CAR C++ PROJECT

AUTHOR : Abdalla Mohamed

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

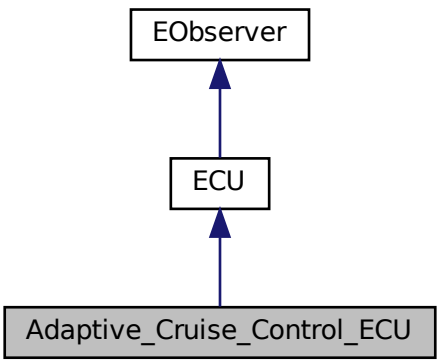
Class Documentation

3.1 Adaptive_Cruise_Control_ECU Class Reference

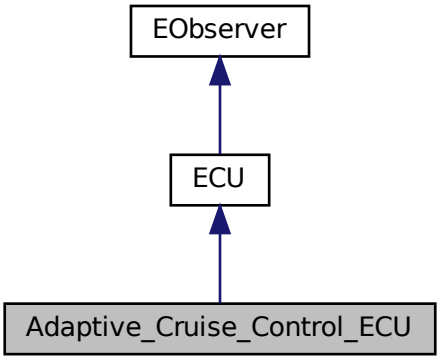
Class representing an Adaptive Cruise Control ECU.

`#include <Adaptive_Cruise_Control_ECU.hpp>`

Inheritance diagram for Adaptive_Cruise_Control_ECU:



Collaboration diagram for Adaptive_Cruise_Control_ECU:



Public Member Functions

- Adaptive_Cruise_Control_ECU ()

  Default constructor for Adaptive_Cruise_Control_ECU.
- void AttachSensor (std::shared_ptr< Sensor > s) override

  Attaches a sensor to the adaptive cruise control ECU.
- void DeattachSensor (std::shared_ptr< Sensor > s) override

  Detaches a sensor from the adaptive cruise control ECU.
- std::string getName () const override

  Gets the name of the adaptive cruise control ECU.
- int getID () const override

  Gets the ID of the adaptive cruise control ECU.
- void PerformFunction (Car c) override

  Performs the function of the adaptive cruise control ECU.
- ~Adaptive_Cruise_Control_ECU ()

  Destructor for Adaptive_Cruise_Control_ECU.
- bool IsON ()

  Checks if the adaptive cruise control is currently active.

Additional Inherited Members

3.1.1 Detailed Description

Class representing an Adaptive Cruise Control ECU.

This class inherits from the ECU base class and implements methods to attach and detach sensors, perform functions related to adaptive cruise control, and manage the state of the adaptive cruise control system.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Adaptive_Cruise_Control_ECU()

```
Adaptive_Cruise_Control_ECU::Adaptive_Cruise_Control_ECU ( )
```

Default constructor for Adaptive_Cruise_Control_ECU.

Constructs an Adaptive_Cruise_Control_ECU object.

Initializes the ECU and sets the adaptive cruise control status.

Initializes the name and type of the ECU and sets the adaptive cruise control status to off.

3.1.2.2 ~Adaptive_Cruise_Control_ECU()

```
Adaptive_Cruise_Control_ECU::~Adaptive_Cruise_Control_ECU ( )
```

Destructor for Adaptive_Cruise_Control_ECU.

Cleans up resources and performs any necessary shutdown procedures.

Logs a message indicating the destruction of the ECU.

3.1.3 Member Function Documentation

3.1.3.1 AttachSensor()

```
void Adaptive_Cruise_Control_ECU::AttachSensor (
            std::shared_ptr< Sensor > s )  [override], [virtual]
```

Attaches a sensor to the adaptive cruise control ECU.

Parameters

s	A shared pointer to the sensor to attach.

Checks if the sensor is already subscribed. If not, it adds the sensor to the list of subscribed sensors and logs the action.

Parameters

s	A shared pointer to the sensor to attach.

Implements ECU.

3.1.3.2 DeattachSensor()

```
void Adaptive_Cruise_Control_ECU::DeattachSensor (
            std::shared_ptr< Sensor > s )  [override], [virtual]
```

Detaches a sensor from the adaptive cruise control ECU.

Parameters

s	A shared pointer to the sensor to detach.

Searches for the sensor in the list of subscribed sensors and removes it if found. Logs the action.

Parameters

s	A shared pointer to the sensor to detach.

Implements ECU.

