Julio Vegans and Sons Greenhouse Monitoring System

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

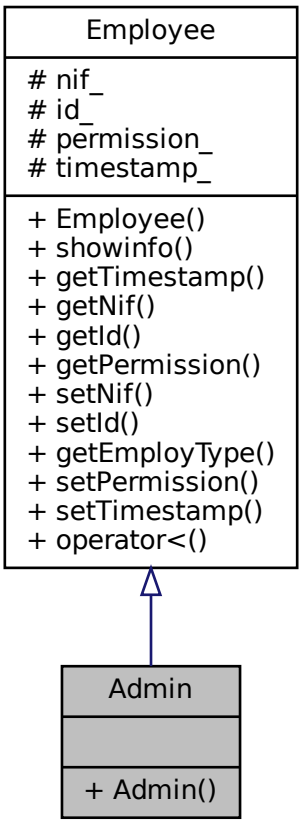
Class Documentation

4.1 Admin Class Reference

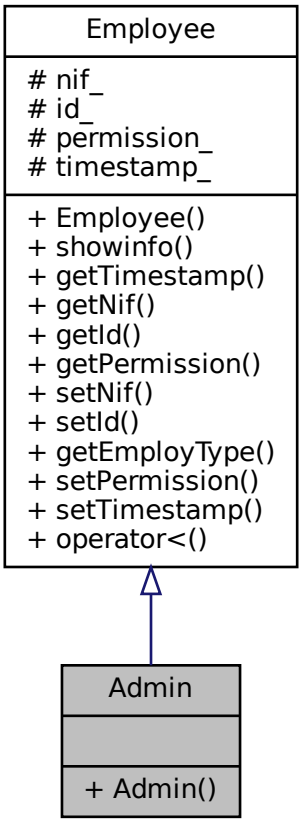
Represents an Administrator, which is a type of Employee.

`#include <Admin.h>`

Inheritance diagram for Admin:



Collaboration diagram for Admin:



Public Member Functions

- Admin (int nif=0, int id=0)

  Constructor for the Admin class.

Additional Inherited Members

4.1.1 Detailed Description

Represents an Administrator, which is a type of Employee.

This class represents an Administrator, which is a type of Employee with elevated permissions. He can manage everything.

Definition at line 15 of file Admin.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Admin()

```
Admin::Admin (
            int nif = 0,
            int id = 0 )  [inline]
```

Constructor for the Admin class.

Parameters

nif	National Identification Number of the administrator.
id	Unique identifier of the administrator.

< Set permission level to 3 for administrators.

Definition at line 22 of file Admin.h.

```
22                                  :Employee(nif, id) {
23     this->permission_ = 3;
24   };
```

References Employee::permission_.

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

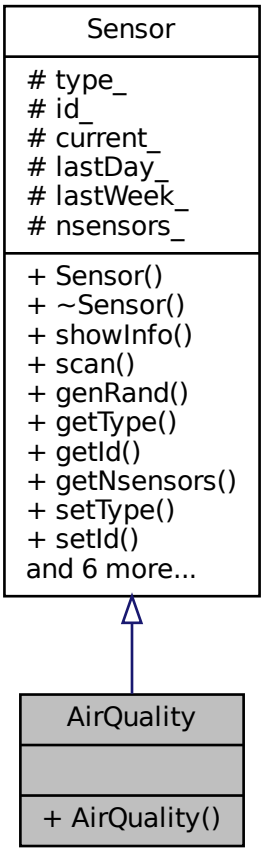
- include/Admin.h

4.2 AirQuality Class Reference

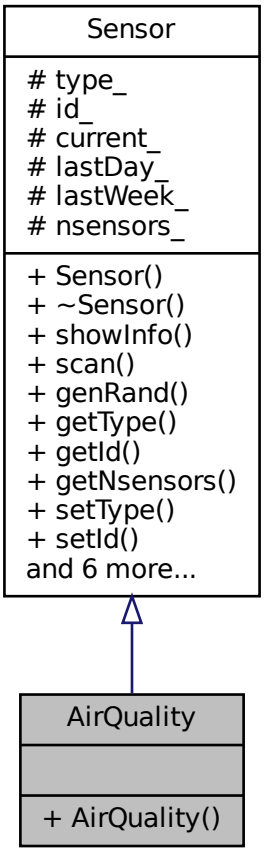
Represents a Sensor for measuring air quality.

```
#include <AirQuality.h>
```

Inheritance diagram for AirQuality:



Collaboration diagram for AirQuality:



Public Member Functions

- AirQuality (int id=0, int type=0)

  Constructor for the AirQuality class.

Additional Inherited Members

4.2.1 Detailed Description

Represents a Sensor for measuring air quality.

This class represents a sensor specialized in measuring air quality. It inherits from the Sensor class.

Definition at line 17 of file AirQuality.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 AirQuality()

```
AirQuality::AirQuality (
            int id = 0,
            int type = 0 )  [inline]
```

Constructor for the AirQuality class.

Parameters

id	Unique identifier of the air quality sensor.
type	Type of the air quality sensor.

< Set the sensor type to TYPE_AIR.

Definition at line 24 of file AirQuality.h.

```
24                                      :Sensor(id, type) {
25     this->type_ = TYPE_AIR;
26   };
```

References Sensor::type_, and TYPE_AIR.

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

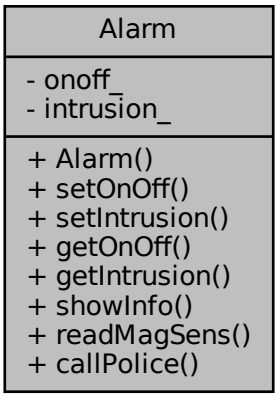
- include/AirQuality.h

4.3 Alarm Class Reference

Represents an Alarm system.

`#include <Alarm.h>`

Collaboration diagram for Alarm:



Public Member Functions

- Alarm (bool onoff=false, bool intrusion=false)

  Constructor for the Alarm class.
- void setOnOff (bool onoff)
- void setIntrusion (bool intrusion)
- bool getOnOff ()
- bool getIntrusion ()
- void showInfo (SensorDB *sensDB)

  Displays information about the alarm like if it is turned on or off, and if an intrusion has been detected.
- bool readMagSens (SensorDB *sensDB)

  Reads magnetic sensors to detect intrusions.
- void callPolice (SensorDB *sensDB)

  Calls the police if an intrusion is detected.

Private Attributes

- bool onoff_
- bool intrusion_

4.3.1 Detailed Description

Represents an Alarm system.

This class represents an alarm system with functionalities such as turning on/off the alarm, detecting intrusions, reading magnetic sensors, and calling the police if necessary.

Definition at line 18 of file Alarm.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Alarm()

```
Alarm::Alarm (
            bool onoff = false,
            bool intrusion = false )
```

Constructor for the Alarm class.

Parameters

onoff	Initial state of the alarm (true for on, false for off).
intrusion	Initial state of the intrusion detection (true if intrusion detected, false otherwise).

Definition at line 3 of file Alarm.cpp.

```
3                                       {
4     this->onoff_ = onoff;
5     this->intrusion_ = intrusion;
6 }
```

References intrusion_, and onoff_.

4.3.3 Member Function Documentation

4.3.3.1 callPolice()

```
void Alarm::callPolice (
             SensorDB * sensDB )
```

Calls the police if an intrusion is detected.

Parameters

sensDB	Pointer to the SensorDB object containing information about sensors.

Definition at line 51 of file Alarm.cpp.

```
51                                      {
52   this->intrusion_ = readMagSens(sensDB);
53   if(this->intrusion_){
54     std::cout « "Police called!" « std::endl;
55   }else{
56     std::cout « "Police not called!" « std::endl;
57   }
58 }
```

References intrusion_, and readMagSens().

Referenced by showInfo().

Here is the call graph for this function:

Here is the caller graph for this function:

4.3.3.2 getIntrusion()

```
bool Alarm::getIntrusion ( )
```

Definition at line 19 of file Alarm.cpp.

```
19                          {
20     return this->intrusion_;
21 }
```

References intrusion_.

4.3.3.3 getOnOff()

```
bool Alarm::getOnOff ( )
```

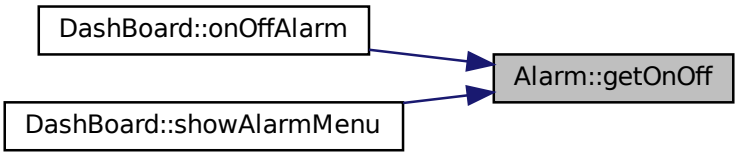
Definition at line 24 of file Alarm.cpp.

```
24                          {
25     return this->onoff_;
26 }
```

References onoff_.

Referenced by DashBoard::onOffAlarm(), and DashBoard::showAlarmMenu().

Here is the caller graph for this function:



4.3.3.4 readMagSens()

```
bool Alarm::readMagSens (
            SensorDB * sensDB )
```

Reads magnetic sensors to detect intrusions.

Parameters

sensDB	Pointer to the SensorDB object containing information about sensors.

Returns

True if an intrusion is detected, false otherwise.

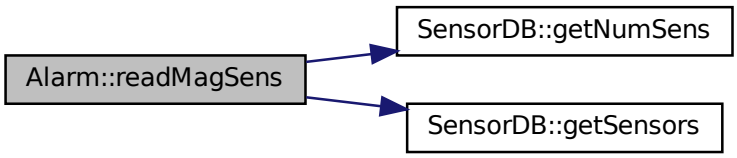
Definition at line 36 of file Alarm.cpp.

```
36                                  {
37   int i, nsens;
38
39   nsens = sensDB->getNumSens();
40   for(i = 0; i < nsens; i++){
41     if(sensDB->getSensors()[i]->getType() == TYPE_MAG){
42       if(sensDB->getSensors()[i]->getCurrent()[0] < 1.0){
43         this->intrusion_ = true;
44       }
45     }
46   }
47   return this->intrusion_;
48 }
```

References SensorDB::getNumSens(), SensorDB::getSensors(), intrusion_, and TYPE_MAG.

Referenced by callPolice().

Here is the call graph for this function:



Here is the caller graph for this function:

4.3.3.5 setIntrusion()

```
void Alarm::setIntrusion (
            bool intrusion )
```

Definition at line 9 of file Alarm.cpp.

```
9                                  {
10     this->intrusion_ = intrusion;
11 }
```

References intrusion_.

4.3.3.6 setOnOff()

```
void Alarm::setOnOff (
            bool onoff )
```

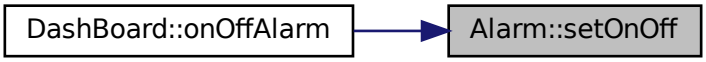
Definition at line 14 of file Alarm.cpp.

```
14                          {
15     this->onoff_ = onoff;
16 }
```

References onoff_.

Referenced by DashBoard::onOffAlarm().

Here is the caller graph for this function:



4.3.3.7 showInfo()

```
void Alarm::showInfo (
            SensorDB * sensDB )
```

Displays information about the alarm like if it is turned on or off, and if an intrusion has been detected.

Parameters

sensDB	Pointer to the SensorDB object containing information about sensors(it will check the magnetic sensors).

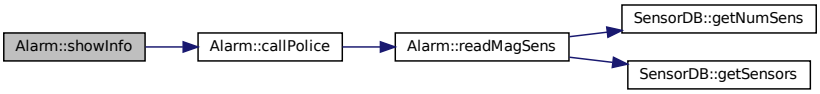
Definition at line 29 of file Alarm.cpp.

```
29                                  {
30   Alarm::callPolice(sensDB);
31   std::cout « "Alarm status: " « (this->onoff_? "ON" : "OFF") « std::endl;
32   std::cout « "Intrusion hapened? " « (this->intrusion_? "YES" : "NO") « std::endl;
33 }
```

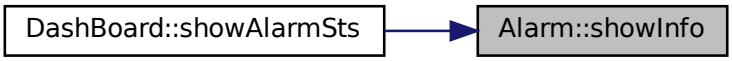
References callPolice(), intrusion_, and onoff_.

Referenced by DashBoard::showAlarmSts().

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.4 Member Data Documentation

4.3.4.1 intrusion_

```
bool Alarm::intrusion_  [private]
```

Indicates whether an intrusion has been detected.

Definition at line 21 of file Alarm.h.

Referenced by Alarm(), callPolice(), getIntrusion(), readMagSens(), setIntrusion(), and showInfo().

4.3.4.2 onoff_

```
bool Alarm::onoff_  [private]
```

Indicates whether the alarm is turned on or off.

Definition at line 20 of file Alarm.h.

Referenced by Alarm(), getOnOff(), setOnOff(), and showInfo().

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

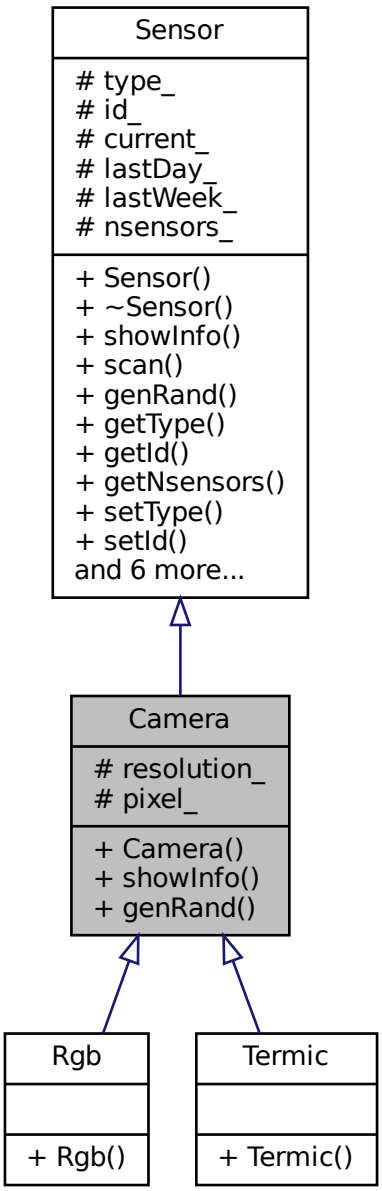
- include/Alarm.h
- src/Alarm.cpp

4.4 Camera Class Reference

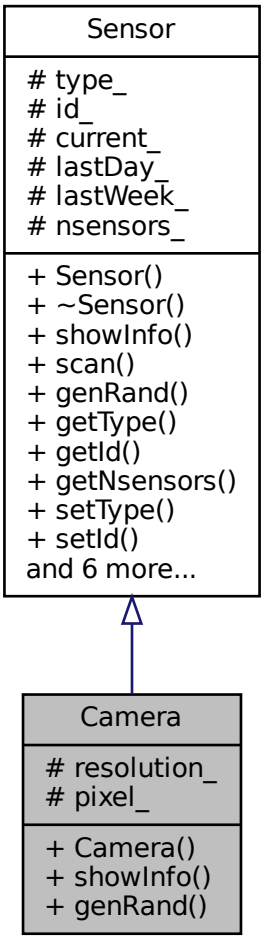
Represents a camera Sensor.

`#include <Camera.h>`

Inheritance diagram for Camera:



Collaboration diagram for Camera:



Public Member Functions

- Camera (int id=0, int type=0, std::tuple< uint, uint > resolution=std::make_tuple(10, 10))

  Constructor for the Camera class.
- void showInfo () override

  Displays information about the camera sensor.
- uint genRand ()

  Generates a random unsigned integer.

Protected Attributes

- std::tuple< uint, uint > resolution_
- std::tuple< uint, uint, uint > pixel_

Additional Inherited Members

4.4.1 Detailed Description

Represents a camera Sensor.

This class represents a camera sensor, which is a type of sensor with resolution and pixel information.

Definition at line 20 of file Camera.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Camera()

```
Camera::Camera (
            int id = 0,
            int type = 0,
            std::tuple< uint, uint > resolution = std::make_tuple(10, 10) )  [explicit]
```

Constructor for the Camera class.

Parameters

id	Unique identifier of the camera sensor.
type	Type of the camera sensor.
resolution	Resolution of the camera sensor (width x height).

Definition at line 3 of file Camera.cpp.

```
3                                                                   {
4   uint totres;
5   uint C1, C2, C3;
6   uint i;
7
8   this->id_ = id;
9   this->type_ = TYPE_CAM;
10   this->resolution_ = resolution;
11   totres = std::get<0>(resolution_) * std::get<1>(resolution_);
12   delete [] this->current_;
13   this->current_ = new double[totres];
14   for(i = 0; i < totres; i+=3){
15     C1 = Camera::genRand();
16     C2 = Camera::genRand();
17     C3 = Camera::genRand();
18     this->pixel_ = std::make_tuple(C1, C2, C3);
19     this->current_[i] = static_cast<double>(std::get<0>(this->pixel_));
20     if (i+1 < totres) this->current_[i+1] = static_cast<double>(std::get<1>(this->pixel_));
21     if (i+2 < totres) this->current_[i+2] = static_cast<double>(std::get<2>(this->pixel_));
22   }
23 }
```

References Sensor::current_, genRand(), Sensor::id_, pixel_, resolution_, Sensor::type_, and TYPE_CAM.

Here is the call graph for this function:

4.4.3 Member Function Documentation

4.4.3.1 genRand()

```
uint Camera::genRand ( )
```

Generates a random unsigned integer.

Returns

Random unsigned integer.

Definition at line 49 of file Camera.cpp.

```
49                 {
50     auto time_micros = std::chrono::duration_cast<std::chrono::microseconds>(
51     std::chrono::system_clock::now().time_since_epoch()).count(); // getting time in microseconds
52
53     srand(time_micros); // using current time in microseconds as seed for random generator
54     uint randNum = rand() % 256;
55     return randNum;
56 }
```

Referenced by Camera().

Here is the caller graph for this function:

4.4.3.2 showInfo()

```
void Camera::showInfo ( )  [override], [virtual]
```

Displays information about the camera sensor.

Reimplemented from Sensor.

Definition at line 26 of file Camera.cpp.

```
26                {
27   uint i, totres;
28
29   totres = std::get<0>(resolution_) * std::get<1>(resolution_);
30   switch(this->type_){
31   case TYPE_RGB:
32     std::cout « "Type: " « this->type_ « ". RGB Camera.\n";
33     break;
34   case TYPE_TERMIC:
35     std::cout « "Type: " « this->type_ « ". Termic Camera.\n";
36     break;
37   case TYPE_CAM:
38     std::cout « "Type: " « this->type_ « ". Camera.\n";
39   }
40   std::cout « "Id: " « this->id_ « std::endl;
41   std::cout « "Current:\n[";
42   for(i = 0; i < totres; i++){
43     i != totres -1? std::cout « this->current_[i] « ", ": std::cout « this->current_[i] « "]\n";
44   }
45
46 }
```

References Sensor::current_, Sensor::id_, resolution_, Sensor::type_, TYPE_CAM, TYPE_RGB, and TYPE_↩ TERMIC.

4.4.4 Member Data Documentation

4.4.4.1 pixel_

```
std::tuple<uint, uint, uint> Camera::pixel_  [protected]
```

Pixel information (RGB) of the camera sensor.

Definition at line 43 of file Camera.h.

Referenced by Camera().

4.4.4.2 resolution_

```
std::tuple<uint, uint> Camera::resolution_  [protected]
```

Resolution of the camera sensor (width x height).

Definition at line 42 of file Camera.h.

Referenced by Camera(), and showInfo().

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

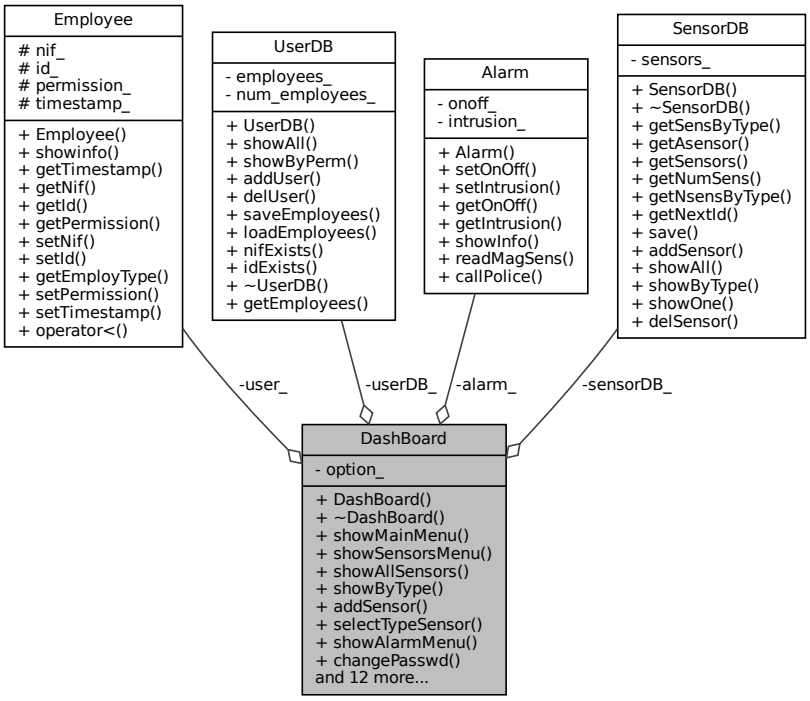
- include/Camera.h
- src/Camera.cpp

4.5 DashBoard Class Reference

Represents a dashboard for system management.

`#include <DashBoard.h>`

Collaboration diagram for DashBoard:



Public Member Functions

- DashBoard (Employee *user=nullptr, SensorDB *sensorDB=nullptr, Alarm *alarm=nullptr, UserDB *user↩ DB=nullptr)

  Constructor for the DashBoard class.
- ~DashBoard ()

  Destructor for the DashBoard class.
- void showMainMenu ()

  Displays the main menu.
- void showSensorsMenu ()

  Displays the sensors menu.
- void showAllSensors (SensorDB *sensorDB)

  Displays information about all sensors.
- void showByType (SensorDB *sensorDB)

  Displays information about sensors of a specific type.
- void addSensor (SensorDB *sensorDB)

  Adds a new sensor to the database.
- int selectTypeSensor ()

Prompts the user to select a sensor type.

- void showAlarmMenu (Alarm alarm)

  Displays the alarm menu.
- void changePasswd (UserDB *userDB, Employee *user)

  Changes the password of a user.
- void showByPerm (UserDB *userDB)

  Displays information about users by permission level.
- void delUser (UserDB *userDB)

  Deletes a user from the database.
- void showAllUsers (UserDB *userDB)

  Displays information about all users.
- void addUser (UserDB *userDB)

  Adds a new user to the database.
- void showUsersMenu ()

  Displays the users menu.
- int askForOption (int max)

  Prompts the user to select an option.
- void delSensor (SensorDB *sensorDB)

  Deletes a sensor from the database.
- void showAlarmSts (Alarm *alarm, SensorDB *sensorDB)

  Displays the status of the alarm.
- void onOffAlarm (Alarm *alarm)

  Turns the alarm on or off.
- int getOption ()
- Employee * getUser ()
- void setUser (Employee *user)

Private Attributes

- Employee * user_
- int option_
- SensorDB * sensorDB_
- Alarm * alarm_
- UserDB * userDB_

4.5.1 Detailed Description

Represents a dashboard for system management.

