7.42 modules/image-provider-video/image_provider_video.h File Reference

This graph shows which files directly or indirectly include this file:



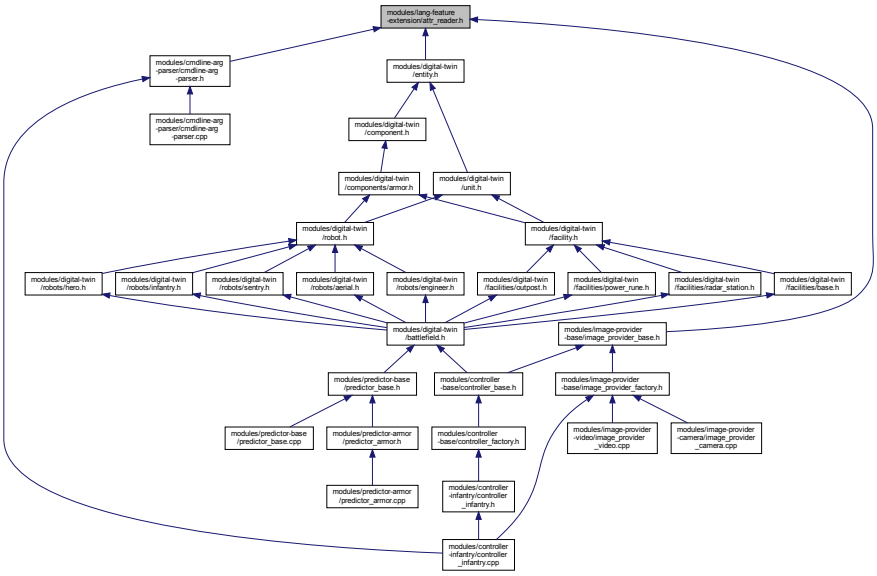
Classes

- class **ImageProviderVideo**

  Video image provider class implementation.

7.43 modules/lang-feature-extension/attr_reader.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define **__ATTR_READER__**(_var, _func) inline auto _func() const { return _var; }

  *Use of **ATTR_READER**(var_name, func_name) will create a function named func_name which only returns var_↩ name, corresponding "attr_reader" in ruby.*

- #define **__ATTR_READER_REF__**(_var, _func) inline auto &_func() const { return _var; }

  *Use of **ATTR_READER_REF**(var_name, func_name) will create a function named func_name which only returns a read-only reference of var_name, corresponding "attr_reader" in ruby.*

7.43.1 Macro Definition Documentation

7.43.1.1 __ATTR_READER__

```
#define __ATTR_READER__(
              _var,
              _func ) inline auto _func() const { return _var; }
```

Use of **ATTR_READER**(var_name, func_name) will create a function named func_name which only returns var_↩ name, corresponding "attr_reader" in ruby.

Attribute reader method generator for C++.

Author

trantuan-20048607

Date

2022.1.14

Introduced from ruby programming language, attribute reader generator is used to quickly generate a function to READ a private attribute in a class.

Use this macro in a class to generate a reader function for a private variable:

```
class Foo() {
public:
    __ATTR_READER__(private_var_, PrivateVar)  // No need to add a semicolon.
private:
    int private_var_{};
}
int main() {
    Foo bar;
    int private_var = bar.PrivateVar();  // Get the value.
}
```

Definition at line 31 of file attr_reader.h.

7.43.1.2 __ATTR_READER_REF__

```
#define __ATTR_READER_REF__(
              _var,
              _func ) inline auto &_func() const { return _var; }
```

Use of **ATTR_READER_REF**(var_name, func_name) will create a function named func_name which only returns a read-only reference of var_name, corresponding "attr_reader" in ruby.