3.1.3.3 getID()

```
int Adaptive_Cruise_Control_ECU::getID ( ) const  [override], [virtual]
```

Gets the ID of the adaptive cruise control ECU.

Returns

The ID of the ECU as an integer.

Implements ECU.

3.1.3.4 getName()

```
std::string Adaptive_Cruise_Control_ECU::getName ( ) const  [override], [virtual]
```

Gets the name of the adaptive cruise control ECU.

Returns

The name of the ECU as a string.

Implements ECU.

3.1.3.5 IsON()

```
bool Adaptive_Cruise_Control_ECU::IsON ( )
```

Checks if the adaptive cruise control is currently active.

Returns

True if the adaptive cruise control is on, false otherwise.

3.1.3.6 PerformFunction()

```
void Adaptive_Cruise_Control_ECU::PerformFunction (
            Car c )  [override], [virtual]
```

Performs the function of the adaptive cruise control ECU.

Activates the adaptive cruise control functionality.

Parameters

c	The car object that the ECU is controlling.

Logs the activation of the adaptive cruise control mode and sets the status to on.

Parameters

c	The car object that the ECU is controlling.

Implements ECU.

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

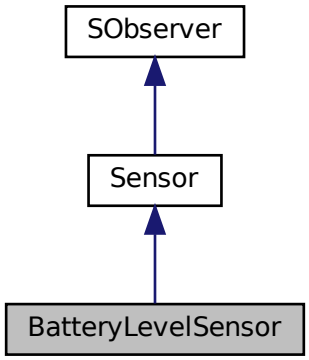
- Sensors/Adaptive_Cruise_Control_ECU.hpp
- Sensors/Adaptive_Cruise_Control_ECU.cpp

3.2 BatteryLevelSensor Class Reference

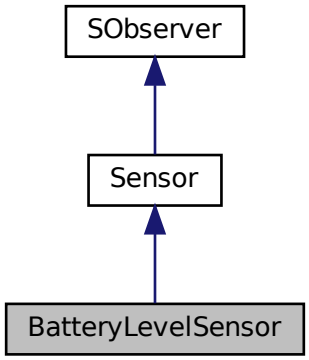
Represents a battery level sensor in a car system.

```
#include <BatteryLevelSensor.hpp>
```

Inheritance diagram for BatteryLevelSensor:



Collaboration diagram for BatteryLevelSensor:



Public Member Functions

- BatteryLevelSensor ()

  Constructs a BatteryLevelSensor object.
- ~BatteryLevelSensor ()

  Destroys the BatteryLevelSensor object.
- int getSensorID () const override

  Gets the unique sensor ID.
- double GetSensorData () override

  Retrieves the current battery level sensor data.
- void sensorRead () override

Reads the sensor data.

- int getSensorCount () const

  Gets the total count of battery level sensors.

- void PrintInfo () override

  Prints information about the battery level sensor.

- std::string getType () override

  Gets the type of the sensor.

- **BatteryLevelSensor** (const BatteryLevelSensor &)=delete
- BatteryLevelSensor & **operator=** (const BatteryLevelSensor &)=delete
- **BatteryLevelSensor** (BatteryLevelSensor &&)=delete
- BatteryLevelSensor & **operator=** (BatteryLevelSensor &&)=delete
- void AttachECU (std::weak_ptr< ECU > E) override

  Attaches an ECU to this sensor.

- void DeAttachECU (std::weak_ptr< ECU > E) override

  Detaches an ECU from this sensor.

- void updateECU (std::weak_ptr< ECU > E) override

  Updates the attached ECU.

- void NotifyAllECUs () override

  Notifies all attached ECUs of changes.

- virtual int getTotalSensorsCount () override

  Gets the total number of sensors of this type.

Additional Inherited Members

3.2.1 Detailed Description

Represents a battery level sensor in a car system.

The BatteryLevelSensor class inherits from the Sensor base class and provides functionality to retrieve and manage battery level data.

3.2.2 Member Function Documentation

3.2.2.1 AttachECU()

```
void BatteryLevelSensor::AttachECU (
            std::weak_ptr< ECU > Ecu )  [override], [virtual]
```

Attaches an ECU to this sensor.

Parameters

E	A weak pointer to the ECU to attach.
Ecu	A weak pointer to the ECU to attach.

Implements Sensor.

3.2.2.2 DeAttachECU()

```
void BatteryLevelSensor::DeAttachECU (
            std::weak_ptr< ECU > Ecu )  [override], [virtual]
```

Detaches an ECU from this sensor.