This class represents a dashboard for system management, with functionalities for displaying menus and managing sensors.

Definition at line 23 of file DashBoard.h.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 DashBoard()

```
DashBoard::DashBoard (
            Employee * user = nullptr,
            SensorDB * sensorDB = nullptr,
            Alarm * alarm = nullptr,
            UserDB * userDB = nullptr )
```

Constructor for the DashBoard class.

Parameters

user	Pointer to the logged-in user.
sensorDB	Pointer to the SensorDB object.
alarm	Pointer to the Alarm object.
userDB	Pointer to the UserDB object.

Definition at line 3 of file DashBoard.cpp.

```
3                                                                                {
4   this->user_ = user;
5   this->option_ = 0;
6   this->sensorDB_ = sensorDB;
7   this->alarm_ = alarm;
8   this->userDB_ = userDB;
9 }
```

References alarm_, option_, sensorDB_, user_, and userDB_.

4.5.2.2 ~DashBoard()

```
DashBoard::~DashBoard ( )
```

Destructor for the DashBoard class.

Definition at line 11 of file DashBoard.cpp.

```
11                          {
12   delete this->sensorDB_;
13 }
```

References sensorDB_.

4.5.3 Member Function Documentation

4.5.3.1 addSensor()

```
void DashBoard::addSensor (
            SensorDB * sensorDB )
```

Adds a new sensor to the database.

Parameters

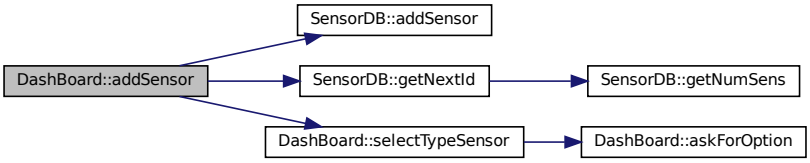
sensorDB	Pointer to the SensorDB object.

Definition at line 324 of file DashBoard.cpp.

```
324                                     {
325   int type, id;
326
327   std::cout « "Select type to add:\n";
328   type = DashBoard::selectTypeSensor();
329   id = sensordb->getNextId();
330   sensordb->addSensor(type, id);
331 }
```

References SensorDB::addSensor(), SensorDB::getNextId(), and selectTypeSensor().

Here is the call graph for this function:



4.5.3.2 addUser()

```
void DashBoard::addUser (
            UserDB * userDB )
```

Adds a new user to the database.

Parameters

userDB	Pointer to the UserDB object.

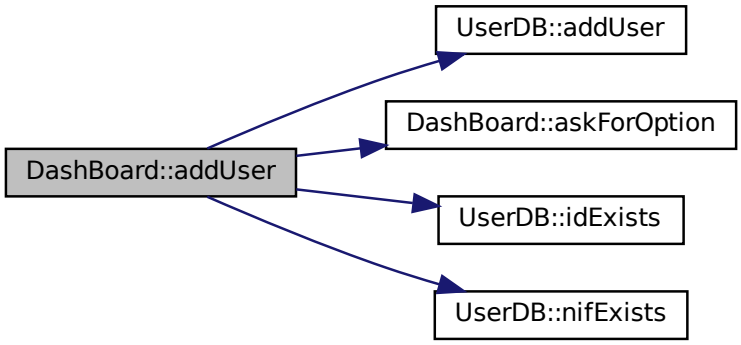
Definition at line 203 of file DashBoard.cpp.

```
203                                     {
204   int nif;
205   int password;
206   int permission;
207   bool valid = false;
208   Employee *newuser;
209
210   while(!valid){
211     std::cout « "Insert NIF: ";
212     try{
213       nif = DashBoard::askForOption(99999999);
214       if(userDB->nifExists(nif)){
215         std::cout « "NIF already exists, try again: ";
216       }else{
217         valid = true;
218       }
219     }catch(ValueError &e){
220       std::cout « "\nInvalid number, try again\n\n";
221     }
```

```
222   }
223   valid = false;
224   while(!valid){
225     std::cout « "Insert password: ";
226     try{
227       password = DashBoard::askForOption(99999);
228       if(userDB->idExists(password)){
229         std::cout « "Invalid password, try again: ";
230       }else{
231         valid = true;
232       }
233     }catch(ValueError &e){
234       std::cout « "\nInvalid number, try again\n\n";
235     }
236   }
237   valid = false;
238   while(!valid){
239     std::cout « "Insert permission(1 for worker, 2 for supervisor, 3 for admin): ";
240     try{
241       permission = DashBoard::askForOption(3);
242       valid = true;
243     }catch(ValueError &e){
244       std::cout « "\nInvalid number, try again\n\n";
245     }
246   }
247   newuser = new Employee(nif, password, permission);
248   userDB->addUser(newuser);
249 }
```

References UserDB::addUser(), askForOption(), UserDB::idExists(), and UserDB::nifExists().

Here is the call graph for this function:



4.5.3.3 askForOption()

```
int DashBoard::askForOption (
            int max )
```

Prompts the user to select an option.

Parameters

max	Maximum allowed option (between 1 and max).

Returns

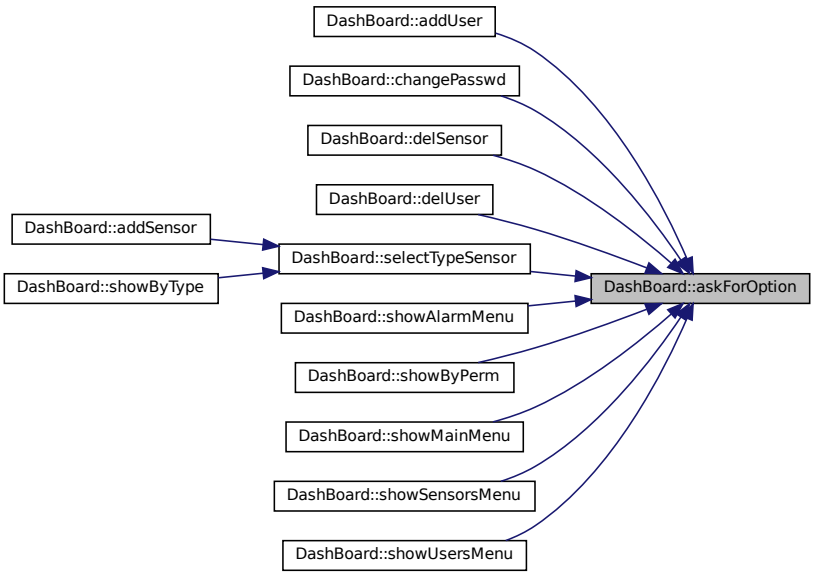
Selected option.

Definition at line 29 of file DashBoard.cpp.

```
29                                {
30   int option;
31
32   std::cin » option;
33   if (option < 1 || option > max){
34     if (std::cin.fail()) {
35       // Limpiamos el estado de error de std::cin
36       std::cin.clear();
37       // Descartamos cualquier entrada incorrecta en el búfer
38       std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
39     }
40     throw ValueError();
41   }
42   std::cout « std::endl;
43   std::cout « "Selected option: " « option « std::endl « std::endl;
44   return option;
45 }
```

Referenced by addUser(), changePasswd(), delSensor(), delUser(), selectTypeSensor(), showAlarmMenu(), showByPerm(), showMainMenu(), showSensorsMenu(), and showUsersMenu().

Here is the caller graph for this function:



4.5.3.4 changePasswd()

```
void DashBoard::changePasswd (
            UserDB * userDB,
            Employee * user )
```

Changes the password of a user.

Parameters

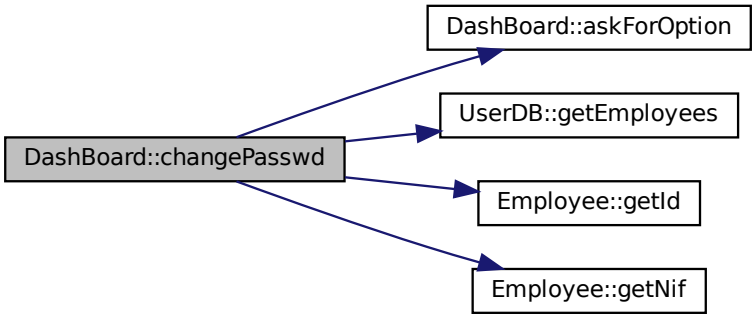
userDB	Pointer to the UserDB object.
user	Pointer to the Employee object representing the user.

Definition at line 114 of file DashBoard.cpp.

```
114                                                    {
115   bool valid = false;
116   int oldpasswd, newpasswd;
117
118   while(!valid){
119     std::cout « "Enter your current password: ";
120     try{
121       oldpasswd = DashBoard::askForOption(99999);
122     }catch(ValueError &e){
123       std::cerr « "\nInvalid number, try again\n\n";
124       continue;
125     }
126     if(user->getId() == oldpasswd){
127       valid = true;
128     }else{
129       std::cerr « "\nInvalid number, try again\n\n";
130     }
131   }
132   valid = false;
133   while(!valid){
134     std::cout « "Enter your new password: ";
135     try{
136       newpasswd = DashBoard::askForOption(99999);
137       valid = true;
138     }catch(ValueError &e){
139       std::cerr « "\nInvalid number, try again\n\n";
140     }
141   }
142   for(auto const& employee: userDB->getEmployees()){
143     if (employee->getNif() == user->getNif()){
144       employee->setId(newpasswd);
145       break;
146     }
147   }
148   std::cout « "Password changed successfully.\n" « std::endl;
149 }
```

References askForOption(), UserDB::getEmployees(), Employee::getId(), and Employee::getNif().

Here is the call graph for this function:



4.5.3.5 delSensor()

```
void DashBoard::delSensor (
            SensorDB * sensorDB )
```

Deletes a sensor from the database.

Parameters

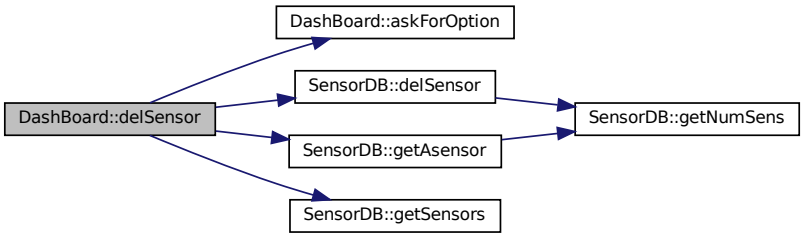
sensorDB	Pointer to the SensorDB object.

Definition at line 334 of file DashBoard.cpp.

```
334                                         {
335   int id, i, maxid = 0;
336   bool valid = false;
337
338   std::cout « "Select sensor to delete(the id):\n";
339   for(i = 0; i < (int)sensorDB->getSensors().size(); i++){
340     if(maxid < sensorDB->getSensors()[i]->getId()){
341       maxid = sensorDB->getSensors()[i]->getId();
342     }
343     std::cout « "Id: " « sensorDB->getSensors()[i]->getId() « " - ";
344     switch(sensorDB->getSensors()[i]->getType()){
345     case(TYPE_TEMP):
346       std::cout « "Temperature.\n";
347       break;
348     case(TYPE_HUM):
349       std::cout « "Humidity.\n";
350       break;
351     case(TYPE_MAG):
352       std::cout « "Magnetic.\n";
353       break;
354     case(TYPE_LIGHT):
355       std::cout « "Light.\n";
356       break;
357     case(TYPE_AIR):
358       std::cout « "Air quality.\n";
359       break;
360     case(TYPE_TERMIC):
361       std::cout « "Termic Camera.\n";
362       break;
363     case(TYPE_RGB):
364       std::cout « "RGB Camera.\n";
365       break;
366     case(TYPE_CAM):
367       std::cout « "Camera.\n";
368       break;
369     }
370   }
371   while(!valid){
372     std::cout « "Your option: ";
373     try{
374       id = DashBoard::askForOption(maxid);
375       if(sensorDB->getAsensor(id) == nullptr){
376         throw ValueError();
377       }
378       valid = true;
379     }catch(ValueError &e){
380       std::cerr « "\nInvalid number, try again(1 - " « maxid « ")\n\n";
381     }
382   }
383   sensorDB->delSensor(id);
384   std::cout « "\nSensor deleted.\n" « std::endl;
385 }
```

References askForOption(), SensorDB::delSensor(), SensorDB::getAsensor(), SensorDB::getSensors(), TYPE_↩ AIR, TYPE_CAM, TYPE_HUM, TYPE_LIGHT, TYPE_MAG, TYPE_RGB, TYPE_TEMP, and TYPE_TERMIC.

Here is the call graph for this function:



4.5.3.6 delUser()

```
void DashBoard::delUser (
            UserDB * userDB )
```

Deletes a user from the database.

Parameters

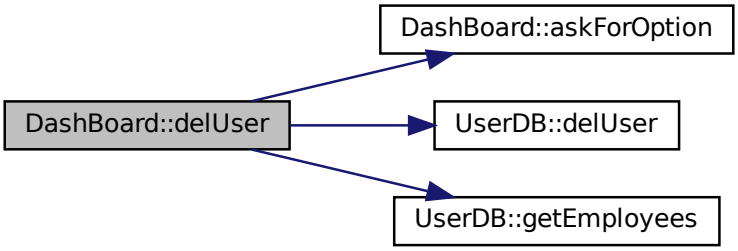
userDB	Pointer to the UserDB object.

Definition at line 48 of file DashBoard.cpp.

```
48                                      {
49   bool valid = false;
50   int i, nif;
51
52   while(!valid){
53     i = 1;
54     std::cout « "NIFs to delete:\n";
55     for (auto const& employee: userDB->getEmployees()) {
56       std::cout « i « ". " « employee->getNif() « std::endl;
57       i++;
58     }
59     try{
60       std::cout « "Select the NIF of the user to delete: ";
61       nif = DashBoard::askForOption(i - 1);
62       valid = true;
63     }catch(ValueError &e){
64       std::cerr « "\nInvalid number, try again(1 - " « i - 1 « ")\n\n";
65     }
66   }
67   userDB->delUser(nif - 1);
68 }
```

References askForOption(), UserDB::delUser(), and UserDB::getEmployees().

Here is the call graph for this function:



4.5.3.7 getOption()

```
int DashBoard::getOption ( )
```

Definition at line 402 of file DashBoard.cpp.

```
402                         {
403   return this->option_;
404 }
```

References option_.

4.5.3.8 getUser()

```
Employee * DashBoard::getUser ( )
```

Definition at line 407 of file DashBoard.cpp.

```
407                        {
408   return this->user_;
409 }
```

References user_.

4.5.3.9 onOffAlarm()

```
void DashBoard::onOffAlarm (
            Alarm * alarm )
```

Turns the alarm on or off.

Parameters

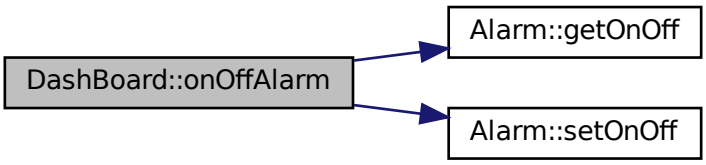
alarm	Pointer to the Alarm object.

Definition at line 252 of file DashBoard.cpp.

```
252                                 {
253   alarm->setOnOff(!alarm->getOnOff());
254   std::cout « "Alarm turned " « (alarm->getOnOff()? "on.\n" : "off.\n") « std::endl;
255 }
```

References Alarm::getOnOff(), and Alarm::setOnOff().

Here is the call graph for this function:



4.5.3.10 selectTypeSensor()

```
int DashBoard::selectTypeSensor ( )
```

Prompts the user to select a sensor type.

Returns

Selected sensor type.

Definition at line 300 of file DashBoard.cpp.

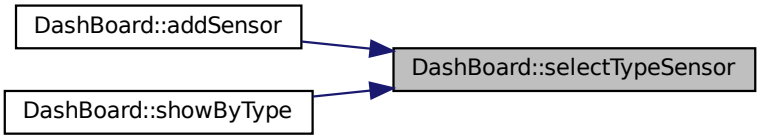
```
300                             {
301   bool numvalid = false;
302   int type;
303 
304   while(!numvalid){
305     std::cout « "Types:\n";
306     std::cout « "1. Temperature.\n";
307     std::cout « "2. Humidity.\n";
308     std::cout « "3. Light quality.\n";
309     std::cout « "4. Air Quality.\n";
310     std::cout « "5. Camera RGB.\n";
311     std::cout « "6. Camera Termic.\n";
312     std::cout « "Your option: ";
313     try{
314       type = DashBoard::askForOption(6);
315       numvalid = true;
316     }catch (ValueError &e){
317       std::cerr « "\nInvalid number, try again(1 - 6)\n\n";
318     }
319   }
320   return type;
321 }
```

References askForOption().

Referenced by addSensor(), and showByType().

Here is the call graph for this function:

Here is the caller graph for this function:



4.5.3.11 setUser()

```
void DashBoard::setUser (
            Employee * user )
```

Definition at line 24 of file DashBoard.cpp.

```
24                                  {
25   this->user_ = user;
26 }
```

References user_.

4.5.3.12 showAlarmMenu()

```
void DashBoard::showAlarmMenu (
            Alarm alarm )
```

Displays the alarm menu.

Parameters

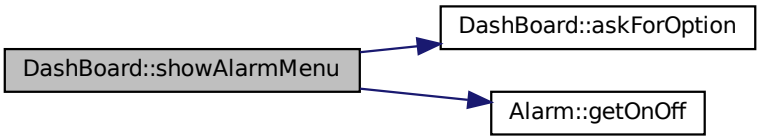
alarm	Reference to the Alarm object.

Definition at line 263 of file DashBoard.cpp.

```
263                                     {
264   bool valid = false;
265
266   while(!valid){
267     std::cout « "Manage Alarm." « std::endl;
268     if(alarm.getOnOff()){
269     std::cout « "1. Turn off Alarm." « std::endl;
270     }else{
271         std::cout « "1. Turn on Alarm." « std::endl;
272     }
273     std::cout « "2. Show status." « std::endl;
274     std::cout « "3. Go back." « std::endl;
275     std::cout « "Your option: ";
276     try{
277       this->option_ = DashBoard::askForOption(3);
278       valid = true;
279     }catch(ValueError &e){
280       std::cout « "\nInvalid number, try again(1 - 2)\n\n";
281     }
282   }
283 }
```

References askForOption(), Alarm::getOnOff(), and option_.

Here is the call graph for this function:



4.5.3.13 showAlarmSts()

```
void DashBoard::showAlarmSts (
            Alarm * alarm,
            SensorDB * sensorDB )
```

Displays the status of the alarm.

Parameters

alarm	Pointer to the Alarm object.
sensorDB	Pointer to the SensorDB object.

Definition at line 258 of file DashBoard.cpp.

```
258                                                 {
```

```
259   alarm->showInfo(sensorDB);
260 }
```

References Alarm::showInfo().

Here is the call graph for this function:

4.5.3.14 showAllSensors()

```
void DashBoard::showAllSensors (
            SensorDB * sensorDB )
```

Displays information about all sensors.

Parameters

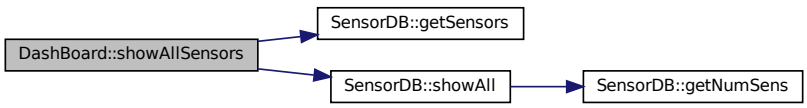
sensorDB	Pointer to the SensorDB object.

Definition at line 286 of file DashBoard.cpp.

```
286                                          {
287   int i;
288
289   for(i = 0; i < (int)sensorDB->getSensors().size(); i++){
290     if (sensorDB->getSensors()[i]->getType() < TYPE_RGB){
291       sensorDB->getSensors()[i]->scan(0.0, 100.0);
292     }
293   }
294   std::cout « "\nSensors: " « std::endl;
295   sensorDB->showAll();
296   std::cout « std::endl;
297 }
```

References SensorDB::getSensors(), SensorDB::showAll(), and TYPE_RGB.

Here is the call graph for this function:



4.5.3.15 showAllUsers()

```
void DashBoard::showAllUsers (
            UserDB * userDB )
```

Displays information about all users.

Parameters

userDB	Pointer to the UserDB object.

Definition at line 15 of file DashBoard.cpp.

```
15                                          {
16   std::cout « "All users:" « std::endl;
17   for (auto const& employee: userDB->getEmployees()) {
18     employee->showinfo();
19   }
20   std::cout « std::endl;
21 }
```

References UserDB::getEmployees().

Here is the call graph for this function:

4.5.3.16 showByPerm()

```
void DashBoard::showByPerm (
            UserDB * userDB )
```

Displays information about users by permission level.

Parameters

userDB	Pointer to the UserDB object.

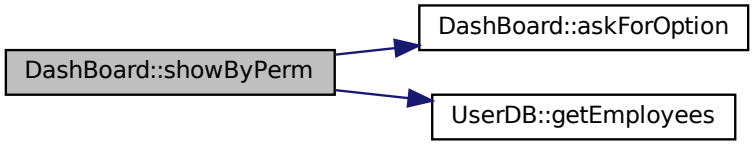
Definition at line 71 of file DashBoard.cpp.

```
71                                          {
72   bool valid = false;
73   int permission;
74
75   while(!valid){
76     std::cout « "Enter the permission level: ";
77     try{
78       permission = DashBoard::askForOption(3);
79       valid = true;
80     }catch(ValueError &e){
```

```
81        std::cerr « "\nInvalid number, try again(1 - 3)\n\n";
82      }
83    }
84    for (auto const& employee: userDB->getEmployees()) {
85      if (employee->getPermission() == permission){
86        employee->showinfo();
87      }
88    }
89    std::cout « std::endl;
90  }
```

References askForOption(), and UserDB::getEmployees().

Here is the call graph for this function:



4.5.3.17 showByType()

```
void DashBoard::showByType (
            SensorDB * sensorDB )
```

Displays information about sensors of a specific type.

Parameters

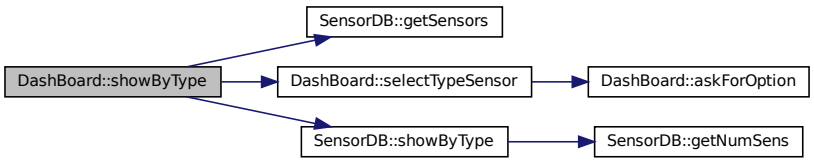
sensorDB	Pointer to the SensorDB object.