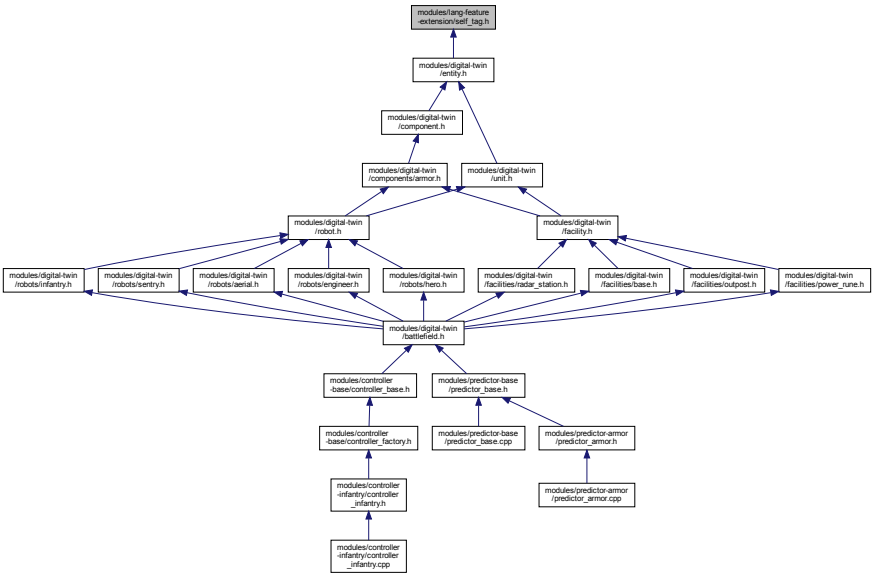
Use this macro in a class to generate a reader function for a private variable:

```
class Foo() {
public:
    Foo() { private_var_.push_back(0); }
    __ATTR_READER__(private_var_, PrivateVar)  // No need to add a semicolon.
private:
    std::vector<int> private_var_;
}
int main() {
    Foo bar;
    for (auto i: bar.PrivateVar())
        std::cout « i « std::endl;  // Allowed.  => 0
    bar.PrivateVar().push_back(0);  // Prohibited. This operation will change the value inside the object
       bar.
}
```

Definition at line 57 of file attr_reader.h.

7.44 modules/lang-feature-extension/self_tag.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define **__SELF__** ∗this

  SELF *represents for C++ object itself (simply defined as ∗this), corresponding "self" in ruby and python.*

7.44.1 Macro Definition Documentation

7.44.1.1 __SELF__

```
#define __SELF__ *this
```

SELF represents for C++ object itself (simply defined as ∗this), corresponding "self" in ruby and python.

Self tag for C++.

Author

trantuan-20048607

Date

2022.1.20

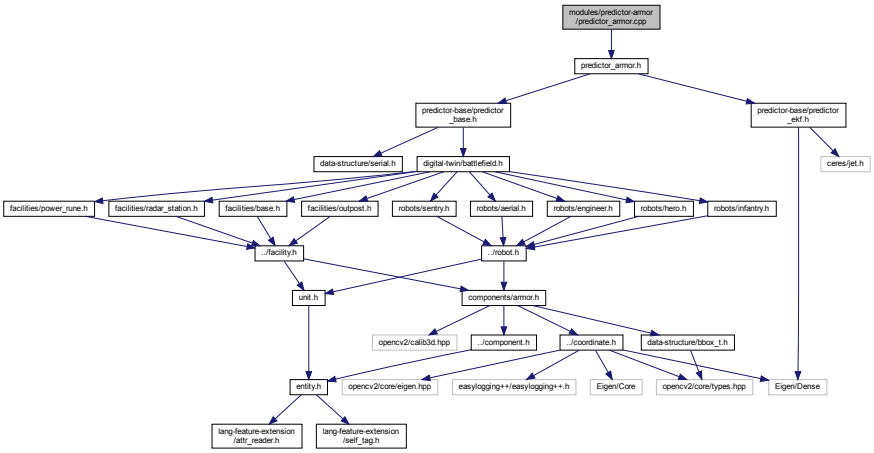
Introduced from ruby and python, self tag is used to replace ugly ∗this to refer to object itself.

Definition at line 16 of file self_tag.h.