Parameters

E	A weak pointer to the ECU to detach.
Ecu	A weak pointer to the ECU to detach.

Implements Sensor.

3.2.2.3 getSensorCount()

```
int BatteryLevelSensor::getSensorCount ( ) const
```

Gets the total count of battery level sensors.

Gets the count of battery level sensors created.

Returns

The count of battery level sensors.

3.2.2.4 GetSensorData()

```
double BatteryLevelSensor::GetSensorData ( )  [override], [virtual]
```

Retrieves the current battery level sensor data.

Returns

The battery level value.

The current battery level value.

Implements Sensor.

3.2.2.5 getSensorID()

```
int BatteryLevelSensor::getSensorID ( ) const  [override], [virtual]
```

Gets the unique sensor ID.

Returns

The sensor ID.

Implements Sensor.

3.2.2.6 getTotalSensorsCount()

```
int BatteryLevelSensor::getTotalSensorsCount ( )  [override], [virtual]
```

Gets the total number of sensors of this type.

Gets the total count of all sensors.

Returns

The total count of battery level sensors.

The total count of sensors.

Implements Sensor.

3.2.2.7 getType()

```
std::string BatteryLevelSensor::getType ( )  [override], [virtual]
```

Gets the type of the sensor.

Gets the type of the battery level sensor.

Returns

A string representing the sensor type.

Implements Sensor.

3.2.2.8 NotifyAllECUs()

```
void BatteryLevelSensor::NotifyAllECUs ( )  [override], [virtual]
```

Notifies all attached ECUs of changes.

Notifies all attached ECUs of the latest sensor data.

Implements Sensor.

3.2.2.9 sensorRead()

```
void BatteryLevelSensor::sensorRead ( )  [override], [virtual]
```

Reads the sensor data.

Reads the sensor data and updates the battery level.

Implements Sensor.

3.2.2.10 updateECU()

```
void BatteryLevelSensor::updateECU (
            std::weak_ptr< ECU > E )  [override], [virtual]
```

Updates the attached ECU.

Updates the attached ECU with the latest battery level data.

Parameters

E	A weak pointer to the ECU to update.

Implements Sensor.

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

- Sensors/BatteryLevelSensor.hpp
- Sensors/BatteryLevelSensor.cpp

3.3 Car Class Reference

Represents a car with various sensors and ECUs (Electronic Control Units).

```
#include <Car.hpp>
```

Public Member Functions

- Car (const std::string &model, const std::string &make)

  Constructs a Car object with the specified model and make.
- ~Car ()

  Destroys the Car object and releases resources.
- void ActivateECU (std::shared_ptr< ECU > E)

  Activates the specified ECU.
- void ActivateSensor (std::shared_ptr< Sensor > S)

  Activates the specified sensor.
- void StartDiagonisticTool () const

  Starts the diagnostic tool for the car.
- void CarINIT ()

  Initializes the car systems.
- void setAdaptiveMode (bool mode)

  Sets the adaptive cruise control mode.
- bool getAdaptiveMode ()

  Retrieves the current state of the adaptive cruise control mode.
- void UpdateSensorsData ()

  Updates the data from all sensors in the car.
- void DisplayStatus ()

  Displays the current status of the car, including sensor data.

3.3.1 Detailed Description

Represents a car with various sensors and ECUs (Electronic Control Units).

The Car class encapsulates the functionality of a vehicle, including activating ECUs and sensors, updating sensor data, and displaying the car's status.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Car()

```
Car::Car (
            const std::string & model,
            const std::string & make )
```

Constructs a Car object with the specified model and make.

Parameters

model	The model of the car.
make	The make of the car.

3.3.3 Member Function Documentation

3.3.3.1 ActivateECU()

```
void Car::ActivateECU (
            std::shared_ptr< ECU > E )
```

Activates the specified ECU.

Parameters

E	A shared pointer to the ECU to be activated.

Activates a new ECU and adds it to the car's ECU list.

Parameters

E	A shared pointer to the ECU to be activated.

3.3.3.2 ActivateSensor()

```
void Car::ActivateSensor (
            std::shared_ptr< Sensor > S )
```

Activates the specified sensor.

Parameters

S	A shared pointer to the sensor to be activated.

Activates a new sensor and adds it to the car's sensor list.

Parameters

S	A shared pointer to the sensor to be activated.