Definition at line 388 of file DashBoard.cpp.

```
388                                        {
389    int type, i;
390
391    type = DashBoard::selectTypeSensor();
392    for(i = 0; i < (int)sensorDB->getSensors().size(); i++){
393      if (sensorDB->getSensors()[i]->getType() == type){
394        sensorDB->getSensors()[i]->scan(0.0, 100.0);
395        }
396    }
397    std::cout « "\nSensors of type " « type « ": " « std::endl;
398    sensorDB->showByType(type);
399    std::cout « std::endl;
400  }
```

References SensorDB::getSensors(), selectTypeSensor(), and SensorDB::showByType().

Here is the call graph for this function:



4.5.3.18 showMainMenu()

```
void DashBoard::showMainMenu ( )
```

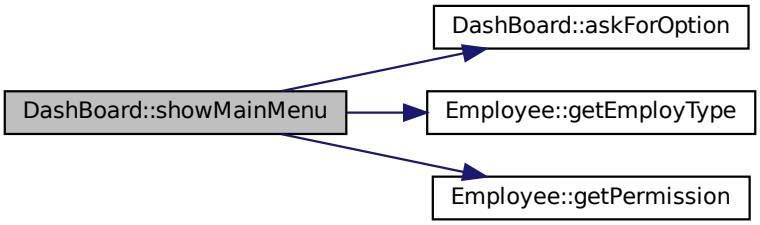
Displays the main menu.

Definition at line 152 of file DashBoard.cpp.

```
152                         {
153   bool numvalid = false;
154   int max = 4;
155
156   std::cout « "Welcome to the DashBoard. Logged as: " « user_->getEmployType() « std::endl;
157   while(!numvalid){
158     std::cout « "What would you like to do?" « std::endl;
159     std::cout « "1. Manage Sensors." « std::endl;
160     std::cout « "2. Change my password." « std::endl;
161     std::cout « "3. Log out." « std::endl;
162     std::cout « "4. Exit." « std::endl;
163     if (this->user_->getPermission() >= 2){
164       std::cout « "5. Manage Alarm." « std::endl;
165       max = 5;
166     }
167     if (this->user_->getPermission() >= 3){
168       std::cout « "6. Manage Users." « std::endl;
169       max = 6;
170     }
171     std::cout « "Your option: ";
172     try{
173       this->option_ = DashBoard::askForOption(max);
174       numvalid = true;
175     }catch(ValueError &e){
176       std::cerr « "\nInvalid number, try again(1 - " « max « ")\n\n";
177     }
178   }
179 }
```

References askForOption(), Employee::getEmployType(), Employee::getPermission(), option_, and user_.

Here is the call graph for this function:



4.5.3.19 showSensorsMenu()

```
void DashBoard::showSensorsMenu ( )
```

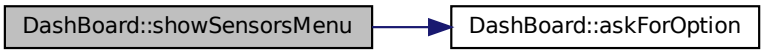
Displays the sensors menu.

Definition at line 182 of file DashBoard.cpp.

```
182                                {
183   bool numvalid = false;
184
185   while(!numvalid){
186     std::cout « "Manage Sensors." « std::endl;
187     std::cout « "1. Add a sensor." « std::endl;
188     std::cout « "2. Show all sensors." « std::endl;
189     std::cout « "3. Show sensors by type." « std::endl;
190     std::cout « "4. Delete a sensor." « std::endl;
191     std::cout « "5. Save and go back." « std::endl;
192     std::cout « "Your option: ";
193     try{
194       this->option_ = DashBoard::askForOption(5);
195       numvalid = true;
196     }catch (ValueError &e){
197       std::cout « "\nInvalid number, try again(1 - 5)\n\n";
198     }
199   }
200 }
```

References askForOption(), and option_.

Here is the call graph for this function:



4.5.3.20 showUsersMenu()

```
void DashBoard::showUsersMenu ( )
```

Displays the users menu.

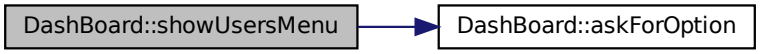
Definition at line 93 of file DashBoard.cpp.

```
93                              {
94   bool numvalid = false;
95   int max = 5;
96   std::cout « "What would you like to do?" « std::endl;
97   while(!numvalid){
98     std::cout « "1. Add user." « std::endl;
99     std::cout « "2. Show all users." « std::endl;
100     std::cout « "3. Show users by permission." « std::endl;
101     std::cout « "4. Delete user." « std::endl;
102     std::cout « "5. Save and Back." « std::endl;
103     std::cout « "Your option: ";
104     try{
105       this->option_ = DashBoard::askForOption(max);
106       numvalid = true;
```

```
107     }catch(ValueError &e){
108       std::cerr « "\nInvalid number, try again(1 - " « max « ")\n\n";
109     }
110   }
111 }
```

References askForOption(), and option_.

Here is the call graph for this function:



4.5.4 Member Data Documentation

4.5.4.1 alarm_

```
Alarm* DashBoard::alarm_  [private]
```

Pointer to the Alarm object.

Definition at line 28 of file DashBoard.h.

Referenced by DashBoard().

4.5.4.2 option_

```
int DashBoard::option_  [private]
```

Selected option in the menu.

Definition at line 26 of file DashBoard.h.

Referenced by DashBoard(), getOption(), showAlarmMenu(), showMainMenu(), showSensorsMenu(), and showUsersMenu().

4.5.4.3 sensorDB_

```
SensorDB* DashBoard::sensorDB_  [private]
```

Pointer to the SensorDB object.

Definition at line 27 of file DashBoard.h.

Referenced by DashBoard(), and ~DashBoard().

4.5.4.4 user_

```
Employee* DashBoard::user_  [private]
```

Pointer to the logged-in user.

Definition at line 25 of file DashBoard.h.

Referenced by DashBoard(), getUser(), setUser(), and showMainMenu().

4.5.4.5 userDB_

```
UserDB* DashBoard::userDB_  [private]
```

Pointer to the UserDB object.

Definition at line 29 of file DashBoard.h.

Referenced by DashBoard().

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

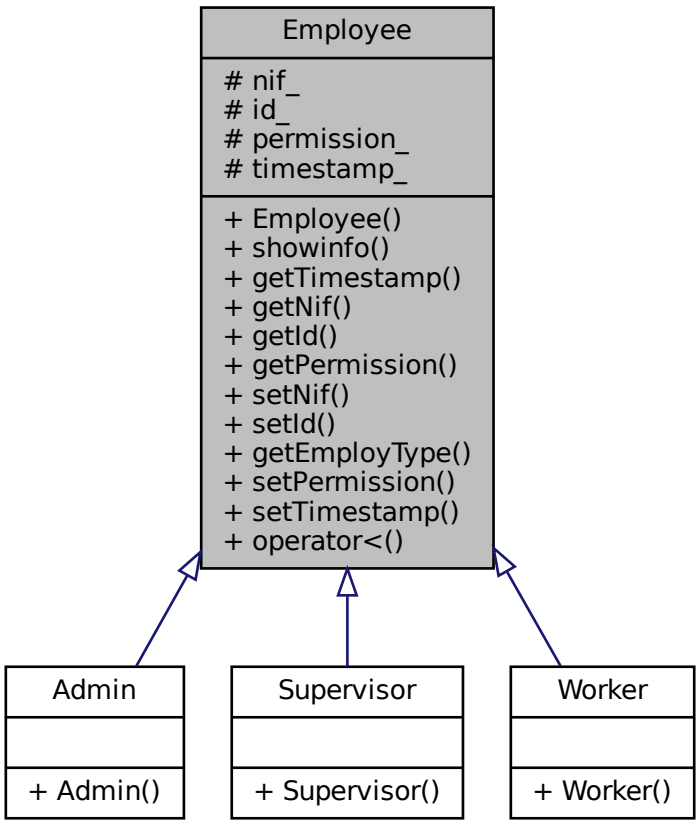
- include/DashBoard.h
- src/DashBoard.cpp

4.6 Employee Class Reference

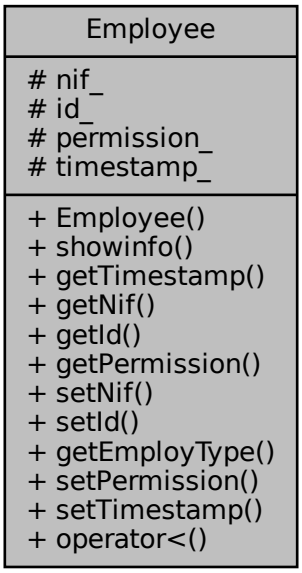
Represents an Employee.

`#include <Employee.h>`

Inheritance diagram for Employee:



Collaboration diagram for Employee:



Public Member Functions

- Employee (int nif=0, int id=0, int permission=0)

  Constructor for the Employee class.
- void showinfo ()

  Displays information about the employee.
- std::time_t getTimestamp ()
- int getNif () const
- int getId () const
- int getPermission () const
- void setNif (int nif)
- void setId (int id)
- std::string getEmployType ()
- void setPermission (int permission)
- void setTimestamp (std::time_t timestamp)
- bool operator< (const Employee &right) const

  Overloaded less-than operator to compare two employees.

Protected Attributes

- int nif_
- int id_
- int permission_
- std::time_t timestamp_

4.6.1 Detailed Description

Represents an Employee.

This class represents an Employee with attributes such as NIF, ID, permission level, and timestamp.

Definition at line 18 of file Employee.h.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 Employee()

```
Employee::Employee (
            int nif = 0,
            int id = 0,
            int permission = 0 )
```

Constructor for the Employee class.

Parameters

nif	National Identification Number of the employee.
id	Unique identifier of the employee.
permission	Permission level of the employee.

Definition at line 4 of file Employee.cpp.

```
4                                                   {
5   this->nif_ = nif;
6   this->id_ = id;
7   this->permission_ = permission;
8   this->timestamp_ = std::time(0);
9 }
```

References id_, nif_, permission_, and timestamp_.

4.6.3 Member Function Documentation

4.6.3.1 getEmployType()

```
std::string Employee::getEmployType ( )
```

Definition at line 12 of file Employee.cpp.

```
12                          {
13   if (this->permission_ == 1)
14     return "Worker";
15   else if (this->permission_ == 2)
16     return "Supervisor";
17   else
```

```
18     return "Admin";
19 }
```

References permission_.

Referenced by DashBoard::showMainMenu().

Here is the caller graph for this function:

4.6.3.2 getId()

```
int Employee::getId ( ) const
```

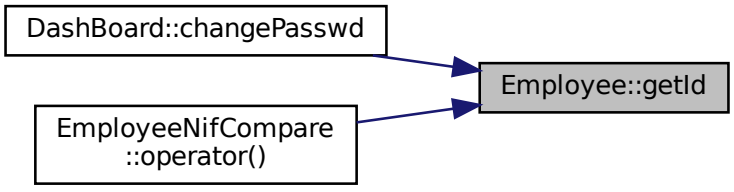
Definition at line 27 of file Employee.cpp.

```
27                      {
28   return id_;
29 }
```

References id_.

Referenced by DashBoard::changePasswd(), and EmployeeNifCompare::operator()().

Here is the caller graph for this function:



4.6.3.3 getNif()

```
int Employee::getNif ( ) const
```

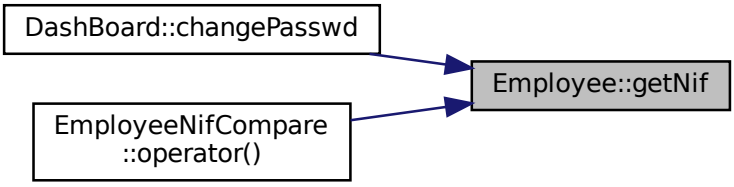
Definition at line 22 of file Employee.cpp.

```
22                      {
23   return nif_;
24 }
```

References nif_.

Referenced by DashBoard::changePasswd(), and EmployeeNifCompare::operator()().

Here is the caller graph for this function:



4.6.3.4 getPermission()

```
int Employee::getPermission ( ) const
```

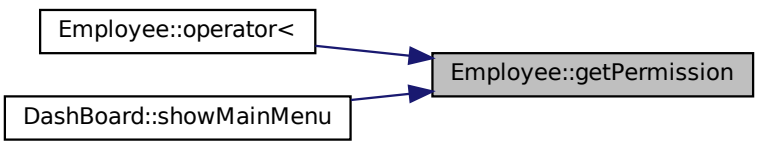
Definition at line 32 of file Employee.cpp.

```
32                            {
33   return permission_;
34 }
```

References permission_.

Referenced by operator<(), and DashBoard::showMainMenu().

Here is the caller graph for this function:



4.6.3.5 getTimestamp()

```
std::time_t Employee::getTimestamp ( )
```

Definition at line 37 of file Employee.cpp.

```
37                                  {
38   return timestamp_;
39 }
```

References timestamp_.

4.6.3.6 operator<()

```
bool Employee::operator< (
            const Employee & right ) const
```

Overloaded less-than operator to compare two employees.

Parameters

right	The employee to compare with.

Returns

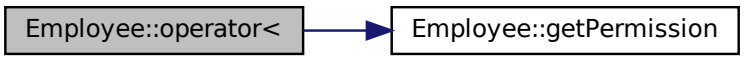
True if this employee is less than the other employee, false otherwise.

Definition at line 69 of file Employee.cpp.

```
70                                                  {
71   return (this->permission_ < right.getPermission());
72 }
```

References getPermission(), and permission_.

Here is the call graph for this function:



4.6.3.7 setId()

```
void Employee::setId (
            int id )
```

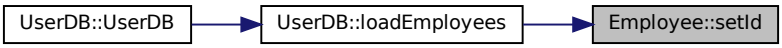
Definition at line 47 of file Employee.cpp.

```
47                      {
48   id_ = id;
49 }
```

References id_.

Referenced by UserDB::loadEmployees().

Here is the caller graph for this function:



4.6.3.8 setNif()

```
void Employee::setNif (
            int nif )
```

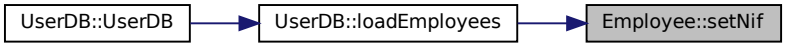
Definition at line 42 of file Employee.cpp.

```
42                      {
43   nif_ = nif;
44 }
```

References nif_.

Referenced by UserDB::loadEmployees().

Here is the caller graph for this function:



4.6.3.9 setPermission()

```
void Employee::setPermission (
            int permission )
```

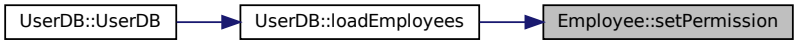
Definition at line 52 of file Employee.cpp.

```
52                                      {
53   permission_ = permission;
54 }
```

References permission_.

Referenced by UserDB::loadEmployees().

Here is the caller graph for this function:



4.6.3.10 setTimestamp()

```
void Employee::setTimestamp (
            std::time_t timestamp )
```

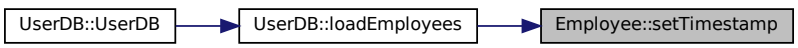
Definition at line 57 of file Employee.cpp.

```
57                                          {
58   timestamp_ = timestamp;
59 }
```

References timestamp_.

Referenced by UserDB::loadEmployees().

Here is the caller graph for this function:



4.6.3.11 showinfo()

```
void Employee::showinfo ( )
```

Displays information about the employee.

Definition at line 62 of file Employee.cpp.

```
62                  {
63   std::cout « "NIF: " « nif_ « std::endl;
64   std::cout « "ID: " « id_ « std::endl;
65   std::cout « "permission: " « permission_ « std::endl;
66   std::cout « "Timestamp: " « timestamp_ « std::endl;
67 }
```

References id_, nif_, permission_, and timestamp_.

4.6.4 Member Data Documentation

4.6.4.1 id_

```
int Employee::id_  [protected]
```

Unique identifier of the employee. Used as password in the login process.

Definition at line 21 of file Employee.h.

Referenced by Employee(), getId(), setId(), and showinfo().

4.6.4.2 nif_

```
int Employee::nif_  [protected]
```

National Identification Number of the employee.

Definition at line 20 of file Employee.h.

Referenced by Employee(), getNif(), setNif(), and showinfo().

4.6.4.3 permission_

```
int Employee::permission_  [protected]
```

Permission level of the employee (3 for admins, 2 for supervisors, 1 for workers).

Definition at line 22 of file Employee.h.

Referenced by Admin::Admin(), Employee(), getEmployType(), getPermission(), operator<(), setPermission(), showinfo(), Supervisor::Supervisor(), and Worker::Worker().

4.6.4.4 timestamp_

```
std::time_t Employee::timestamp_  [protected]
```

Timestamp indicating when the employee was created.

Definition at line 23 of file Employee.h.

Referenced by Employee(), getTimestamp(), setTimestamp(), and showinfo().

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

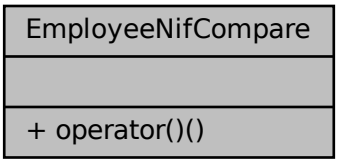
- include/Employee.h
- src/Employee.cpp

4.7 EmployeeNifCompare Struct Reference

For comparing Employees based on their NIF and ID.

```
#include <UserDB.h>
```

Collaboration diagram for EmployeeNifCompare:



Public Member Functions

- bool operator() (const Employee *left, const Employee *right) const
  Overloaded function call operator to compare two Employees.

4.7.1 Detailed Description

For comparing Employees based on their NIF and ID.

Definition at line 16 of file UserDB.h.

4.7.2 Member Function Documentation

4.7.2.1 operator()()

```
bool EmployeeNifCompare::operator() (
            const Employee * left,
            const Employee * right ) const  [inline]
```

Overloaded function call operator to compare two Employees.

Parameters

left	Pointer to the left Employee.
right	Pointer to the right Employee.

Returns

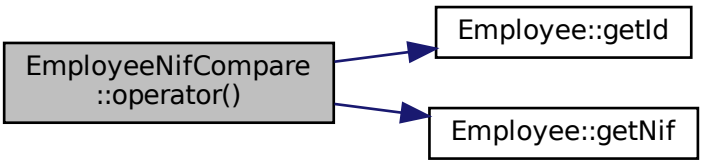
True if the Nif and the Id of the left is diferent from the right one.

Definition at line 23 of file UserDB.h.

```
23                                                                  {
24     return (left->getNif() != right->getNif()) && (left->getId() != right->getId());
25   }
```

References Employee::getId(), and Employee::getNif().

Here is the call graph for this function:



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

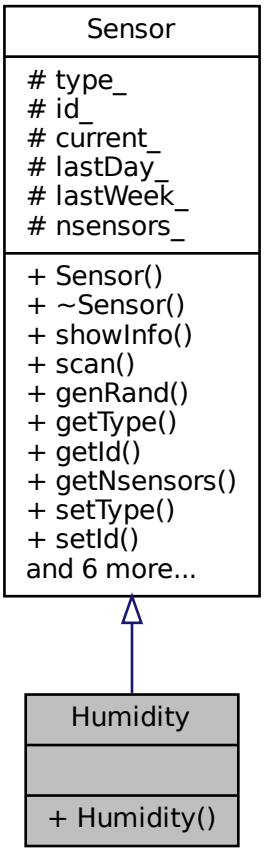
- include/UserDB.h

4.8 Humidity Class Reference

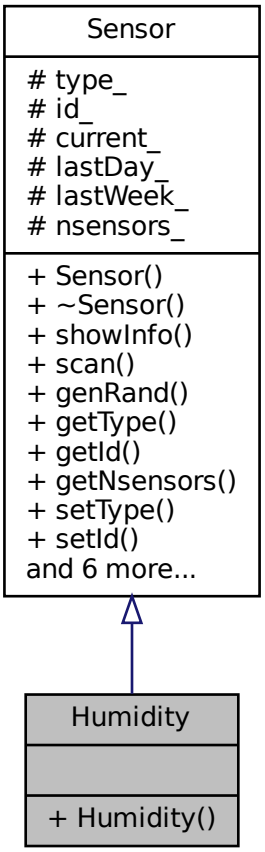
Represents a humidity Sensor.

`#include <Humidity.h>`

Inheritance diagram for Humidity:



Collaboration diagram for Humidity:



Public Member Functions

- Humidity (int id=0, int type=0)

  Constructor for the Humidity class.

Additional Inherited Members

4.8.1 Detailed Description

Represents a humidity Sensor.

This class represents a humidity sensor, which is a type of sensor specialized in measuring humidity.

Definition at line 17 of file Humidity.h.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 Humidity()

```
Humidity::Humidity (
            int id = 0,
            int type = 0 )  [inline]
```

Constructor for the Humidity class.

Parameters

id	Unique identifier of the humidity sensor.
type	Type of the humidity sensor.

< Set the sensor type to TYPE_HUM.

Definition at line 24 of file Humidity.h.

```
24                                  :Sensor(id, type) {
25     this->type_ = TYPE_HUM;
26   };
```

References Sensor::type_, and TYPE_HUM.

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

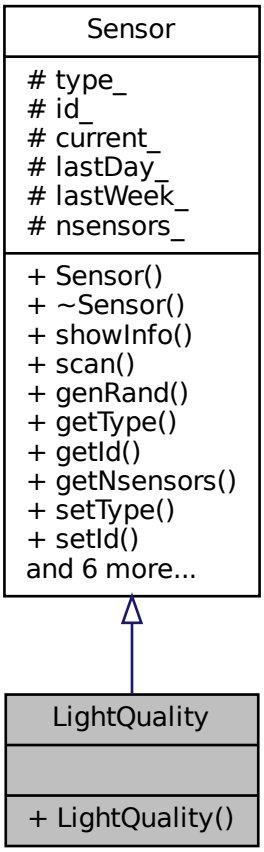
- include/Humidity.h

4.9 LightQuality Class Reference

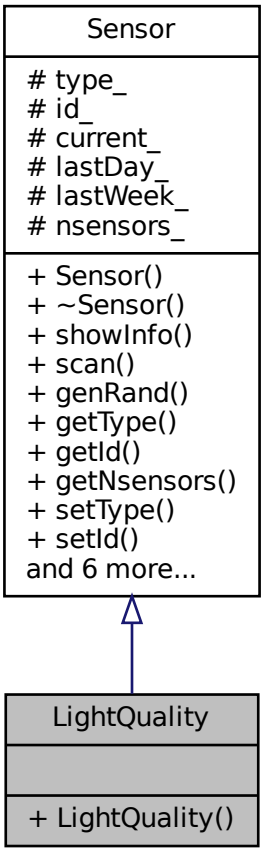
Represents a light quality Sensor.

```
#include <LightQuality.h>
```

Inheritance diagram for LightQuality:



Collaboration diagram for LightQuality:



Public Member Functions

- LightQuality (int id=0, int type=0)

  Constructor for the LightQuality class.

Additional Inherited Members

4.9.1 Detailed Description

Represents a light quality Sensor.