7.45 modules/predictor-armor/predictor_armor.cpp File Reference

```
#include "predictor_armor.h"
```

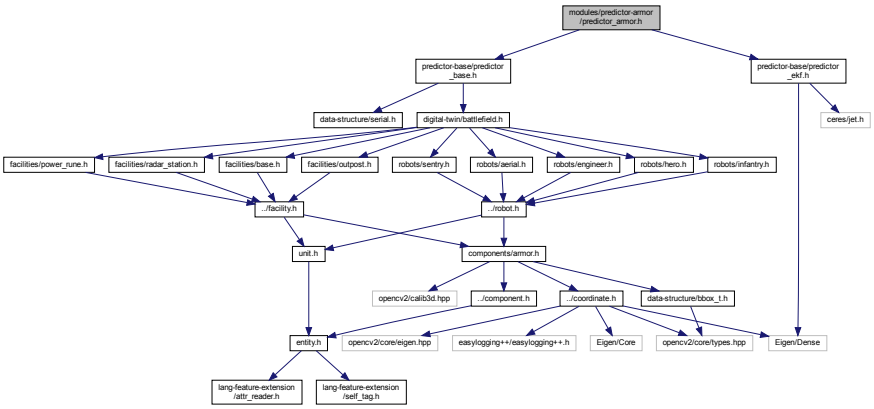
Include dependency graph for predictor_armor.cpp:



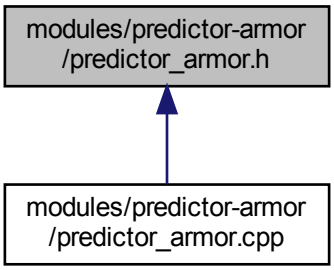
7.46 modules/predictor-armor/predictor_armor.h File Reference

```
#include "predictor-base/predictor_base.h"
#include "predictor-base/predictor_ekf.h"
```

Include dependency graph for predictor_armor.h:



This graph shows which files directly or indirectly include this file:



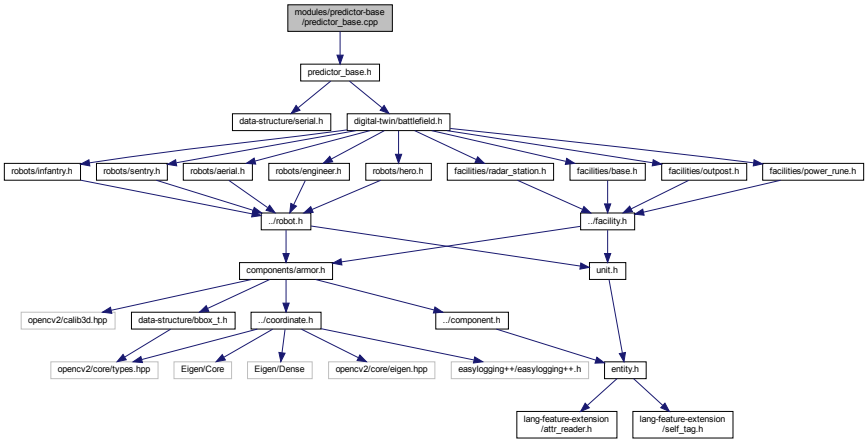
Classes

- class **ArmorPredictor**
- struct **ArmorPredictor::ArmorPredictorNode**

7.47 modules/predictor-base/predictor_base.cpp File Reference

```
#include "predictor_base.h"
```

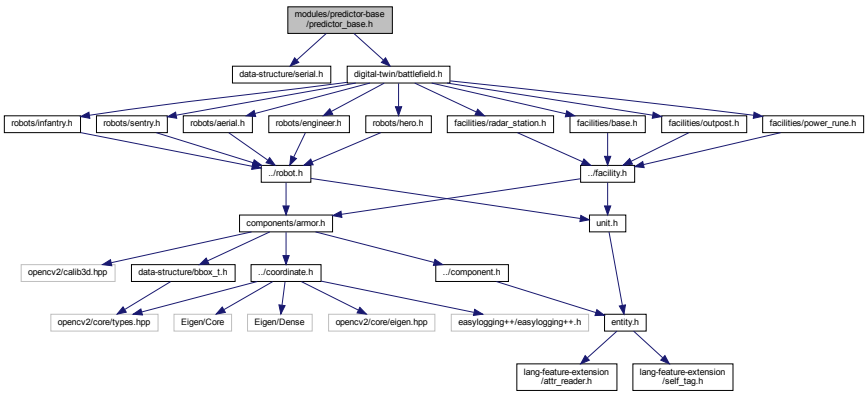
Include dependency graph for predictor_base.cpp:



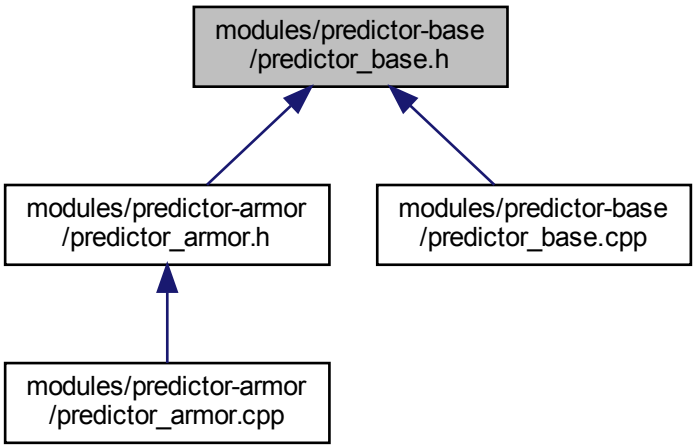
7.48 modules/predictor-base/predictor_base.h File Reference

```
#include "data-structure/serial.h"
#include "digital-twin/battlefield.h"
```

Include dependency graph for predictor_base.h:



This graph shows which files directly or indirectly include this file:



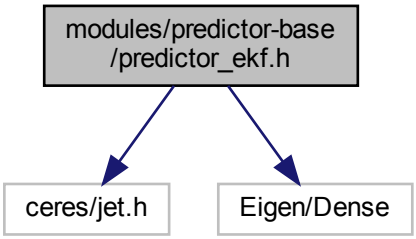
Classes

- class **Predictor**

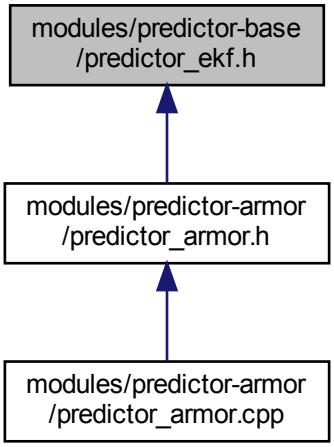
7.49 modules/predictor-base/predictor_ekf.h File Reference

```
#include <ceres/jet.h>
#include <Eigen/Dense>
```

Include dependency graph for predictor_ekf.h:



This graph shows which files directly or indirectly include this file:



Classes

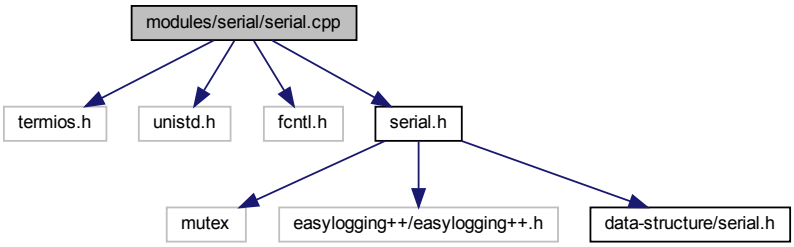
- class **ExtendedKalmanFilter**< **DataType, Nx, Ny** >

7.50 modules/serial/serial.cpp File Reference

```
#include <termios.h>
#include <unistd.h>
#include <fcntl.h>
```

```
#include "serial.h"
```

Include dependency graph for serial.cpp:



Functions

- std::string **GetUARTDeviceName** ()

  Automatically acquire UART device connected to the system.
- unsigned int **ConvertBaudRate** (unsigned int baud_rate)

  Convert uint type baud-rate to termios type.

7.50.1 Function Documentation

7.50.1.1 ConvertBaudRate()

```
unsigned int ConvertBaudRate (
            unsigned int baud_rate )  [inline]
```

Convert uint type baud-rate to termios type.

Definition at line 20 of file serial.cpp.

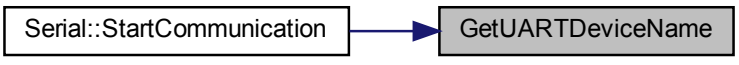
7.50.1.2 GetUARTDeviceName()

```
std::string GetUARTDeviceName ( )  [inline]
```

Automatically acquire UART device connected to the system.

Definition at line 8 of file serial.cpp.

Here is the caller graph for this function:



Index