3.3.3.3 DisplayStatus()

```
void Car::DisplayStatus ( )
```

Displays the current status of the car, including sensor data.

Displays the current status of the car, including speed, temperature, battery level, radar status, and adaptive mode.

3.3.3.4 getAdaptiveMode()

```
bool Car::getAdaptiveMode ( )
```

Retrieves the current state of the adaptive cruise control mode.

Returns

bool True if adaptive mode is active, false otherwise.

Retrieves the current state of the adaptive cruise control mode.

Returns

bool True if adaptive mode is active, false otherwise.

3.3.3.5 setAdaptiveMode()

```
void Car::setAdaptiveMode (
            bool mode )
```

Sets the adaptive cruise control mode.

Parameters

mode	A boolean indicating the desired adaptive mode state.

Sets the adaptive cruise control mode and triggers its function if enabled.

Parameters

mode	A boolean indicating whether to enable or disable the adaptive mode.

3.3.3.6 StartDiagonisticTool()

```
void Car::StartDiagonisticTool ( ) const
```

Starts the diagnostic tool for the car.

Starts the diagnostic tool for the car, attaching all sensors to the diagnostic ECU.

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

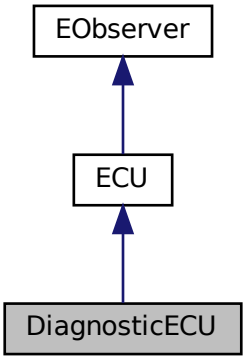
- Sensors/Car.hpp
- Sensors/Car.cpp

3.4 DiagnosticECU Class Reference

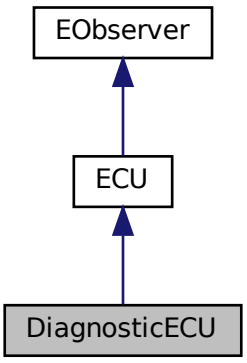
Class representing a Diagnostic ECU, inheriting from the ECU base class.

`#include <DiagnosticsECU.hpp>`

Inheritance diagram for DiagnosticECU:



Collaboration diagram for DiagnosticECU:



Public Member Functions

- DiagnosticECU ()

  Constructor for the DiagnosticECU class.
- void AttachSensor (std::shared_ptr< Sensor > s) override

Attaches a sensor to this Diagnostic ECU.

- void DeattachSensor (std::shared_ptr< Sensor > s) override

  Detaches a sensor from this Diagnostic ECU.
- std::string getName () const override

  Retrieves the name of this Diagnostic ECU.
- int getID () const override

  Retrieves the ID of this Diagnostic ECU.
- void PerformFunction (Car c) override

  Performs the main function of the Diagnostic ECU with the specified car.
- void update ()

  Updates the state of the Diagnostic ECU.
- ~DiagnosticECU ()

  Destructor for the DiagnosticECU class.
- bool IsON ()

  Checks if the Diagnostic ECU is turned ON.

Additional Inherited Members

3.4.1 Detailed Description

Class representing a Diagnostic ECU, inheriting from the ECU base class.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 DiagnosticECU()

```
DiagnosticECU::DiagnosticECU ( )
```

Constructor for the DiagnosticECU class.

Initializes a new instance of the DiagnosticECU.

Initializes the Diagnostic ECU with its type and state.

3.4.2.2 ~DiagnosticECU()

```
DiagnosticECU::~DiagnosticECU ( )
```

Destructor for the DiagnosticECU class.

Cleans up resources used by the DiagnosticECU.

Logs the destruction of the Diagnostic ECU.

3.4.3 Member Function Documentation

3.4.3.1 AttachSensor()

```
void DiagnosticECU::AttachSensor (
            std::shared_ptr< Sensor > s )  [override], [virtual]
```

Attaches a sensor to this Diagnostic ECU.

Attaches a sensor to the Diagnostic ECU.

Parameters

s	A shared pointer to the sensor to be attached.

Checks if the sensor is already subscribed. If not, it subscribes the sensor and logs the action.

Parameters

s	A shared pointer to the sensor to be attached.

Implements ECU.

3.4.3.2 DeattachSensor()

```
void DiagnosticECU::DeattachSensor (
            std::shared_ptr< Sensor > s )  [override], [virtual]
```

Detaches a sensor from this Diagnostic ECU.

Detaches a sensor from the Diagnostic ECU.

Parameters

s	A shared pointer to the sensor to be detached.