This class represents a light quality sensor, which is a type of sensor specialized in measuring light quality.

Definition at line 17 of file LightQuality.h.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 LightQuality()

```
LightQuality::LightQuality (
            int id = 0,
            int type = 0 )  [inline]
```

Constructor for the LightQuality class.

Parameters

id	Unique identifier of the light quality sensor.
type	Type of the light quality sensor.

< Set the sensor type to TYPE_LIGHT.

Definition at line 24 of file LightQuality.h.

```
24                                      :Sensor(id, type) {
25       this->type_ = TYPE_LIGHT;
26   };
```

References Sensor::type_, and TYPE_LIGHT.

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

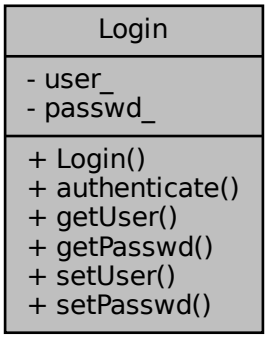
- include/LightQuality.h

4.10 Login Class Reference

Represents a login session.

`#include <Login.h>`

Collaboration diagram for Login:



Public Member Functions

- Login (int user_=0, int passwd_=0)

  Constructor for the Login class.
- Employee * authenticate (UserDB *userDB)

  Authenticates the user.
- int getUser ()
- int getPasswd ()
- void setUser (int user)
- void setPasswd (int passwd)

Private Attributes

- int user_
- int passwd_

4.10.1 Detailed Description

Represents a login session.

This class represents a login session with functionalities for authentication.

Definition at line 19 of file Login.h.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 Login()

```
Login::Login (
            int user_ = 0,
            int passwd_ = 0 )
```

Constructor for the Login class.

Parameters

*user_*	User NIF.
*passwd_*	Password(ID).

Definition at line 3 of file Login.cpp.

```
3                                {
4   this->user_ = user;
5   this->passwd_ = passwd;
6   std::cout « "Login" « std::endl;
7   std::cout « "introduce your NIF without the final letter: " « std::endl;
8   std::cin » this->user_;
9   std::cout « "introduce your password: " « std::endl;
```

```
10   std::cin » this->passwd_;
11 }
```

References passwd_, and user_.

4.10.3 Member Function Documentation

4.10.3.1 authenticate()

```
Employee * Login::authenticate (
             UserDB * userDB )
```

Authenticates the user.

Parameters

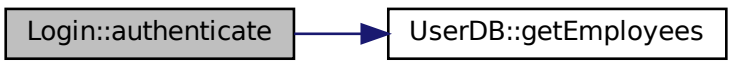
userDB	Pointer to the UserDB object containing user information.

Definition at line 14 of file Login.cpp.

```
14                                  {
15   std::cout « "Authenticating..." « std::endl;
16   for (const auto& employee: userDB->getEmployees()){
17     if (employee->getNif() == this->user_ && employee->getId() == this->passwd_){
18       std::cout « "You have been authenticated." « std::endl;
19       return employee;
20     }
21   }
22   std::cout « "NIF or password incorrect" « std::endl;
23   return nullptr;
24 }
```

References UserDB::getEmployees().

Here is the call graph for this function:



4.10.3.2 getPasswd()

```
int Login::getPasswd ( )
```

Definition at line 42 of file Login.cpp.

```
42                  {
43   return this->passwd_;
44 }
```

References passwd_.

4.10.3.3 getUser()

```
int Login::getUser ( )
```

Definition at line 37 of file Login.cpp.

```
37              {
38   return this->user_;
39 }
```

References user_.

4.10.3.4 setPasswd()

```
void Login::setPasswd (
            int passwd )
```

Definition at line 32 of file Login.cpp.

```
32                          {
33   this->passwd_ = passwd;
34 }
```

References passwd_.

4.10.3.5 setUser()

```
void Login::setUser (
            int user )
```

Definition at line 27 of file Login.cpp.

```
27                        {
28   this->user_ = user;
29 }
```

References user_.

4.10.4 Member Data Documentation

4.10.4.1 passwd_

```
int Login::passwd_  [private]
```

Pasword(ID).

Definition at line 22 of file Login.h.

Referenced by getPasswd(), Login(), and setPasswd().

4.10.4.2 user_

```
int Login::user_  [private]
```

User NIF.

Definition at line 21 of file Login.h.

Referenced by getUser(), Login(), and setUser().

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

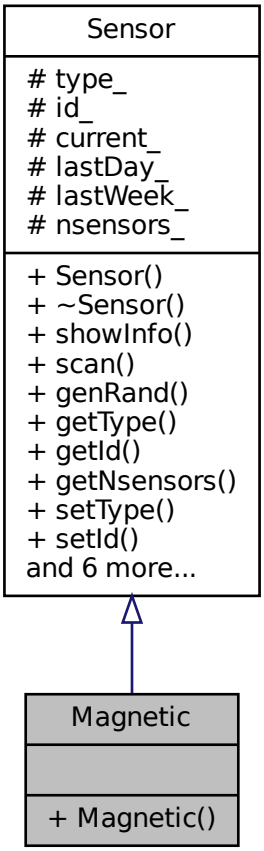
- include/Login.h
- src/Login.cpp

4.11 Magnetic Class Reference

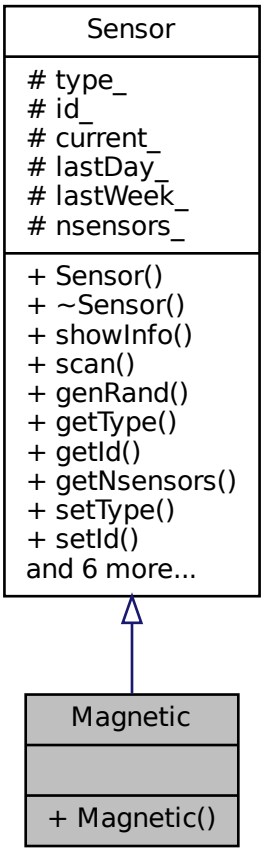
Represents a magnetic Sensor.

```
#include <Magnetic.h>
```

Inheritance diagram for Magnetic:



Collaboration diagram for Magnetic:



Public Member Functions

- Magnetic (int id=0, int type=0)

  Constructor for the Magnetic class.

Additional Inherited Members

4.11.1 Detailed Description

Represents a magnetic Sensor.

This class represents a magnetic sensor, which is a type of sensor specialized in measuring magnetic fields.

Definition at line 17 of file Magnetic.h.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 Magnetic()

```
Magnetic::Magnetic (
            int id = 0,
            int type = 0 )  [inline]
```

Constructor for the Magnetic class.

Parameters

id	Unique identisfier of the magnetic sensor.
type	Type of the magnetic sensor.

< Set the sensor type to TYPE_MAG.

< Set the initial current value for the magnetic sensor.

Definition at line 24 of file Magnetic.h.

```
24                                  :Sensor(id, type) {
25       this->type_ = TYPE_MAG;
26       this->current_[0] = 1.0;
27   };
```

References Sensor::current_, Sensor::type_, and TYPE_MAG.

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

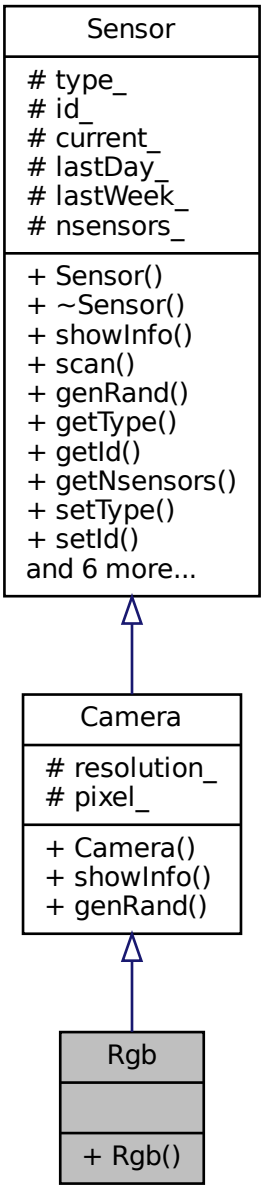
- include/Magnetic.h

4.12 Rgb Class Reference

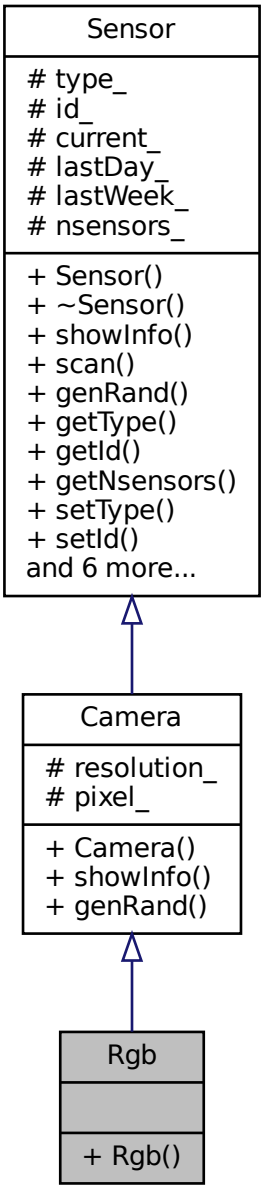
Represents an RGB Camera Sensor.

```
#include <RGB.h>
```

Inheritance diagram for Rgb:



Collaboration diagram for Rgb:



Public Member Functions

- Rgb (int id=0, int type=0, std::tuple< uint, uint > resolution=std::make_tuple(10, 10))

  Constructor for the Rgb class.

Additional Inherited Members

4.12.1 Detailed Description

Represents an RGB Camera Sensor.

This class represents an RGB sensor, which is a type of camera sensor specialized in capturing RGB images.

Definition at line 17 of file RGB.h.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 Rgb()

```
Rgb::Rgb (
            int id = 0,
            int type = 0,
            std::tuple< uint, uint > resolution = std::make_tuple(10, 10) )  [inline]
```

Constructor for the Rgb class.

Parameters

id	Unique identifier of the RGB sensor.
type	Type of the RGB sensor.
resolution	Resolution of the RGB sensor.

< Set the sensor type to TYPE_RGB.

Definition at line 25 of file RGB.h.

```
25                                                                                  :
26       Camera(id, type, resolution) {
27       this->type_ = TYPE_RGB;
28   };
```

References Sensor::type_, and TYPE_RGB.

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

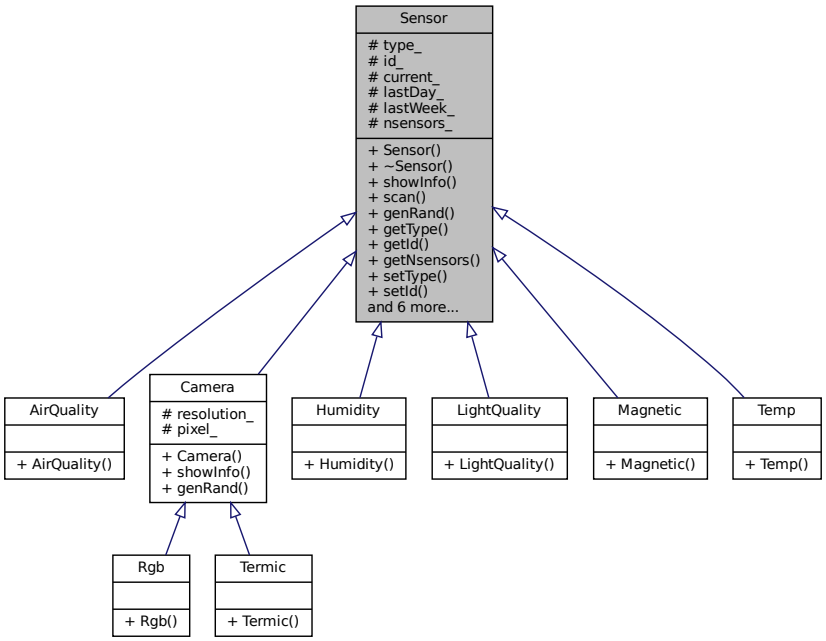
- include/RGB.h

4.13 Sensor Class Reference

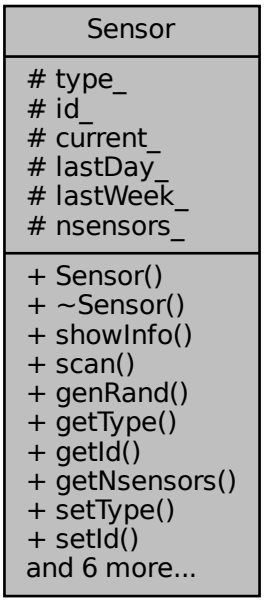
Represents a generic sensor.

```
#include <Sensor.h>
```

Inheritance diagram for Sensor:



Collaboration diagram for Sensor:



Public Member Functions

- Sensor (int type=0, int id=0)

  Constructor for the Sensor class.
- virtual ~Sensor ()

  Destructor for the Sensor class.
- virtual void showInfo ()

  Displays information about the sensor.
- void scan (double min, double max)

  Scans the sensor and updates its current value.
- double genRand (double min, double max)

  Generates a random value within the specified range. It gets current date in microseconds as seed for the random number generator.
- int getType ()
- int getId ()
- int getNsensors ()
- void setType (int type)
- void setId (int id)
- void setCurrent (double *current)
- void setLastDay (double *lastDay)
- void setLastWeek (double *lastWeek)
- double * getCurrent ()
- double * getLastDay ()
- double * getLastWeek ()

Protected Attributes

- int type_
- int id_
- double * current_
- double * lastDay_
- double * lastWeek_

Static Protected Attributes

- static int nsensors_ = 0

4.13.1 Detailed Description

Represents a generic sensor.

This class represents a generic sensor with functionality for scanning, calculating mean values, generating random values, and managing sensor information.

Definition at line 38 of file Sensor.h.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 Sensor()

```
Sensor::Sensor (
            int type = 0,
            int id = 0 )
```

Constructor for the Sensor class.

Parameters

type	Type of the sensor.
id	Unique identifier of the sensor.

Definition at line 5 of file Sensor.cpp.

```
5                                {
6   this->type_ = type;
7   this->id_ = id;
8   nsensors_++;
9   this->current_ = new double[1];
10   this->lastDay_ = new double[SAMPLES_DAY];
11   this->lastWeek_ = new double[SAMPLES_WEEK];
12 }
```

References current_, id_, lastDay_, lastWeek_, nsensors_, SAMPLES_DAY, SAMPLES_WEEK, and type_.

4.13.2.2 ~Sensor()

```
Sensor::~Sensor ( )  [virtual]
```

Destructor for the Sensor class.

Definition at line 19 of file Sensor.cpp.

```
19                {
20   delete[] this->current_;
21   delete[] this->lastDay_;
22   delete[] this->lastWeek_;
23 }
```

References current_, lastDay_, and lastWeek_.

4.13.3 Member Function Documentation

4.13.3.1 genRand()

```
double Sensor::genRand (
            double min,
            double max )
```

Generates a random value within the specified range. It gets current date in microseconds as seed for the random number generator.

Parameters

min	Minimum value for the random value.
max	Maximum value for the random value.

Returns

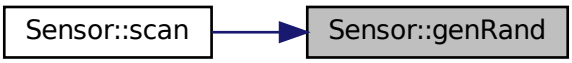
Random value within the specified range.

Definition at line 75 of file Sensor.cpp.

```
75                                  {
76   auto time_micros = std::chrono::duration_cast<std::chrono::microseconds>(
77   std::chrono::system_clock::now().time_since_epoch()).count(); // getting time in microseconds
78
79   srand(time_micros); // using current time in microseconds as seed for random generator
80   double randNum = min + (rand() / (double)RAND_MAX) * (max - min);
81   return randNum;
82 }
```

Referenced by scan().

Here is the caller graph for this function:



4.13.3.2 getCurrent()

```
double * Sensor::getCurrent ( )
```

Definition at line 127 of file Sensor.cpp.

```
127                  {
128   return this->current_;
129 }
```

References current_.

4.13.3.3 getId()

```
int Sensor::getId ( )
```

Definition at line 90 of file Sensor.cpp.

```
90              {
91   return this->id_;
92 }
```

References id_.

4.13.3.4 getLastDay()

```
double * Sensor::getLastDay ( )
```

Definition at line 132 of file Sensor.cpp.

```
132                     {
133   return this->lastDay_;
134 }
```

References lastDay_.

4.13.3.5 getLastWeek()

```
double * Sensor::getLastWeek ( )
```

Definition at line 137 of file Sensor.cpp.

```
137                     {
138   return this->lastWeek_;
139 }
```

References lastWeek_.

4.13.3.6 getNsensors()

```
int Sensor::getNsensors ( )
```

Definition at line 14 of file Sensor.cpp.

```
14                         {
15   return nsensors_;
16 }
```

References nsensors_.

4.13.3.7 getType()

```
int Sensor::getType ( )
```

Definition at line 85 of file Sensor.cpp.

```
85                  {
86   return this->type_;
87 }
```

References type_.

4.13.3.8 scan()

```
void Sensor::scan (
            double min,
            double max )
```

Scans the sensor and updates its current value.

Parameters

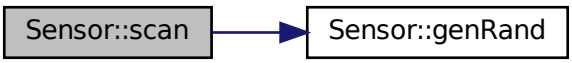
min	Minimum value for the sensor.
max	Maximum value for the sensor.

Definition at line 61 of file Sensor.cpp.

```
61                                  {
62   int i;
63
64   std::cout « "Scanning..." « std::endl;
65   this->current_[0] = 10.0;
66   for(i = 0; i < SAMPLES_DAY; i++){
67     this->lastDay_[i] = Sensor::genRand(min, max);
68   }
69   for(i = 0; i < SAMPLES_WEEK; i++){
70     this->lastWeek_[i] = Sensor::genRand(min, max);
71   }
72 }
```

References current_, genRand(), lastDay_, lastWeek_, SAMPLES_DAY, and SAMPLES_WEEK.

Here is the call graph for this function:



4.13.3.9 setCurrent()

```
void Sensor::setCurrent (
              double * current )
```

Definition at line 106 of file Sensor.cpp.

```
106                                  {
107   this->current_[0] = current[0];
108 }
```

References current_.

4.13.3.10 setId()

```
void Sensor::setId (
              int id )
```

Definition at line 100 of file Sensor.cpp.

```
100                    {
101
102   this->id_ = id;
103 }
```

References id_.

4.13.3.11 setLastDay()

```
void Sensor::setLastDay (
            double * lastDay )
```

Definition at line 111 of file Sensor.cpp.

```
111                                  {
112   std::cout « "Setting last day..." « std::endl;
113   for(int i = 0; i < SAMPLES_DAY; i++){
114     this->lastDay_[i] = lastDay[i];
115   }
116 }
```

References lastDay_, and SAMPLES_DAY.

4.13.3.12 setLastWeek()

```
void Sensor::setLastWeek (
            double * lastWeek )
```

Definition at line 119 of file Sensor.cpp.

```
119                                   {
120   std::cout « "Setting last week..." « std::endl;
121   for(int i = 0; i < SAMPLES_WEEK; i++){
122     this->lastWeek_[i] = lastWeek[i];
123   }
124 }
```

References lastWeek_, and SAMPLES_WEEK.

4.13.3.13 setType()

```
void Sensor::setType (
            int type )
```

Definition at line 95 of file Sensor.cpp.

```
95                          {
96   this->type_ = type;
97 }
```

References type_.

4.13.3.14 showInfo()

```
void Sensor::showInfo ( )  [virtual]
```

Displays information about the sensor.

Reimplemented in Camera.

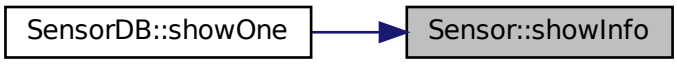
Definition at line 26 of file Sensor.cpp.

```
26                {
27   int i;
28
29   switch(this->type_){
30   case(TYPE_TEMP):
31     std::cout « "Type: " « this->type_ « ". Temperature.\n";
32     break;
33   case(TYPE_HUM):
34     std::cout « "Type: " « this->type_ « ". Humidity.\n";
35     break;
36   case(TYPE_MAG):
37     std::cout « "Type: " « this->type_ « ". Magnetic.\n";
38     break;
39   case(TYPE_LIGHT):
40     std::cout « "Type: " « this->type_ « ". Light.\n";
41     break;
42   case(TYPE_AIR):
43     std::cout « "Type: " « this->type_ « ". Air quality.\n";
44     break;
45   case(TYPE_CAM):
46     std::cout « "Type: " « this->type_ « ". Camera.\n";
47   }
48   std::cout « "Id: " « this->id_ « std::endl;
49   std::cout « "Current: " « this->current_[0] « std::endl;
50   std::cout « "Last day: [";
51   for(i = 0; i < SAMPLES_DAY; i++){
52     i != SAMPLES_DAY -1? std::cout « this->lastDay_[i] « ", ": std::cout « this->lastDay_[i] « "]\n";
53   }
54   std::cout « "Last week: [";
55   for(i = 0; i < SAMPLES_WEEK; i++){
56     i != SAMPLES_WEEK -1? std::cout « this->lastWeek_[i] « ", ": std::cout « this->lastWeek_[i] « "]\n";
57   }
58 }
```

References current_, id_, lastDay_, lastWeek_, SAMPLES_DAY, SAMPLES_WEEK, type_, TYPE_AIR, TYPE_↩CAM, TYPE_HUM, TYPE_LIGHT, TYPE_MAG, and TYPE_TEMP.

Referenced by SensorDB::showOne().

Here is the caller graph for this function:



4.13.4 Member Data Documentation

4.13.4.1 current_

```
double* Sensor::current_  [protected]
```

Array containing current sensor values.

Definition at line 43 of file Sensor.h.

Referenced by Camera::Camera(), getCurrent(), Magnetic::Magnetic(), scan(), Sensor(), setCurrent(), showInfo(), Camera::showInfo(), and ~Sensor().

4.13.4.2 id_

```
int Sensor::id_  [protected]
```

Unique identifier of the sensor.

Definition at line 41 of file Sensor.h.

Referenced by Camera::Camera(), getId(), Sensor(), setId(), showInfo(), and Camera::showInfo().

4.13.4.3 lastDay_

```
double* Sensor::lastDay_  [protected]
```

Array containing last day's sensor values.

Definition at line 44 of file Sensor.h.

Referenced by getLastDay(), scan(), Sensor(), setLastDay(), showInfo(), and ~Sensor().

4.13.4.4 lastWeek_

```
double* Sensor::lastWeek_  [protected]
```

Array containing last week's sensor values.

Definition at line 45 of file Sensor.h.

Referenced by getLastWeek(), scan(), Sensor(), setLastWeek(), showInfo(), and ~Sensor().

4.13.4.5 nsensors_

```
int Sensor::nsensors_ = 0  [static], [protected]
```

Number of sensors created.

Definition at line 42 of file Sensor.h.

Referenced by getNsensors(), and Sensor().

4.13.4.6 type_

```
int Sensor::type_  [protected]
```

Type of the sensor.

Definition at line 40 of file Sensor.h.

Referenced by AirQuality::AirQuality(), Camera::Camera(), getType(), Humidity::Humidity(), LightQuality::Light↩ Quality(), Magnetic::Magnetic(), Rgb::Rgb(), Sensor(), setType(), showInfo(), Camera::showInfo(), Temp::Temp(), and Termic::Termic().

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

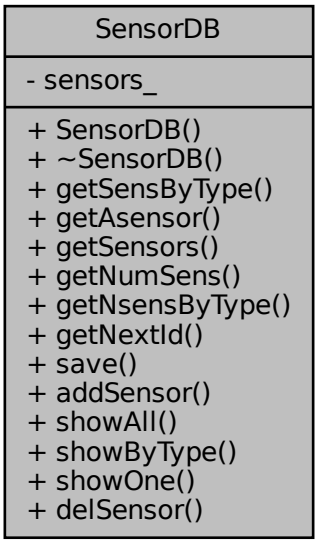
- include/Sensor.h
- src/Sensor.cpp

4.14 SensorDB Class Reference

Represents a database of sensors.

`#include <SensorDB.h>`

Collaboration diagram for SensorDB:



Public Member Functions

- SensorDB ()

  Constructor for the SensorDB class.
- ∼SensorDB ()

  Destructor for the SensorDB class.
- Sensor ∗ getSensByType (int type)

  Retrieves the sensors of the specified type.
- Sensor ∗ getAsensor (int id)

  Retrieves a sensor with the specified ID.
- std::vector< Sensor ∗ > getSensors ()
- int getNumSens ()

  Retrieves the number of sensors in the database.
- int getNsensByType (int type)

  Retrieves the number of sensors of the specified type in the database.
- int getNextId ()

  Retrieves the next available ID for a new sensor.
- int save ()

  Saves sensor information to a file.
- void addSensor (int type, int id)

  Adds a new sensor to the database.
- void showAll ()

  Displays information about all sensors in the database.
- void showByType (int type)

  Displays information about sensors of the specified type in the database.
- void showOne (int id)

  Displays information about the sensor with the specified ID.
- void delSensor (int id)

  Delete the sensor with the specified ID from the database.

Private Attributes

- std::vector< Sensor ∗ > sensors_

4.14.1 Detailed Description

Represents a database of sensors.

This class represents a database of sensors, with functionalities for adding, deleting, retrieving, and displaying sensor information.

Definition at line 29 of file SensorDB.h.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 SensorDB()

```
SensorDB::SensorDB ( )
```

Constructor for the SensorDB class.

Definition at line 35 of file SensorDB.cpp.

```
35                    {
36   std::string line, id, type;
37   uint nline = 1;
38   std::ifstream database("SensorDB.txt");
39
40   if (!database.is_open()){
41     std::cerr « "Error: Cannot open file SensorDB.txt" « std::endl;
42     exit(1);
43   }
44   while (getline(database, line)) { // Lee línea por línea del archivo
45     std::stringstream r_line(line);
46     if (!getline(r_line, id, '-') || !getline(r_line, type)) {
47       std::cerr « "Format error in file SensorDB.txt at line: " « nline « ". Skipping..." « std::endl;
48       nline++;
49       continue;
50     }
51     try{
52       SensorDB::addSensor(stoi(type), stoi(id));
53     }catch(std::exception &e){
54       throw;
55     }
56     nline++;
57   }
58   database.close();
59 }
```

References addSensor().

Here is the call graph for this function:

4.14.2.2 ~SensorDB()

```
SensorDB::~SensorDB ( )
```

Destructor for the SensorDB class.

Definition at line 61 of file SensorDB.cpp.

```
61                     {
62   for (Sensor *sensor : this->sensors_) {
63     delete sensor;
64   }
65   this->sensors_.clear();
66 }
```

References sensors_.

4.14.3 Member Function Documentation

4.14.3.1 addSensor()

```
void SensorDB::addSensor (
            int type,
            int id )
```

Adds a new sensor to the database.

Parameters

type	Type of the sensor to add.
id	ID of the sensor to add.

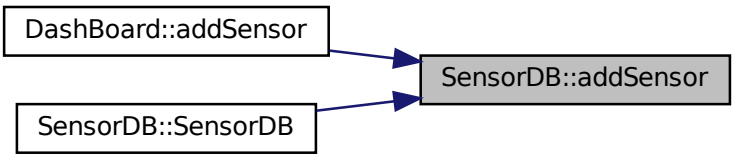
Definition at line 4 of file SensorDB.cpp.

```
4                                       {
5   Sensor *sensorptr;
6
7   switch(type) {
8   case TYPE_TEMP:
9     sensorptr = new Temp(id, type);
10    break;
11  case TYPE_HUM:
12    sensorptr = new Humidity(id, type);
13    break;
14  case TYPE_LIGHT:
15    sensorptr = new LightQuality(id, type);
16    break;
17  case TYPE_AIR:
18    sensorptr = new AirQuality(id, type);
19    break;
20  case TYPE_MAG:
21    sensorptr = new Magnetic(id, type);
22    break;
23  case TYPE_CAM:
24    sensorptr = new Camera(id, type);
25    break;
26  case TYPE_RGB:
27    sensorptr = new Rgb(id, type);
28    break;
29  default:
30    sensorptr = new Termic(id, type);
31  }
32  this->sensors_.push_back(sensorptr);
33 }
```

References sensors_, TYPE_AIR, TYPE_CAM, TYPE_HUM, TYPE_LIGHT, TYPE_MAG, TYPE_RGB, and TYPE_TEMP.

Referenced by DashBoard::addSensor(), and SensorDB().

Here is the caller graph for this function:



4.14.3.2 delSensor()

```
void SensorDB::delSensor (
            int id )
```

Delete the sensor with the specified ID from the database.

Parameters

id	ID of the sensor to delete.

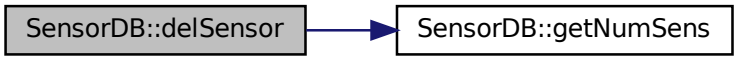
Definition at line 179 of file SensorDB.cpp.

```
179                             {
180   int i, pos = -1;
181   for(i = 0; i < SensorDB::getNumSens(); i++){
182     if(this->sensors_[i]->getId() == id){
183       pos = i;
184       break;
185     }
186   }
187   if(pos!= -1){
188     delete this->sensors_[pos];
189     this->sensors_.erase(this->sensors_.begin() + pos);
190   }else{
191     std::cout « "Sensor not found.\n";
192   }
193 }
```

References getNumSens(), and sensors_.

Referenced by DashBoard::delSensor().

Here is the call graph for this function:



Here is the caller graph for this function:

4.14.3.3 getAsensor()

```
Sensor * SensorDB::getAsensor (
            int id )
```

Retrieves a sensor with the specified ID.

Parameters

id	ID of the sensor to retrieve.

Returns

Pointer to the sensor with the specified ID, or nullptr if not found.

Definition at line 107 of file SensorDB.cpp.

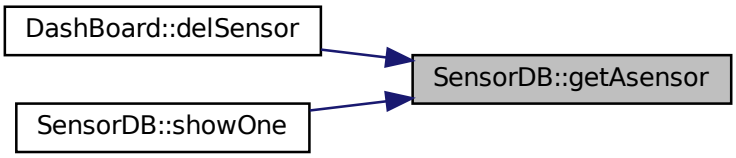
```
107                                {
108   int i;
109
110   for(i = 0; i < SensorDB::getNumSens(); i++){
111     if(this->sensors_[i]->getId() == id){
112       return this->sensors_[i];
113     }
114   }
115   return nullptr;
116 }
```

References getNumSens(), and sensors_.

Referenced by DashBoard::delSensor(), and showOne().

Here is the call graph for this function:

Here is the caller graph for this function:



4.14.3.4 getNextId()

```
int SensorDB::getNextId ( )
```

Retrieves the next available ID for a new sensor.

Returns

Next available ID for a new sensor.

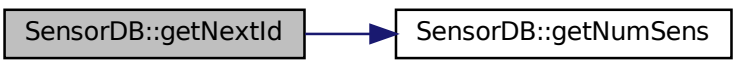
Definition at line 139 of file SensorDB.cpp.

```
139                     {
140   int i, id = 1;
141
142   for(i = 0; i < SensorDB::getNumSens(); i++){
143     if(this->sensors_[i]->getId() > id){
144       id = this->sensors_[i]->getId();
145     }
146   }
147   return id + 1;
148 }
```

References getNumSens(), and sensors_.

Referenced by DashBoard::addSensor().

Here is the call graph for this function:



Here is the caller graph for this function:

4.14.3.5 getNsensByType()

```
int SensorDB::getNsensByType (
            int type )
```

Retrieves the number of sensors of the specified type in the database.

Parameters

type	Type of sensors to count.

Returns

Number of sensors of the specified type in the database.

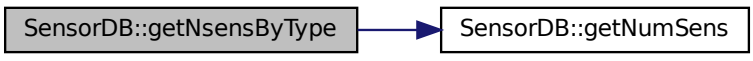
Definition at line 119 of file SensorDB.cpp.

```
119                                 {
120   int i, nsens = 0;
121   for(i = 0; i < SensorDB::getNumSens(); i++){
122     if(this->sensors_[i]->getType() == type){
123       nsens++;
124     }
125   }
126   return nsens;
127 }
```

References getNumSens(), and sensors_.

Referenced by getSensByType().

Here is the call graph for this function:



Here is the caller graph for this function:

4.14.3.6 getNumSens()

```
int SensorDB::getNumSens ( )
```

Retrieves the number of sensors in the database.

Returns

Number of sensors in the database.

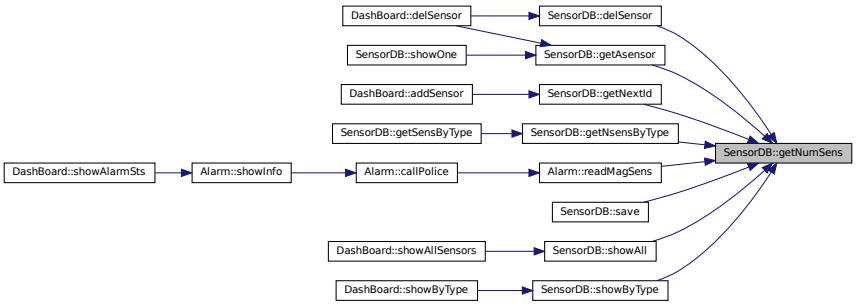
Definition at line 91 of file SensorDB.cpp.

```
91                      {
92   return this->sensors_[0]->getNsensors();
93 }
```

References sensors_.

Referenced by delSensor(), getAsensor(), getNextId(), getNsensByType(), Alarm::readMagSens(), save(), showAll(), and showByType().

Here is the caller graph for this function:



4.14.3.7 getSensByType()

```
Sensor * SensorDB::getSensByType (
            int type )
```

Retrieves the sensors of the specified type.

Parameters

type	Type of the sensor to retrieve.

Returns

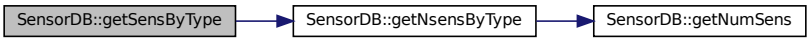
Pointer to the first sensor of the specified type, or nullptr if not found. It allocate memory dinamically.

Definition at line 69 of file SensorDB.cpp.

```
69                                  {
70   int i, j, nsens;
71   Sensor* senslist;
72
73   nsens = SensorDB::getNsensByType(type);
74   senslist = new Sensor[nsens]; // allocating memory for that number
75   j = 0;
76   for(i = 0; i < (int)this->sensors_.size(); i++){ // getting sens of that type
77     if(this->sensors_[i]->getType() == type){
78       senslist[j] = *this->sensors_[i];
79       j++;
80     }
81   }
82   return senslist;
83 }
```

References getNsensByType(), and sensors_.

Here is the call graph for this function:



4.14.3.8 getSensors()

```
std::vector< Sensor * > SensorDB::getSensors ( )
```

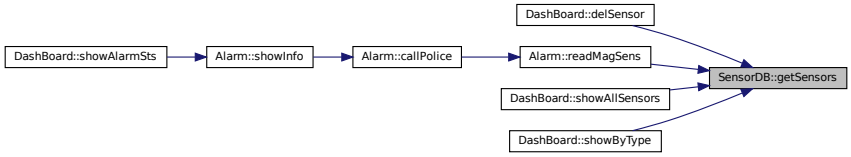
Definition at line 86 of file SensorDB.cpp.

```
86                                  {
87   return this->sensors_;
88 }
```

References sensors_.

Referenced by DashBoard::delSensor(), Alarm::readMagSens(), DashBoard::showAllSensors(), and DashBoard::showByType().

Here is the caller graph for this function:



4.14.3.9 save()

```
int SensorDB::save ( )
```

Saves sensor information to a file.

Returns

0 if successful, -1 otherwise.

Definition at line 152 of file SensorDB.cpp.

```
152             {
153   int i;
154   std::ofstream database("SensorDB.txt", std::ofstream::out | std::ofstream::trunc);
155 
156     // Verificar si el archivo se abrió correctamente
157   if (!database) {
158     std::cerr « "File 'SensorDB.txt' Not Found." « std::endl;
159     return 0;
160   }
161   for (i = 0; i < SensorDB::getNumSens(); i++) {
162     database « this->sensors_[i]->getId() « "-" « this->sensors_[i]->getType() « std::endl;
163   }
164   database.close();
165   return 1;
166 }
```

References getNumSens(), and sensors_.

Here is the call graph for this function:

4.14.3.10 showAll()

```
void SensorDB::showAll ( )
```

Displays information about all sensors in the database.

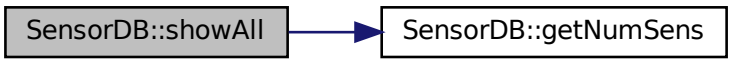
Definition at line 130 of file SensorDB.cpp.

```
130               {
131   int i;
132 
133   for(i = 0; i < SensorDB::getNumSens(); i++){
134     this->sensors_[i]->showInfo();
135   }
136 }
```

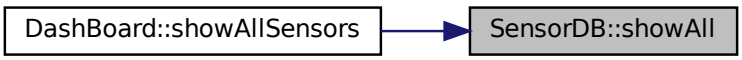
References getNumSens(), and sensors_.

Referenced by DashBoard::showAllSensors().

Here is the call graph for this function:



Here is the caller graph for this function:



4.14.3.11 showByType()

```
void SensorDB::showByType (
            int type )
```

Displays information about sensors of the specified type in the database.

Parameters

type	Type of sensors to display.

Definition at line 96 of file SensorDB.cpp.

```
96                              {
97   int i;
98
99   for(i = 0; i < SensorDB::getNumSens(); i++){
100     if(sensors_[i]->getType() == type){
101       sensors_[i]->showInfo();
102     }
103   }
104 }
```

References getNumSens(), and sensors_.

Referenced by DashBoard::showByType().

Here is the call graph for this function:

Here is the caller graph for this function:

4.14.3.12 showOne()

```
void SensorDB::showOne (
            int id )
```

Displays information about the sensor with the specified ID.

Parameters

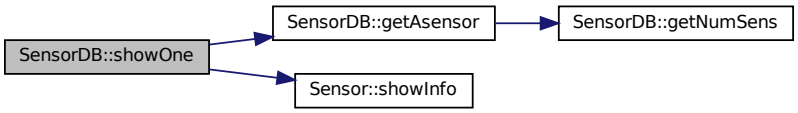
id	ID of the sensor to display.

Definition at line 169 of file SensorDB.cpp.

```
169                         {
170   Sensor *sens = this->getAsensor(id);
171
172   if (sens == nullptr)
173     std::cout « "Sensor not found.\n";
174   else
175     sens->showInfo();
176 }
```

References getAsensor(), and Sensor::showInfo().

Here is the call graph for this function:



4.14.4 Member Data Documentation

4.14.4.1 sensors_

```
std::vector<Sensor*> SensorDB::sensors_  [private]
```

Vector to store pointers to Sensor objects.

Definition at line 31 of file SensorDB.h.

Referenced by addSensor(), delSensor(), getAsensor(), getNextId(), getNsensByType(), getNumSens(), getSens↩ByType(), getSensors(), save(), showAll(), showByType(), and ~SensorDB().

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

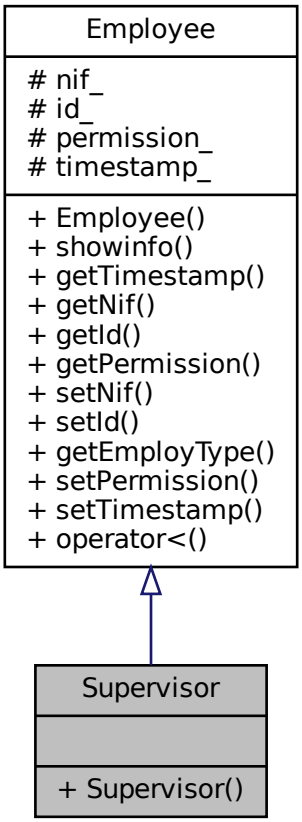
- include/SensorDB.h
- src/SensorDB.cpp

4.15 Supervisor Class Reference

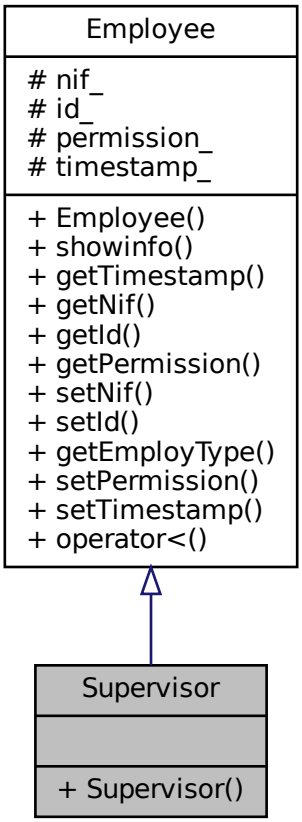
Represents a Supervisor, which is a type of Employee.

```
#include <Supervisor.h>
```

Inheritance diagram for Supervisor:



Collaboration diagram for Supervisor:



Public Member Functions

- Supervisor (int nif=0, int id=0)

  Constructor for the Supervisor class.

Additional Inherited Members

4.15.1 Detailed Description

Represents a Supervisor, which is a type of Employee.

This class represents a Supervisor, which is a type of Employee with intermediate permissions. He can manage alarms but cannot manage user's databases.

Definition at line 17 of file Supervisor.h.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 Supervisor()

```
Supervisor::Supervisor (
            int nif = 0,
            int id = 0 )  [inline]
```

Constructor for the Supervisor class.

Parameters

nif	National Identification Number of the supervisor.
id	Unique identifier of the supervisor.

< Set permission level to 2 for supervisors.

Definition at line 24 of file Supervisor.h.

```
24                                  :Employee(nif, id) {
25       this->permission_ = 2;
26   };
```

References Employee::permission_.

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

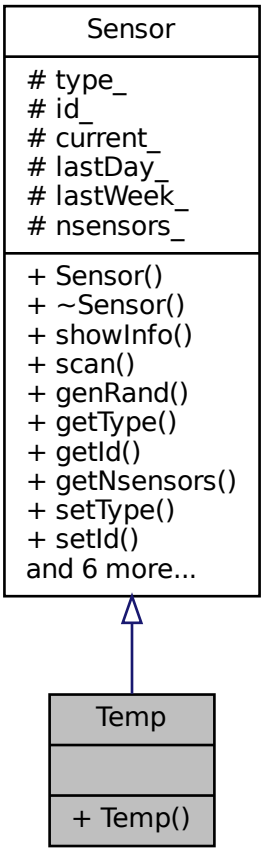
- include/Supervisor.h

4.16 Temp Class Reference

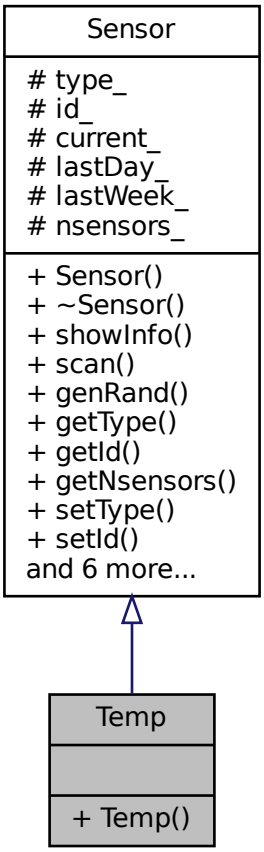
Represents a temperature Sensor.

```
#include <Temp.h>
```

Inheritance diagram for Temp:



Collaboration diagram for Temp:



Public Member Functions

- Temp (int id=0, int type=0)

  Constructor for the Temp class.

Additional Inherited Members

4.16.1 Detailed Description

Represents a temperature Sensor.

This class represents a temperature sensor, which is a type of sensor specialized in measuring temperature.

Definition at line 17 of file Temp.h.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 Temp()

```
Temp::Temp (
            int id = 0,
            int type = 0 )  [inline]
```

Constructor for the Temp class.

Parameters

id	Unique identifier of the temperature sensor.
type	Type of the temperature sensor.

< Set the sensor type to TYPE_TEMP.

Definition at line 24 of file Temp.h.

```
24                              :Sensor(id, type) {
25       this->type_ = TYPE_TEMP;
26   };
```

References Sensor::type_, and TYPE_TEMP.