Removes the specified sensor from the list of subscribed sensors and logs the action.

Parameters

s	A shared pointer to the sensor to be detached.

Implements ECU.

3.4.3.3 getID()

```
int DiagnosticECU::getID ( ) const  [override], [virtual]
```

Retrieves the ID of this Diagnostic ECU.

Retrieves the ID of the Diagnostic ECU.

Returns

int The ID of the Diagnostic ECU.

Implements ECU.

3.4.3.4 getName()

```
std::string DiagnosticECU::getName ( ) const  [override], [virtual]
```

Retrieves the name of this Diagnostic ECU.

Retrieves the name of the Diagnostic ECU.

Returns

std::string The name of the Diagnostic ECU.

Implements ECU.

3.4.3.5 IsON()

```
bool DiagnosticECU::IsON ( )
```

Checks if the Diagnostic ECU is turned ON.

Returns

true if the ECU is ON, false otherwise.

3.4.3.6 PerformFunction()

```
void DiagnosticECU::PerformFunction (
            Car c )  [override], [virtual]
```

Performs the main function of the Diagnostic ECU with the specified car.

Performs the primary function of the Diagnostic ECU.

Parameters

c	The car to perform the function on.

Activates diagnostic mode, updates the sensors, and logs the current data.

Parameters

c	The car instance to perform the function on.

Implements ECU.

3.4.3.7 update()

```
void DiagnosticECU::update ( )
```

Updates the state of the Diagnostic ECU.

Updates the state of the Diagnostic ECU by notifying all subscribed sensors.

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

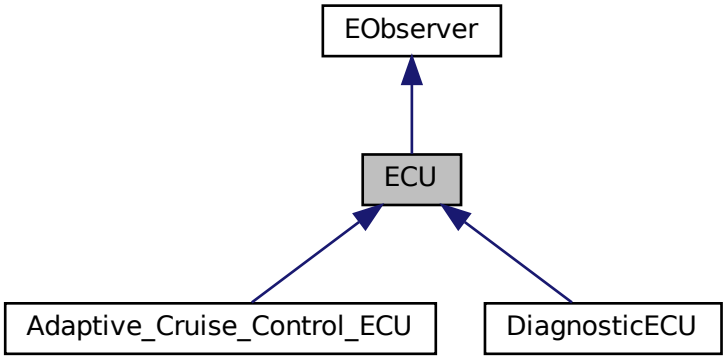
- Sensors/DiagnosticsECU.hpp
- Sensors/DiagnosticsECU.cpp

3.5 ECU Class Reference

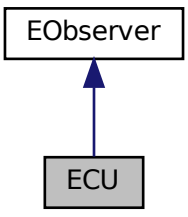
Abstract class representing an Electronic Control Unit (ECU).

```
#include <ECU.hpp>
```

Inheritance diagram for ECU:



Collaboration diagram for ECU:



Public Member Functions

- ECU ()

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

  Destructor for the ECU class.
- virtual int getID () const =0

  Get the unique identifier for the ECU.
- virtual std::string getName () const =0

  Get the name of the ECU.
- virtual void PerformFunction (Car c)=0

  Perform the specific function of the ECU based on a given car state.
- **ECU** (const ECU &)=delete
- ECU & **operator=** (const ECU &)=delete
- **ECU** (ECU &&)=delete
- ECU & **operator=** (ECU &&)=delete
- virtual void AttachSensor (std::shared_ptr< Sensor > s)=0

  Attach a sensor to the ECU for data updates.
- virtual void DeattachSensor (std::shared_ptr< Sensor > s)=0

  Detach a sensor from the ECU.

Static Public Member Functions

- static int getECUCount ()

  Get the current count of ECUs created.

Public Attributes

- std::vector< std::unordered_map< int, double > > **Recent_Sensory_Data**

Protected Attributes

- int ECU_ID
- std::string name
- std::vector< std::shared_ptr< Sensor > > Subscribed_Sensors

Static Protected Attributes

- static std::atomic< int > ECU_Count {0}

3.5.1 Detailed Description

Abstract class representing an Electronic Control Unit (ECU).

The ECU class is responsible for interfacing with various sensors and performing specific functions based on the data received from those sensors.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 ECU()

```
ECU::ECU ( )
```

Constructor for the ECU class.

Initializes the ECU object, increments the count of ECUs, and assigns a unique ID to the ECU.

3.5.2.2 ~ECU()

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

Destructor for the ECU class.