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

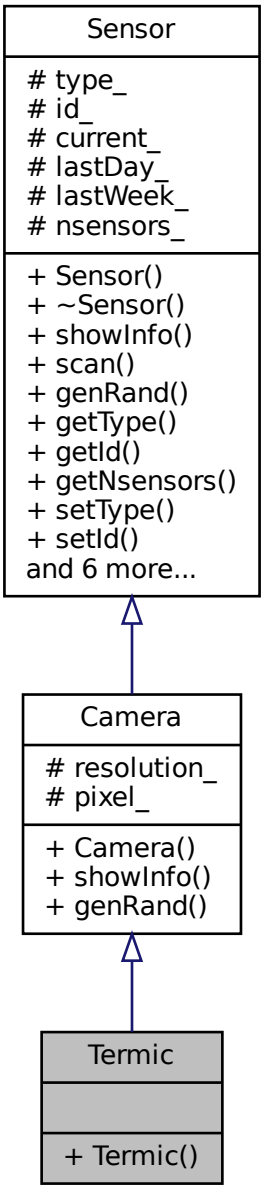
- include/Temp.h

4.17 Termic Class Reference

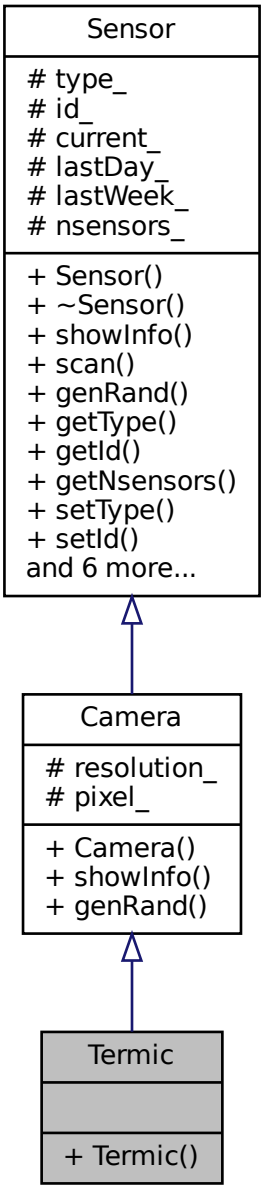
Represents a thermal Camera Sensor.

```
#include <Termic.h>
```

Inheritance diagram for Termic:



Collaboration diagram for Termic:



Public Member Functions

- Termic (int id=0, int type=0, std::tuple< uint, uint > resolution=std::make_tuple(10, 10))

  Constructor for the Termic class.

Additional Inherited Members

4.17.1 Detailed Description

Represents a thermal Camera Sensor.

This class represents a thermal camera sensor, which is a type of camera sensor specialized in capturing thermal images.

Definition at line 17 of file Termic.h.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 Termic()

```
Termic::Termic (
            int id = 0,
            int type = 0,
            std::tuple< uint, uint > resolution = std::make_tuple(10, 10) )  [inline]
```

Constructor for the Termic class.

Parameters

id	Unique identifier of the thermal camera sensor.
type	Type of the thermal camera sensor.
resolution	Resolution of the thermal camera sensor.

< Set the sensor type to TYPE_TERMIC.

Definition at line 25 of file Termic.h.

```
25                                                                                  :
26       Camera(id, type, resolution) {
27       this->type_ = TYPE_TERMIC;
28   };
```

References Sensor::type_, and TYPE_TERMIC.

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

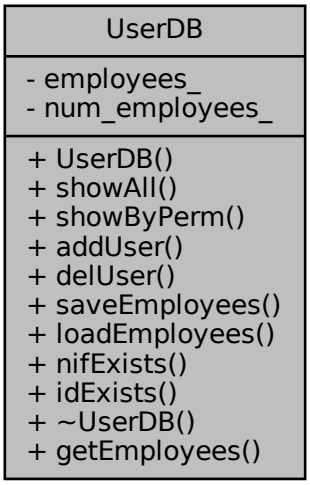
- include/Termic.h

4.18 UserDB Class Reference

Represents a User's database.

```
#include <UserDB.h>
```

Collaboration diagram for UserDB:



Public Member Functions

- UserDB ()

  Default constructor for the UserDB class.
- void showAll ()

  Displays information about all employees in the database.
- void showByPerm (int permission)

  Displays information about employees with a specific permission level.
- void addUser (Employee ∗employee)

  Adds a new employee to the database.
- void delUser (int index)

  Deletes an employee from the database.
- int saveEmployees ()

  Saves all employees to a file.
- void loadEmployees ()

  Loads employees from a file.
- bool nifExists (int nif)

  Checks if an employee with the given NIF exists in the database.
- bool idExists (int id)

  Checks if an employee with the given ID exists in the database.
- ∼UserDB ()
- std::set< Employee ∗, EmployeeNifCompare > getEmployees ()

Private Attributes

- std::set< Employee ∗, EmployeeNifCompare > employees_
- int num_employees_

4.18.1 Detailed Description

Represents a User's database.

This class is a container for the users. It is a set of employees and it is used to manage them.

Definition at line 34 of file UserDB.h.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 UserDB()

```
UserDB::UserDB ( )
```

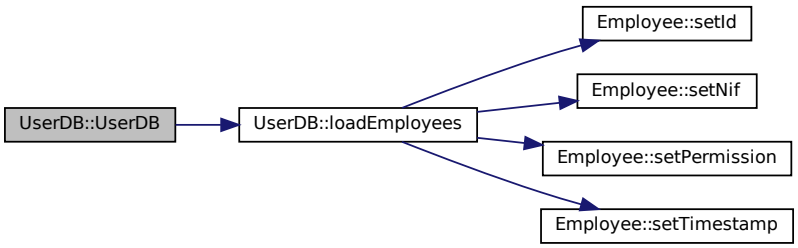
Default constructor for the UserDB class.

Definition at line 64 of file UserDB.cpp.

```
64                  {
65   this->num_employees_ = 0;
66   this->loadEmployees();
67 }
```

References loadEmployees(), and num_employees_.

Here is the call graph for this function:



4.18.2.2 ~UserDB()

```
UserDB::~UserDB ( )
```

Definition at line 130 of file UserDB.cpp.

```
130                  {
131   for (const auto& employee: employees_){
132     delete employee;
133   }
134   employees_.clear();
135 }
```

References employees_.

4.18.3 Member Function Documentation

4.18.3.1 addUser()

```
void UserDB::addUser (
            Employee * employee )
```

Adds a new employee to the database.

Parameters

employee	Pointer to the Employee object to be added.

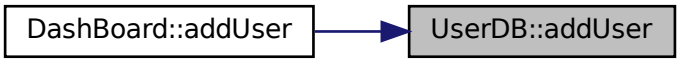
Definition at line 70 of file UserDB.cpp.

```
70                                  {
71
72   std::cout « "Adding user..." « std::endl;
73   employees_.insert(employee);
74   num_employees_++;
75 }
```

References employees_, and num_employees_.

Referenced by DashBoard::addUser().

Here is the caller graph for this function:



4.18.3.2 delUser()

```
void UserDB::delUser (
            int index )
```

Deletes an employee from the database.

Parameters

index	The index of the employee to be deleted.

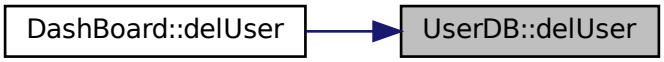
Definition at line 97 of file UserDB.cpp.

```
97                          {
98   std::cout « "Deleting user..." « std::endl;
99   auto usertodel = employees_.begin();
100   std::advance(usertodel, index);
101   employees_.erase(usertodel);
102   delete *usertodel;
103   num_employees_--;
104 }
```

References employees_, and num_employees_.

Referenced by DashBoard::delUser().

Here is the caller graph for this function:



4.18.3.3 getEmployees()

```
std::set<Employee*, EmployeeNifCompare> UserDB::getEmployees ( )  [inline]
```

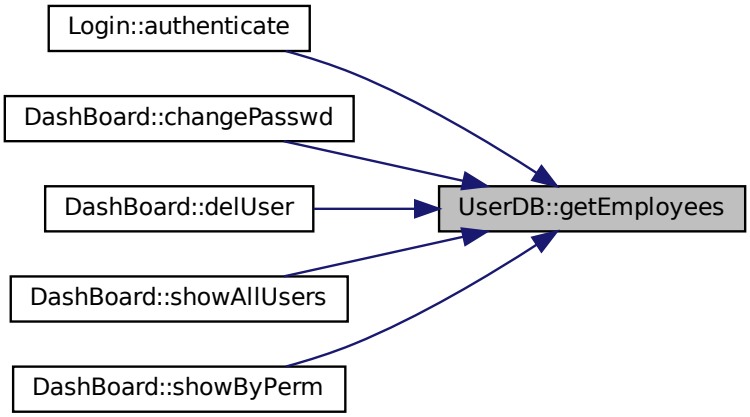
Definition at line 92 of file UserDB.h.

```
92 { return employees_; };
```

References employees_.

Referenced by Login::authenticate(), DashBoard::changePasswd(), DashBoard::delUser(), DashBoard::showAll↩ Users(), and DashBoard::showByPerm().

Here is the caller graph for this function:



4.18.3.4 idExists()

```
bool UserDB::idExists (
            int id )
```

Checks if an employee with the given ID exists in the database.

Parameters

id	The ID to check for.

Returns

True if an employee with the given ID exists, false otherwise.

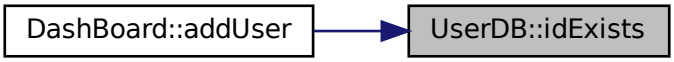
Definition at line 5 of file UserDB.cpp.

```
5                           {
6   for (const auto& employee: this->employees_){
7     if (employee->getId() == id) {
8       return true;
9     }
10   }
11   return false;
12 }
```

References employees_.

Referenced by DashBoard::addUser().

Here is the caller graph for this function:



4.18.3.5 loadEmployees()

```
void UserDB::loadEmployees ( )
```

Loads employees from a file.

Definition at line 24 of file UserDB.cpp.

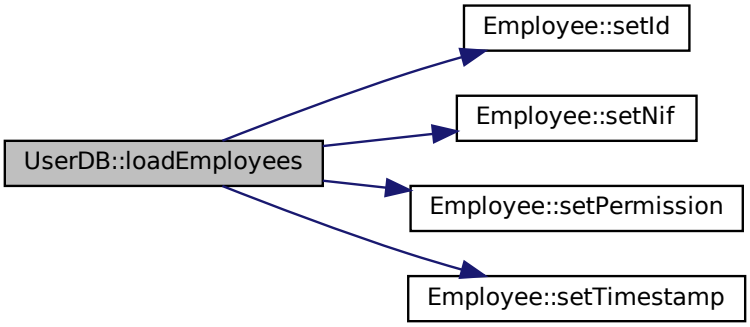
```
24                          {
25   int nif, id, permission;
26   std::time_t timestamp;
27
28   std::ifstream ifs("UserDB.dat", std::ios::in | std::ios::binary);
29   if (!ifs) {
30     std::ofstream ofs("UserDB.dat", std::ios::out | std::ios::binary);
31     if (!ofs) {
32       throw std::runtime_error("Could not open UserDB.dat");
```

```
33      }
34      ofs.close();
35    }
36
37    while(true){
38      Employee* employee = new Employee();
39      if (!ifs.read(reinterpret_cast<char*>(&nif), sizeof(nif))) {
40        delete employee;
41        break;
42      }
43      if (!ifs.read(reinterpret_cast<char*>(&id), sizeof(id))) {
44        delete employee;
45        break;
46      }
47      if (!ifs.read(reinterpret_cast<char*>(&permission), sizeof(permission))) {
48        delete employee;
49        break;
50      }
51      if (!ifs.read(reinterpret_cast<char*>(&timestamp), sizeof(timestamp))) {
52        delete employee;
53        break;
54      }
55      employee->setNif(nif);
56      employee->setId(id);
57      employee->setPermission(permission);
58      employee->setTimestamp(timestamp);
59      employees_.insert(employee);
60    }
61    ifs.close();
62  }
```

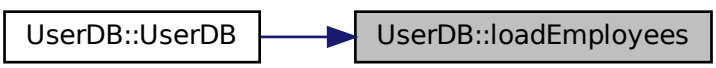
References employees_, Employee::setId(), Employee::setNif(), Employee::setPermission(), and Employee::set↩ Timestamp().

Referenced by UserDB().

Here is the call graph for this function:



Here is the caller graph for this function:



4.18.3.6 nifExists()

```
bool UserDB::nifExists (
            int nif )
```

Checks if an employee with the given NIF exists in the database.

Parameters

nif	The NIF to check for.

Returns

True if an employee with the given NIF exists, false otherwise.

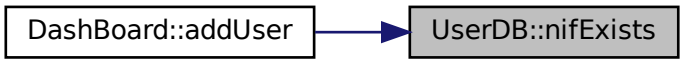
Definition at line 15 of file UserDB.cpp.

```
15                          {
16   for (const auto& employee: this->employees_){
17     if (employee->getNif() == nif) {
18       return true;
19     }
20   }
21   return false;
22 }
```

References employees_.

Referenced by DashBoard::addUser().

Here is the caller graph for this function:



4.18.3.7 saveEmployees()

```
int UserDB::saveEmployees ( )
```

Saves all employees to a file.

Returns

1 if employees were saved successfully, 0 otherwise.

Definition at line 107 of file UserDB.cpp.

```
107                       {
108   std::ofstream ofs("UserDB.dat", std::ios::out | std::ios::binary);
109   if (!ofs) {
110     return 0;
111   }
112   int id;
113   int nif;
114   std::time_t timestamp;
115   int permission;
116   for (const auto& employee : employees_) {
117     nif = employee->getNif();
118     id = employee->getId();
119     timestamp = employee->getTimestamp();
120     permission = employee->getPermission();
121     ofs.write(reinterpret_cast<const char*>(&nif), sizeof(nif));
122     ofs.write(reinterpret_cast<const char*>(&id), sizeof(id));
123     ofs.write(reinterpret_cast<const char*>(&permission), sizeof(permission));
124     ofs.write(reinterpret_cast<const char*>(&timestamp), sizeof(timestamp));
125   }
126   ofs.close();
127   return 1;
128 }
```

References employees_.

4.18.3.8 showAll()

```
void UserDB::showAll ( )
```

Displays information about all employees in the database.

Definition at line 79 of file UserDB.cpp.

```
79                {
80   std::cout « "Showing all users..." « std::endl;
81   for (const auto& employee: employees_){
82     employee->showinfo();
83   }
84 }
```

References employees_.

4.18.3.9 showByPerm()

```
void UserDB::showByPerm (
              int permission )
```

Displays information about employees with a specific permission level.

Parameters

permission	The permission level to filter employees by.

Definition at line 87 of file UserDB.cpp.

```
87                                {
88   std::cout « "Showing users by permision..." « std::endl;
89   for (const auto& employee: employees_){
90     if (employee->getPermission() == permision){
91       employee->showinfo();
92     }
93   }
94 }
```

References employees_.

4.18.4 Member Data Documentation

4.18.4.1 employees_

```
std::set<Employee*, EmployeeNifCompare> UserDB::employees_  [private]
```

Set of employees stored in the database.

Definition at line 36 of file UserDB.h.

Referenced by addUser(), delUser(), getEmployees(), idExists(), loadEmployees(), nifExists(), saveEmployees(), showAll(), showByPerm(), and ∼UserDB().

4.18.4.2 num_employees_

```
int UserDB::num_employees_  [private]
```

Number of employees in the database.

Definition at line 37 of file UserDB.h.

Referenced by addUser(), delUser(), and UserDB().

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

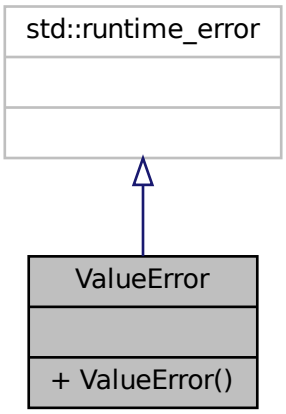
- include/UserDB.h
- src/UserDB.cpp

4.19 ValueError Class Reference

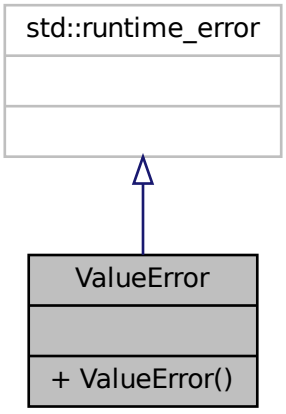
Represents a value error exception.

`#include <ValueError.h>`

Inheritance diagram for ValueError:



Collaboration diagram for ValueError:



Public Member Functions

- ValueError ()

  Constructor for the ValueError class.

4.19.1 Detailed Description

Represents a value error exception.

This class represents a custom exception for value errors, derived from std::runtime_error.

Definition at line 17 of file ValueError.h.

4.19.2 Constructor & Destructor Documentation

4.19.2.1 ValueError()

```
ValueError::ValueError ( )  [inline]
```

Constructor for the ValueError class.

Definition at line 22 of file ValueError.h.

```
23     : std::runtime_error("Value error") {};
```

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

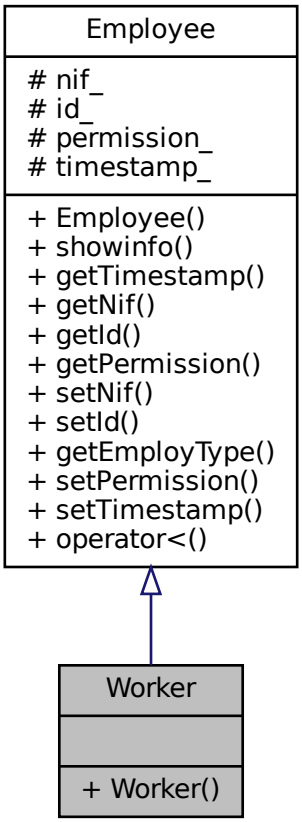
- include/ValueError.h

4.20 Worker Class Reference

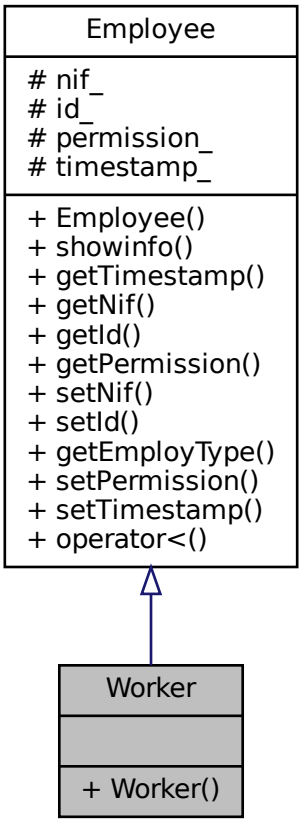
Represents a Worker, which is a type of Employee.

```
#include <Worker.h>
```

Inheritance diagram for Worker:



Collaboration diagram for Worker:



Public Member Functions

- Worker (int nif=0, int id=0)

  Constructor for the Worker class.

Additional Inherited Members

4.20.1 Detailed Description

Represents a Worker, which is a type of Employee.

This class represents a Worker, which is a type of Employee with basic permissions. He can only manage sensors(except alarms).

Definition at line 15 of file Worker.h.

4.20.2 Constructor & Destructor Documentation

4.20.2.1 Worker()

```
Worker::Worker (
            int nif = 0,
            int id = 0 )  [inline]
```

Constructor for the Worker class.

Parameters

nif	National Identification Number of the worker.
id	Unique identifier of the worker.

< Set permission level to 1 for workers.

Definition at line 22 of file Worker.h.

```
22                                  :Employee(nif, id) {
23       this->permission_ = 1;
24   };
```

References Employee::permission_.

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

- include/Worker.h

Chapter 5

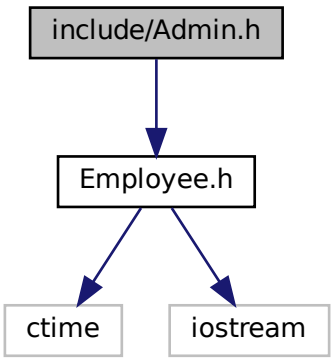
File Documentation

5.1 include/Admin.h File Reference

This file contains the declaration of the Admin class.

```
#include "Employee.h"
```

Include dependency graph for Admin.h:



Classes

- class Admin

  Represents an Administrator, which is a type of Employee.

5.1.1 Detailed Description

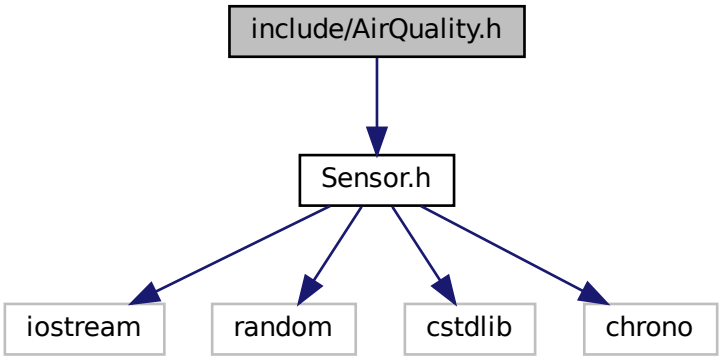
This file contains the declaration of the Admin class.

5.2 include/AirQuality.h File Reference

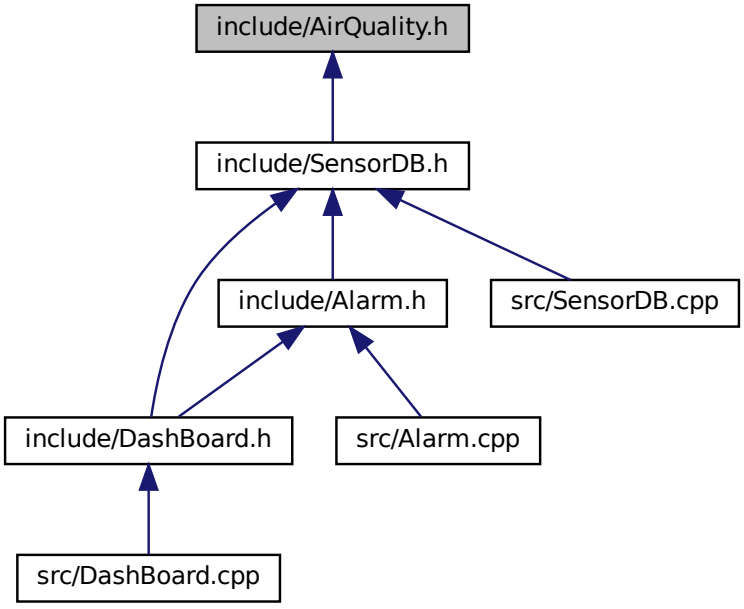
This file contains the declaration of the AirQuality class.

```
#include "Sensor.h"
```

Include dependency graph for AirQuality.h:



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



Classes

- class AirQuality

  Represents a Sensor for measuring air quality.

5.2.1 Detailed Description

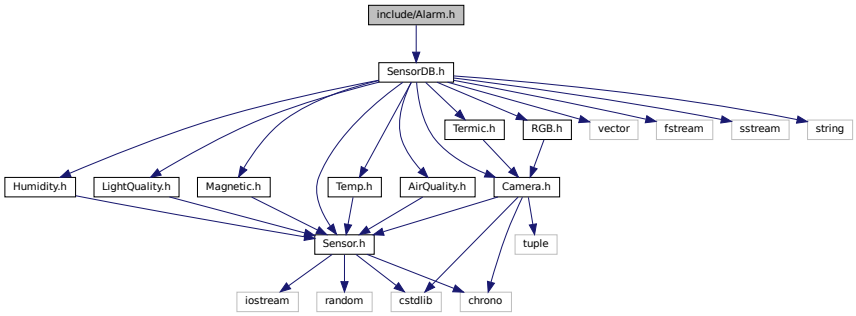
This file contains the declaration of the AirQuality class.