Decreases the count of ECUs when an ECU object is destroyed and logs the remaining count of ECUs.

3.5.3 Member Function Documentation

3.5.3.1 AttachSensor()

```
virtual void ECU::AttachSensor (
            std::shared_ptr< Sensor > s )  [pure virtual]
```

Attach a sensor to the ECU for data updates.

Parameters

s	A shared pointer to the sensor to be attached.

Implemented in Adaptive_Cruise_Control_ECU, and DiagnosticECU.

3.5.3.2 DeattachSensor()

```
virtual void ECU::DeattachSensor (
            std::shared_ptr< Sensor > s )  [pure virtual]
```

Detach a sensor from the ECU.

Parameters

s	A shared pointer to the sensor to be detached.

Implemented in Adaptive_Cruise_Control_ECU, and DiagnosticECU.

3.5.3.3 getECUCount()

```
int ECU::getECUCount ( )  [static]
```

Get the current count of ECUs created.

Returns

int The count of ECUs.

3.5.3.4 getID()

```
virtual int ECU::getID ( ) const  [pure virtual]
```

Get the unique identifier for the ECU.

Returns

int Unique ID of the ECU.

Implemented in DiagnosticECU, and Adaptive_Cruise_Control_ECU.

3.5.3.5 getName()

```
virtual std::string ECU::getName ( ) const  [pure virtual]
```

Get the name of the ECU.

Returns

std::string The name of the ECU.

Implemented in DiagnosticECU, and Adaptive_Cruise_Control_ECU.

3.5.3.6 PerformFunction()

```
virtual void ECU::PerformFunction (
            Car c )  [pure virtual]
```

Perform the specific function of the ECU based on a given car state.

Parameters

c	The current state of the car.

Implemented in DiagnosticECU, and Adaptive_Cruise_Control_ECU.

3.5.4 Member Data Documentation

3.5.4.1 ECU_Count

```
std::atomic< int > ECU::ECU_Count {0}  [static], [protected]
```

Static variable to keep track of the number of ECUs created.

3.5.4.2 ECU_ID

```
int ECU::ECU_ID  [protected]
```

Unique identifier for the ECU.

3.5.4.3 name

```
std::string ECU::name  [protected]
```

Name of the ECU.

3.5.4.4 Subscribed_Sensors

```
std::vector<std::shared_ptr<Sensor> > ECU::Subscribed_Sensors  [protected]
```

List of subscribed sensors.

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

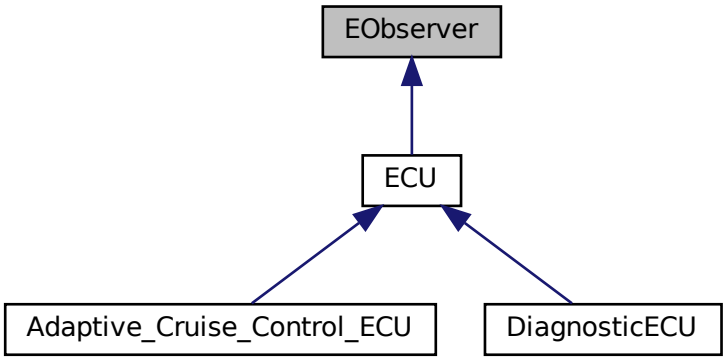
- Sensors/ECU.hpp
- Sensors/ECU.cpp

3.6 EObserver Class Reference

Abstract observer class for ECU.

`#include <ECU.hpp>`

Inheritance diagram for EObserver:



3.6.1 Detailed Description

Abstract observer class for ECU.

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

- Sensors/ECU.hpp

3.7 Logger Class Reference

Logger class for logging messages in a thread-safe manner.

`#include <CarLogger.hpp>`

Public Member Functions

- Logger (const Logger &)=delete

  Deleted copy constructor.
- Logger & operator= (const Logger &)=delete

  Deleted assignment operator.
- void log (const std::string &message)

  Logs a message to the output.

Static Public Member Functions

- static Logger & getInstance ()

  Gets the singleton instance of the Logger.

3.7.1 Detailed Description

Logger class for logging messages in a thread-safe manner.

This class implements the Singleton pattern to ensure only one instance exists throughout the application. It provides a method to log messages with thread safety using mutexes.

3.7.2 Member Function Documentation

3.7.2.1 getInstance()

```
Logger & Logger::getInstance ( )  [static]
```

Gets the singleton instance of the Logger.

Retrieves the singleton instance of the Logger.

This method creates the Logger instance if it doesn't exist and returns the single instance of the Logger.

Returns

Logger& Reference to the singleton Logger instance.

This method ensures that the Logger instance is created only once and returns a reference to that instance. The instance is created on the first call to this method and destroyed when the program exits.

Returns

Logger& Reference to the singleton Logger instance.

3.7.2.2 log()

```
void Logger::log (
            const std::string & message )
```

Logs a message to the output.

Logs a message to the console.

This method logs the specified message in a thread-safe manner.

Parameters

message	The message to log.

This method logs the specified message along with the message number in a thread-safe manner using a mutex lock. It increments the message count each time a new message is logged.

Parameters

message	The message to log.

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

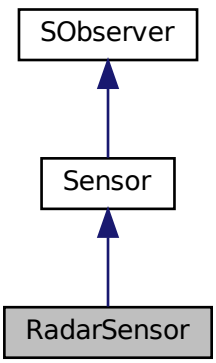
- Sensors/CarLogger.hpp
- Sensors/CarLogger.cpp

3.8 RadarSensor Class Reference

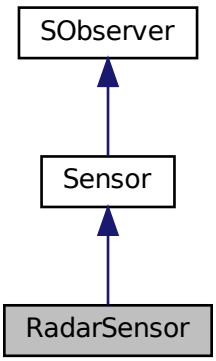
Represents a radar sensor that inherits from the Sensor class.

```
#include <RadarSensor.hpp>
```

Inheritance diagram for RadarSensor:



Collaboration diagram for RadarSensor:



Public Member Functions

- RadarSensor ()

  Default constructor for the RadarSensor class.
- ~RadarSensor ()

  Destructor for the RadarSensor class.
- int getSensorID () const

  Gets the unique identifier for the sensor.
- double GetSensorData () override

  Gets the sensor data by generating a new radar value.
- void sensorRead () override

  Reads the sensor data by generating a random value.
- int getSensorCount () const

  Gets the count of Radar sensors.
- void PrintInfo () override

  Prints information about the radar sensor.
- std::string getType () override

  Gets the type of the sensor.
- **RadarSensor** (const RadarSensor &)=delete
- RadarSensor & **operator=** (const RadarSensor &)=delete
- **RadarSensor** (RadarSensor &&)=delete
- RadarSensor & **operator=** (RadarSensor &&)=delete
- void AttachECU (std::weak_ptr< ECU > E) override

  Attaches an ECU to the radar sensor.
- void DeAttachECU (std::weak_ptr< ECU > E) override

  Detaches an ECU from the radar sensor.
- void updateECU (std::weak_ptr< ECU > E) override

  Updates the attached ECU with the current sensor data.
- void NotifyAllECUs () override

  Notifies all subscribed ECUs with the current sensor data.
- virtual int getTotalSensorsCount () override

  Gets the total count of radar sensors.

Additional Inherited Members

3.8.1 Detailed Description

Represents a radar sensor that inherits from the Sensor class.

This class is responsible for generating random radar data and notifying subscribed ECUs (Electronic Control Units) with the sensor data.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 RadarSensor()

```
RadarSensor::RadarSensor ( )
```

Default constructor for the RadarSensor class.

Constructs a new RadarSensor and increments the sensor count.

3.8.2.2 ~RadarSensor()

```
RadarSensor::~RadarSensor ( )
```

Destructor for the RadarSensor class.

Destructor for RadarSensor. Decreases the sensor count and logs the destruction of the sensor.

3.8.3 Member Function Documentation

3.8.3.1 AttachECU()

```
void RadarSensor::AttachECU (
            std::weak_ptr< ECU > Ecu )  [override], [virtual]
```

Attaches an ECU to the radar sensor.

Attaches a new ECU to the radar sensor.

Parameters

E	Weak pointer to the ECU to attach.
Ecu	A weak pointer to the ECU to be attached.

Implements Sensor.

3.8.3.2 DeAttachECU()

```
void RadarSensor::DeAttachECU (
            std::weak_ptr< ECU > Ecu )  [override], [virtual]
```

Detaches an ECU from the radar sensor.

Parameters

E	Weak pointer to the ECU to detach.
Ecu	A weak pointer to the ECU to be detached.

Implements Sensor.

3.8.3.3 getSensorCount()

```
int RadarSensor::getSensorCount ( ) const
```

Gets the count of Radar sensors.