5.3 include/Alarm.h File Reference

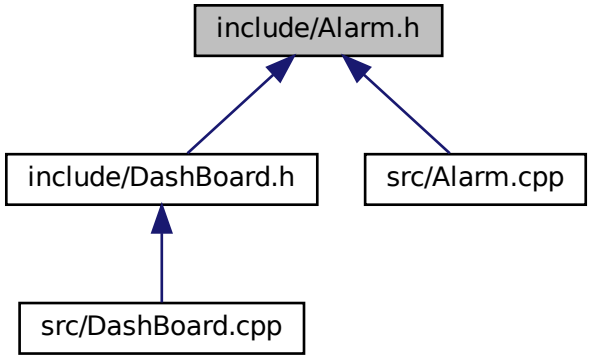
This file contains the declaration of the Alarm class.

```
#include "SensorDB.h"
```

Include dependency graph for Alarm.h:



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



Classes

- class Alarm

  Represents an Alarm system.

5.3.1 Detailed Description

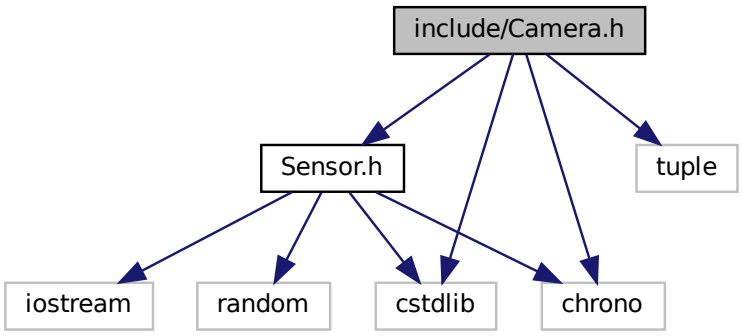
This file contains the declaration of the Alarm class.

5.4 include/Camera.h File Reference

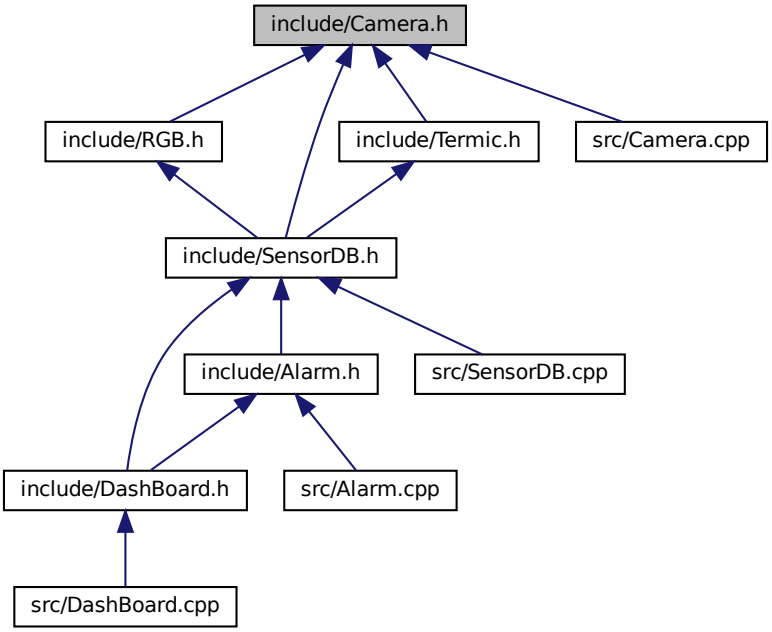
This file contains the declaration of the Camera class.

```
#include "Sensor.h"
#include <tuple>
#include <cstdlib>
#include <chrono>
```

Include dependency graph for Camera.h:



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



Classes

- class Camera

  Represents a camera Sensor.

5.4.1 Detailed Description

This file contains the declaration of the Camera class.

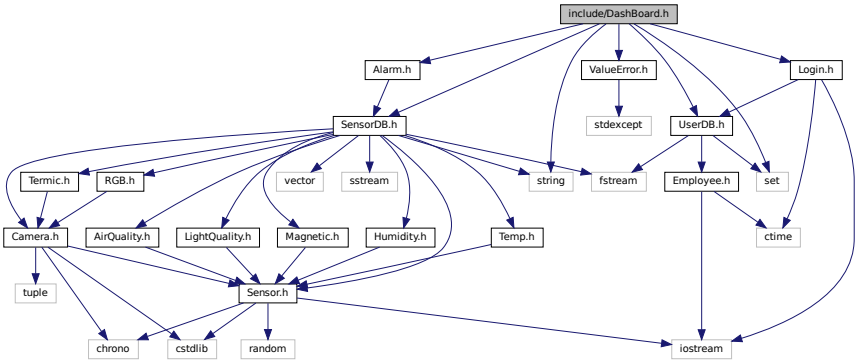
5.5 include/DashBoard.h File Reference

This file contains the declaration of the DashBoard class.

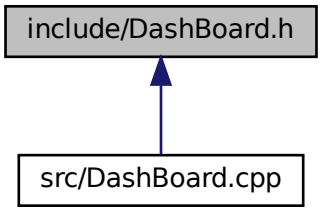
```
#include "UserDB.h"
#include "SensorDB.h"
#include "Alarm.h"
#include "ValueError.h"
#include "Login.h"
#include <string>
```

```
#include <set>
```

Include dependency graph for DashBoard.h:



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



Classes

- class DashBoard

  Represents a dashboard for system management.

5.5.1 Detailed Description

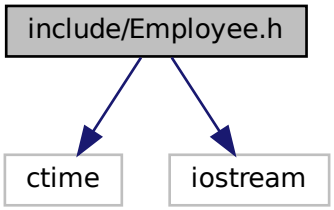
This file contains the declaration of the DashBoard class.

5.6 include/Employee.h File Reference

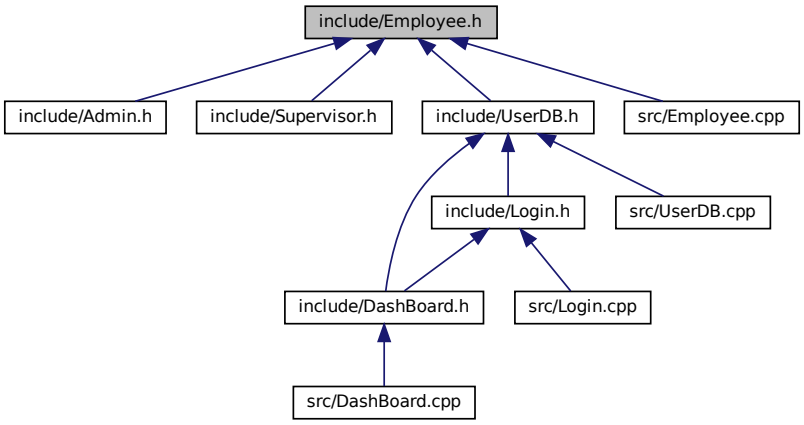
This file contains the declaration of the Employee class.

```
#include <ctime>
#include <iostream>
```

Include dependency graph for Employee.h:



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



Classes

- class Employee

  Represents an Employee.

5.6.1 Detailed Description

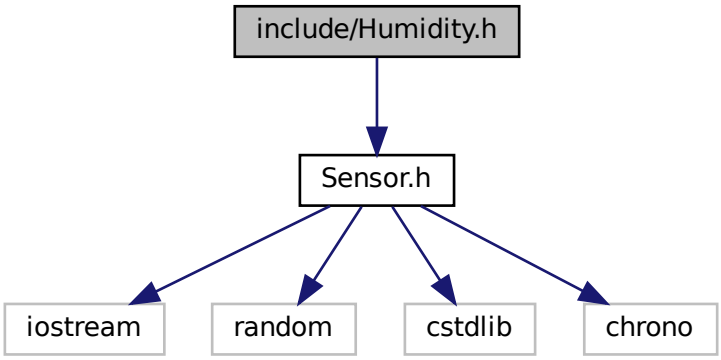
This file contains the declaration of the Employee class.

5.7 include/Humidity.h File Reference

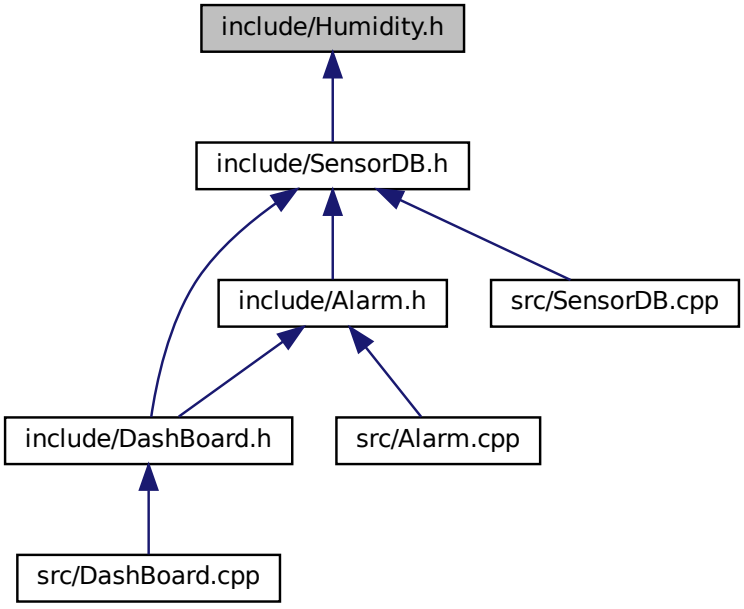
This file contains the declaration of the Humidity class.

```
#include "Sensor.h"
```

Include dependency graph for Humidity.h:



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



Classes

- class Humidity

  Represents a humidity Sensor.

5.7.1 Detailed Description

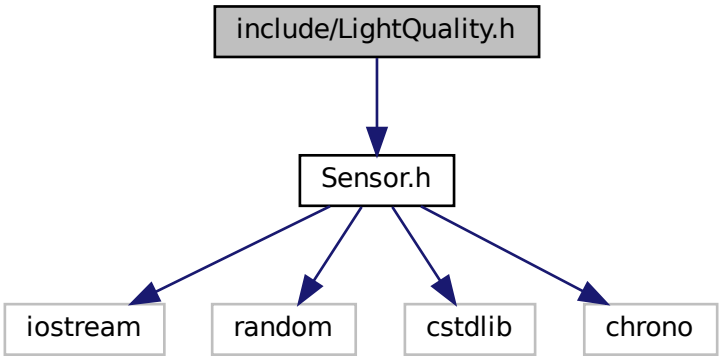
This file contains the declaration of the Humidity class.

5.8 include/LightQuality.h File Reference

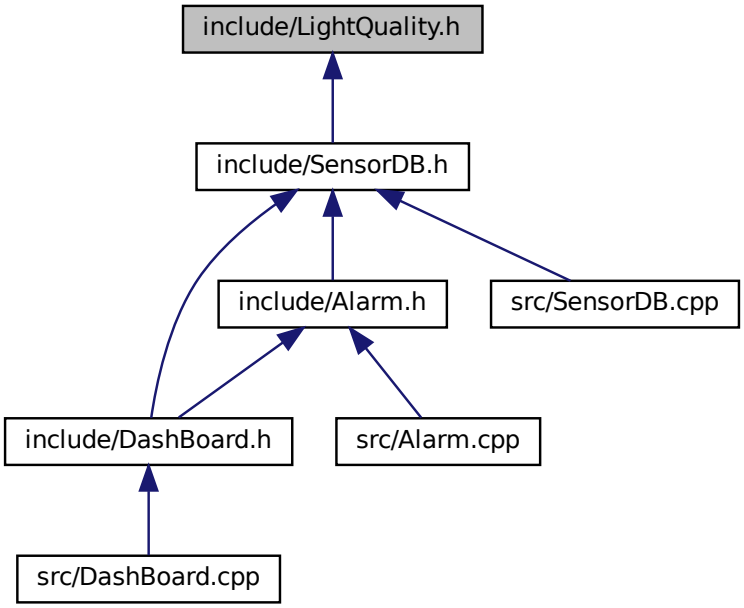
This file contains the declaration of the LightQuality class.

```
#include "Sensor.h"
```

Include dependency graph for LightQuality.h:



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



Classes

- class LightQuality

  Represents a light quality Sensor.

5.8.1 Detailed Description

This file contains the declaration of the LightQuality class.

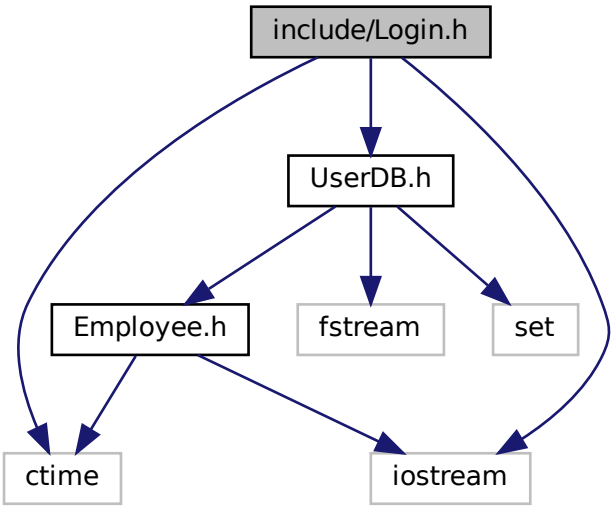
5.9 include/Login.h File Reference

This file contains the declaration of the Login class.

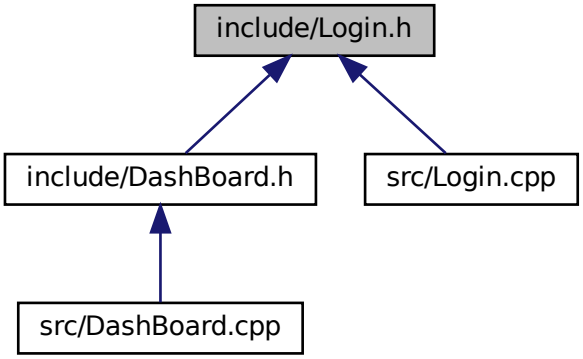
```
#include <ctime>
#include <iostream>
```

```
#include "UserDB.h"
```

Include dependency graph for Login.h:



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



Classes

- class Login

  Represents a login session.

5.9.1 Detailed Description

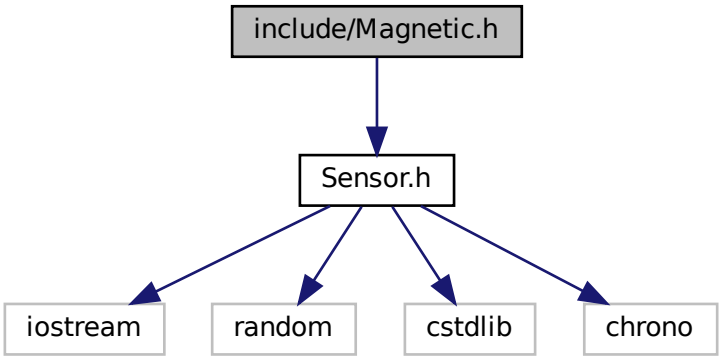
This file contains the declaration of the Login class.

5.10 include/Magnetic.h File Reference

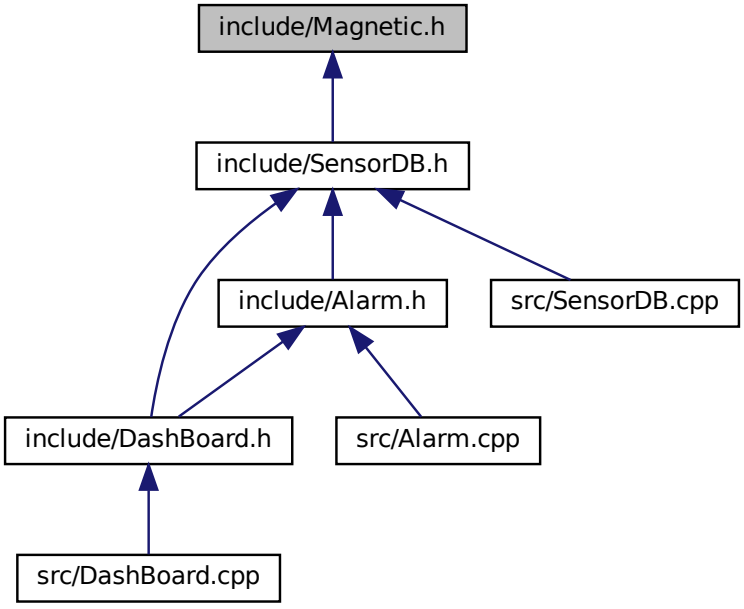
This file contains the declaration of the Magnetic class.

```
#include "Sensor.h"
```

Include dependency graph for Magnetic.h:



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



Classes

- class Magnetic

  Represents a magnetic Sensor.

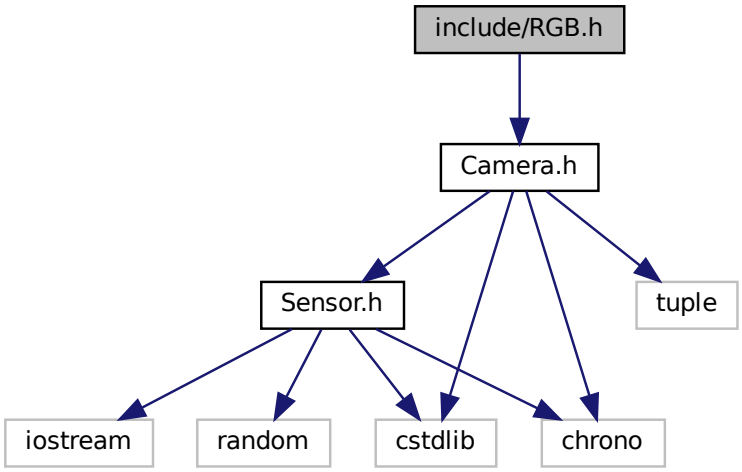
5.10.1 Detailed Description

This file contains the declaration of the Magnetic class.

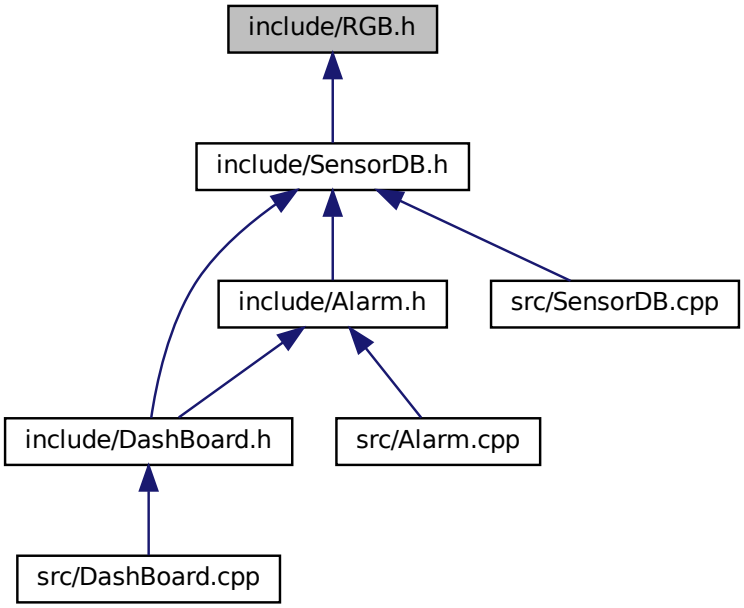
5.11 include/RGB.h File Reference

```
#include "Camera.h"
```

Include dependency graph for RGB.h:



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



Classes

- class Rgb

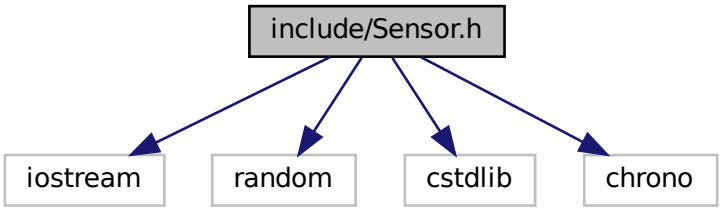
  Represents an RGB Camera Sensor.

5.12 include/Sensor.h File Reference

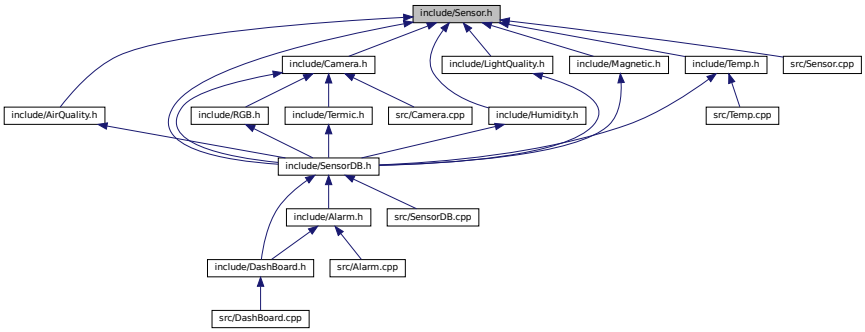
This file contains the declaration of the Sensor class.

```
#include "iostream"
#include <random>
#include <cstdlib>
#include <chrono>
```

Include dependency graph for Sensor.h:



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



Classes

- class Sensor

  Represents a generic sensor.

Enumerations

- enum SensorTypes {
  SAMPLES_DAY = 5 , SAMPLES_WEEK = 8 , TYPE_TEMP = 1 , TYPE_HUM = 2 ,
  TYPE_LIGHT = 3 , TYPE_AIR = 4 , TYPE_RGB = 5 , TYPE_TERMIC = 6 ,
  TYPE_MAG = 7 , TYPE_CAM = 8 }

  Enumerates different types of sensors.

5.12.1 Detailed Description

This file contains the declaration of the Sensor class.

5.12.2 Enumeration Type Documentation

5.12.2.1 SensorTypes

```
enum SensorTypes
```

Enumerates different types of sensors.

Enumerator

SAMPLES_DAY	Number of samples per day.
SAMPLES_WEEK	Number of samples per week.
TYPE_TEMP	Temperature sensor type.
TYPE_HUM	Humidity sensor type.
TYPE_LIGHT	Light sensor type.
TYPE_AIR	Air quality sensor type.
TYPE_RGB	RGB sensor type.
TYPE_TERMIC	Termic sensor type.
TYPE_MAG	Magnetic sensor type.
TYPE_CAM	Camera sensor type.

Definition at line 18 of file Sensor.h.

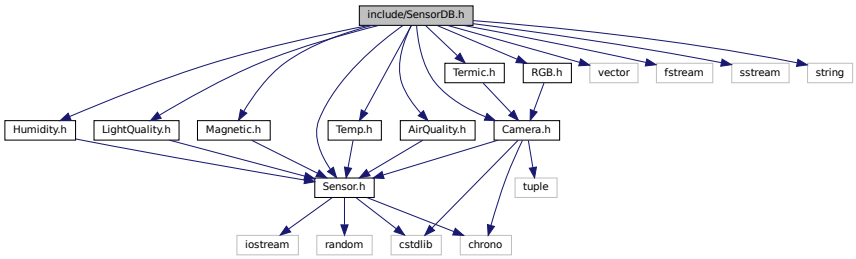
```
18                  {
19   SAMPLES_DAY = 5,
20   SAMPLES_WEEK = 8,
21   TYPE_TEMP = 1,
22   TYPE_HUM = 2,
23   TYPE_LIGHT = 3,
24   TYPE_AIR = 4,
25   TYPE_RGB = 5,
26   TYPE_TERMIC = 6,
27   TYPE_MAG = 7,
28   TYPE_CAM = 8,
29 };
```

5.13 include/SensorDB.h File Reference

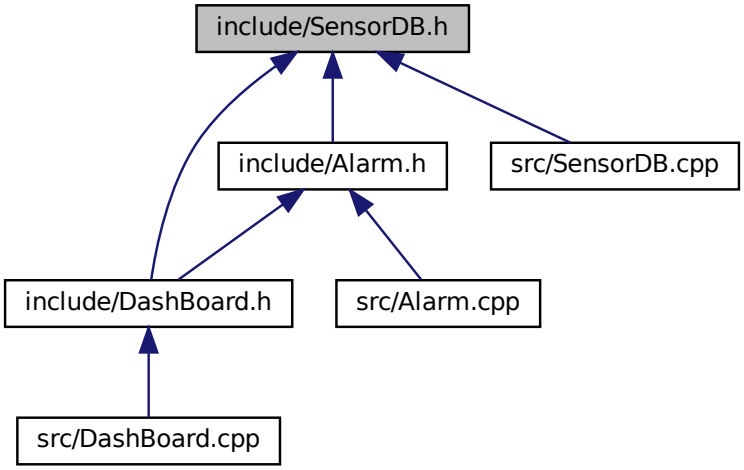
This file contains the declaration of the SensorDB class.

```
#include "Sensor.h"
#include "Temp.h"
#include "AirQuality.h"
#include "Humidity.h"
#include "LightQuality.h"
#include "Magnetic.h"
#include "Camera.h"
#include "Termic.h"
#include "RGB.h"
#include <vector>
#include <fstream>
#include <sstream>
```

`#include <string>`
Include dependency graph for SensorDB.h:



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



Classes

- class SensorDB

  Represents a database of sensors.

5.13.1 Detailed Description

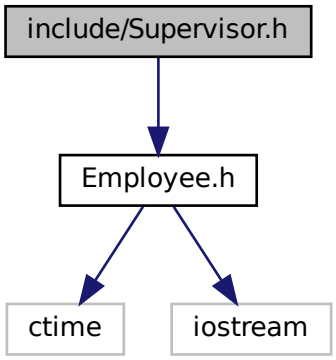
This file contains the declaration of the SensorDB class.

5.14 include/Supervisor.h File Reference

This file contains the declaration of the Supervisor class.

```
#include "Employee.h"
```

Include dependency graph for Supervisor.h:



Classes

- class Supervisor

  Represents a Supervisor, which is a type of Employee.

5.14.1 Detailed Description

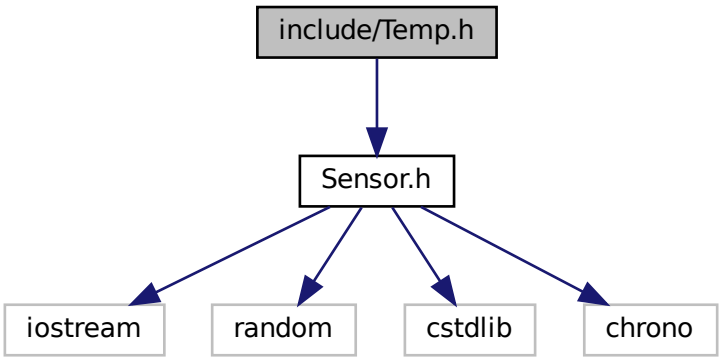
This file contains the declaration of the Supervisor class.

5.15 include/Temp.h File Reference

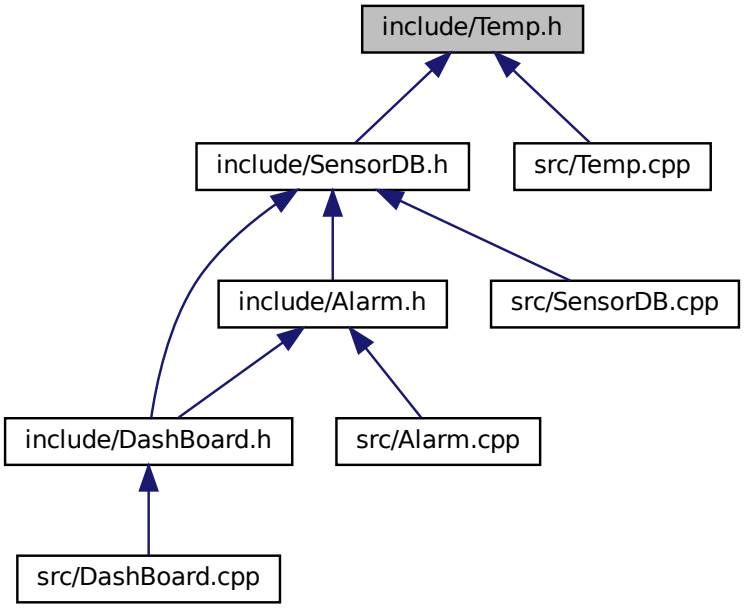
This file contains the declaration of the Temp class.

```
#include "Sensor.h"
```

Include dependency graph for Temp.h:



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



Classes

- class Temp

  Represents a temperature Sensor.

5.15.1 Detailed Description

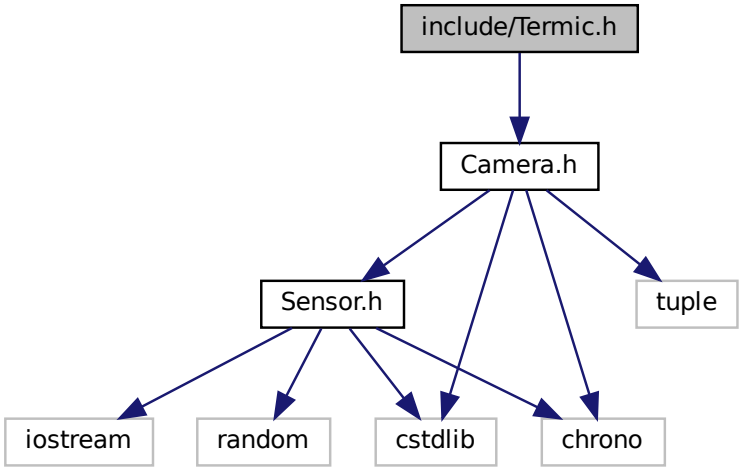
This file contains the declaration of the Temp class.

5.16 include/Termic.h File Reference

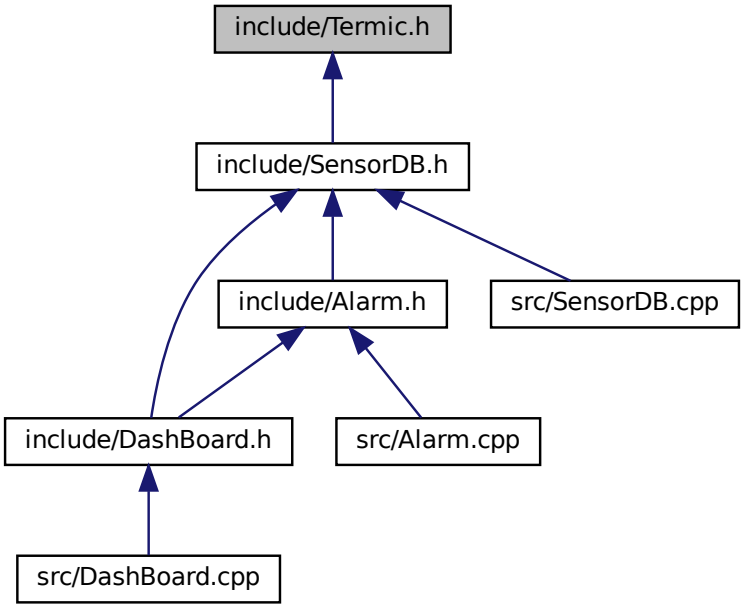
This file contains the declaration of the Termic class.

```
#include "Camera.h"
```

Include dependency graph for Termic.h:



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



Classes

- class Termic

  Represents a thermal Camera Sensor.

5.16.1 Detailed Description

This file contains the declaration of the Termic class.

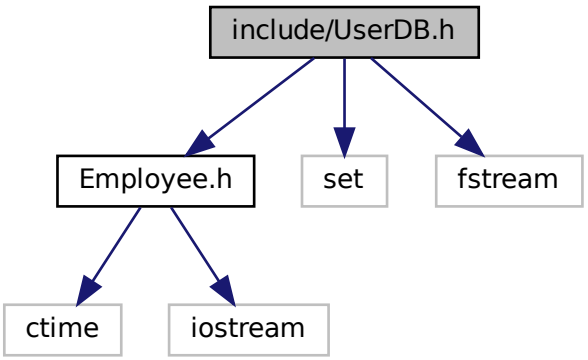
5.17 include/UserDB.h File Reference

This file contains the declaration of the UserDB class.

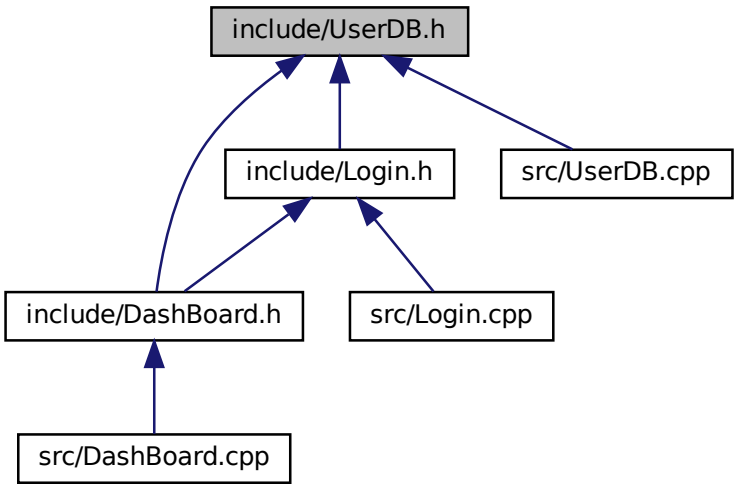
```
#include "Employee.h"
#include <set>
```

```
#include <fstream>
```

Include dependency graph for UserDB.h:



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



Classes

- struct EmployeeNifCompare

  For comparing Employees based on their NIF and ID.
- class UserDB

  Represents a User's database.

5.17.1 Detailed Description

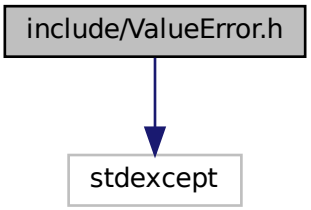
This file contains the declaration of the UserDB class.

5.18 include/ValueError.h File Reference

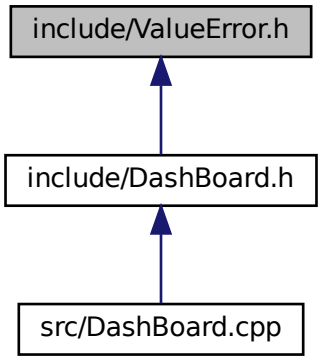
This file contains the declaration of the ValueError class.

```
#include <stdexcept>
```

Include dependency graph for ValueError.h:



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



Classes

- class ValueError

  Represents a value error exception.

5.18.1 Detailed Description

This file contains the declaration of the ValueError class.

5.19 include/Worker.h File Reference

This file contains the declaration of the Worker class.

Classes

- class Worker

  Represents a Worker, which is a type of Employee.

5.19.1 Detailed Description

This file contains the declaration of the Worker class.

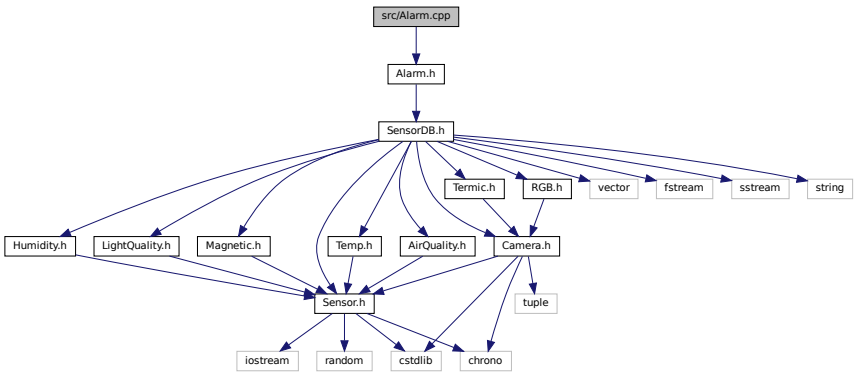
5.20 src/Admin.cpp File Reference

5.21 src/AirQuality.cpp File Reference

5.22 src/Alarm.cpp File Reference

```
#include "Alarm.h"
```

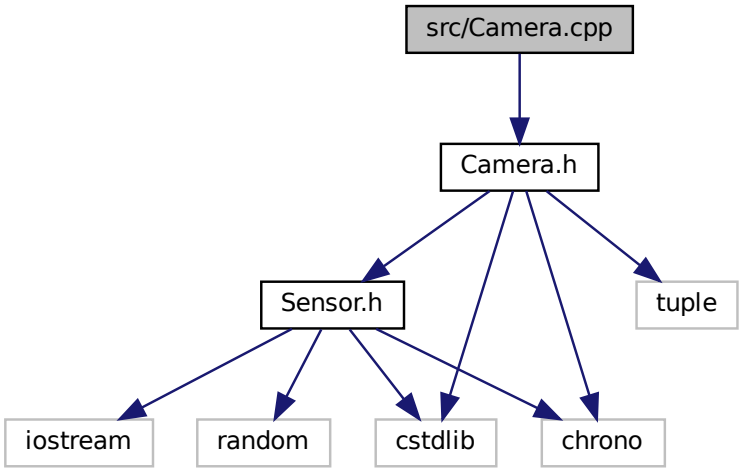
Include dependency graph for Alarm.cpp:



5.23 src/Camera.cpp File Reference

```
#include "Camera.h"
```

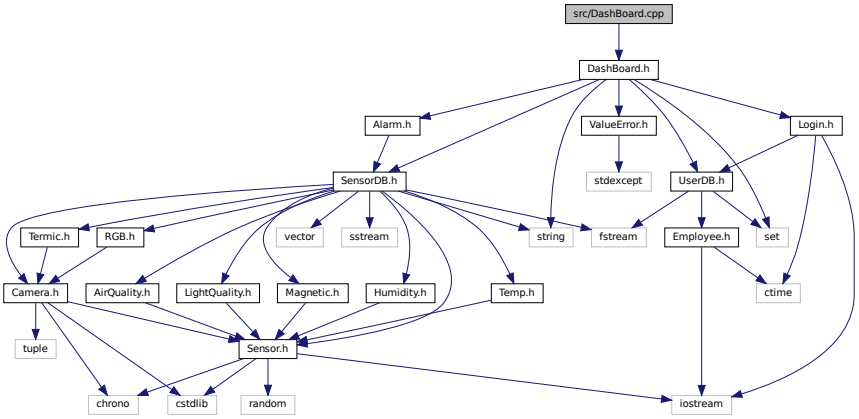
Include dependency graph for Camera.cpp:



5.24 src/DashBoard.cpp File Reference

```
#include "DashBoard.h"
```

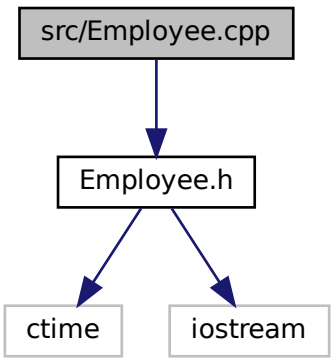
Include dependency graph for DashBoard.cpp:



5.25 src/Employee.cpp File Reference

```
#include "Employee.h"
```

Include dependency graph for Employee.cpp:



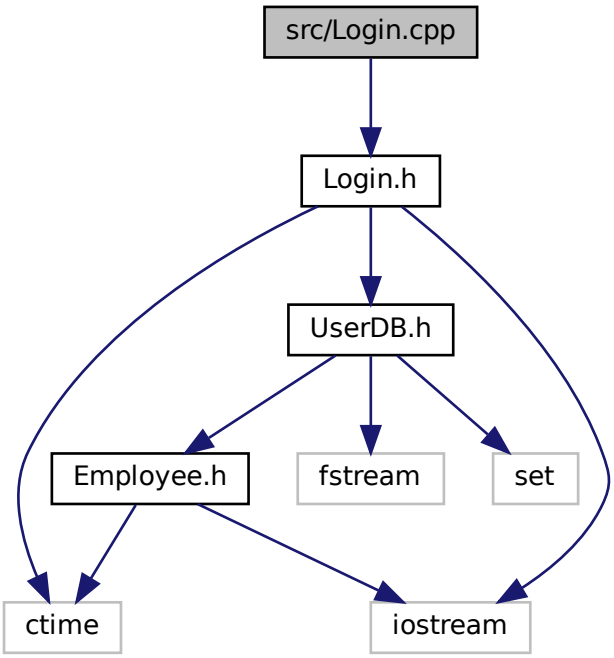
5.26 src/Humidity.cpp File Reference

5.27 src/LightQuality.cpp File Reference

5.28 src/Login.cpp File Reference

```
#include "Login.h"
```

Include dependency graph for Login.cpp:

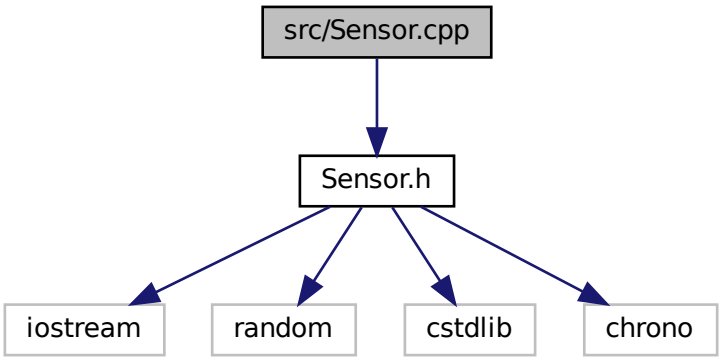


5.29 src/Magnetic.cpp File Reference

5.30 src/RGB.cpp File Reference

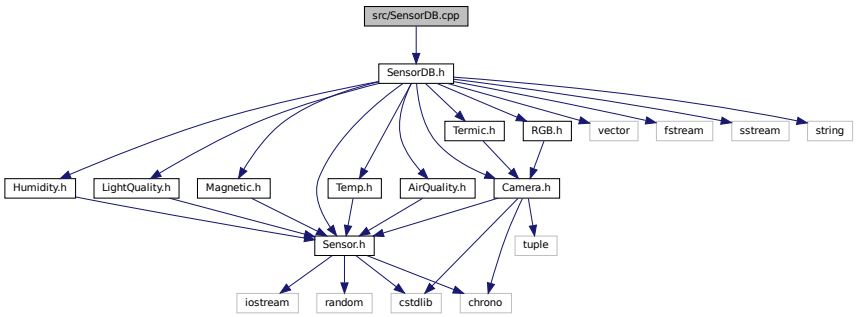
5.31 src/Sensor.cpp File Reference

`#include "Sensor.h"`
Include dependency graph for Sensor.cpp:



5.32 src/SensorDB.cpp File Reference

`#include "SensorDB.h"`
Include dependency graph for SensorDB.cpp:

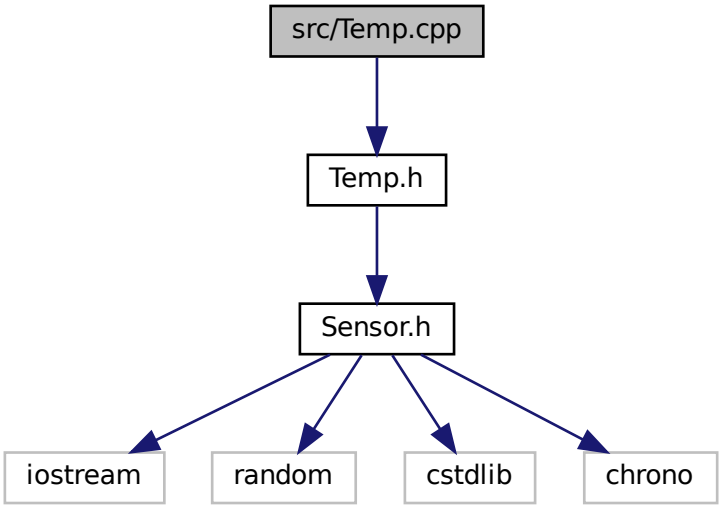


5.33 src/Supervisor.cpp File Reference

5.34 src/Temp.cpp File Reference

```
#include "Temp.h"
```

Include dependency graph for Temp.cpp:

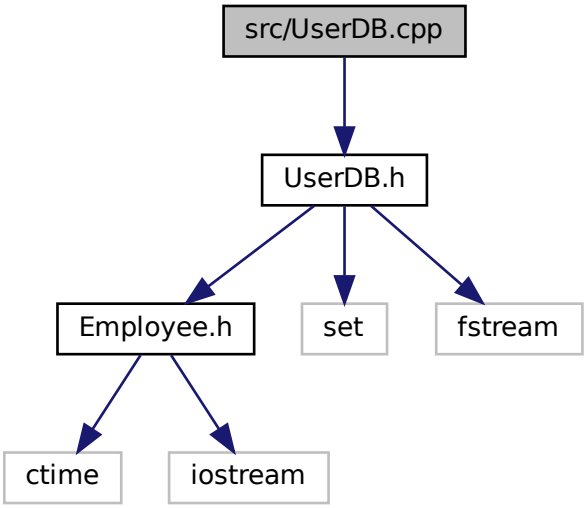


5.35 src/Termic.cpp File Reference

5.36 src/UserDB.cpp File Reference

```
#include "UserDB.h"
```

Include dependency graph for UserDB.cpp:



5.37 src/ValueError.cpp File Reference

5.38 src/Worker.cpp File Reference