Gets the current count of RadarSensor instances.

Returns

The count of Radar sensors.

The number of active RadarSensor instances.

3.8.3.4 GetSensorData()

```
double RadarSensor::GetSensorData ( )  [override], [virtual]
```

Gets the sensor data by generating a new radar value.

Retrieves the current sensor data after reading it.

Returns

The radar sensor data.

The current radar sensor data.

Implements Sensor.

3.8.3.5 getSensorID()

```
int RadarSensor::getSensorID ( ) const  [virtual]
```

Gets the unique identifier for the sensor.

Gets the sensor ID.

Returns

The sensor ID.

The ID of the sensor.

Implements Sensor.

3.8.3.6 getTotalSensorsCount()

```
int RadarSensor::getTotalSensorsCount ( )  [override], [virtual]
```

Gets the total count of radar sensors.

Gets the total number of sensors created.

Returns

The total number of radar sensors.

The total number of sensors.

Implements Sensor.

3.8.3.7 getType()

```
std::string RadarSensor::getType ( )  [override], [virtual]
```

Gets the type of the sensor.

Gets the type of the radar sensor.

Returns

The sensor type as a string.

The type of the sensor as a string.

Implements Sensor.

3.8.3.8 NotifyAllECUs()

```
void RadarSensor::NotifyAllECUs ( )  [override], [virtual]
```

Notifies all subscribed ECUs with the current sensor data.

Notifies all subscribed ECUs with the latest sensor data.

Implements Sensor.

3.8.3.9 PrintInfo()

```
void RadarSensor::PrintInfo ( )  [override], [virtual]
```

Prints information about the radar sensor.

Logs information about the creation of the sensor.

Implements Sensor.

3.8.3.10 sensorRead()

```
void RadarSensor::sensorRead ( )  [override], [virtual]
```

Reads the sensor data by generating a random value.

Reads data from the sensor by generating random values.

Implements Sensor.

3.8.3.11 updateECU()

```
void RadarSensor::updateECU (
            std::weak_ptr< ECU > E )  [override], [virtual]
```

Updates the attached ECU with the current sensor data.

Updates the ECU with the latest sensor data.

Parameters

E	Weak pointer to the ECU to update.
E	A weak pointer to the ECU to be updated.

Implements Sensor.

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

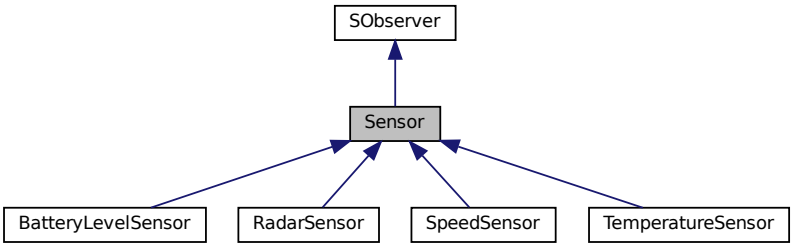
- Sensors/RadarSensor.hpp
- Sensors/RadarSensor.cpp

3.9 Sensor Class Reference

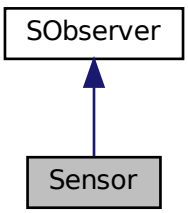
Abstract base class for all sensors.

```
#include <Sensor.hpp>
```

Inheritance diagram for Sensor:



Collaboration diagram for Sensor:



Public Member Functions

- virtual double getRandomData ()=0
  Generate random sensor data.
- virtual void sensorRead ()=0
  Mimic sensor reading by calling getRandomData().
- virtual double GetSensorData ()=0
  Getter for the current sensor data value.
- virtual void PrintInfo ()=0
  Print information about the sensor.
- virtual void AttachECU (std::weak_ptr< ECU > E)=0
  Attach an ECU to the sensor for notifications.
- virtual void DeAttachECU (std::weak_ptr< ECU > E)=0
  Detach an ECU from the sensor.
- virtual void updateECU (std::weak_ptr< ECU > E)=0
  Update the specified ECU with the latest data.
- virtual void NotifyAllECUs ()=0
  Notify all subscribed ECUs of the data update.
- virtual std::string getType ()=0
  Get the type of the sensor.
- virtual int getSensorID () const =0
  Get the unique identifier for the sensor.
- virtual int getTotalSensorsCount ()=0
  Get the total count of sensor instances.

Protected Attributes

- std::vector< std::weak_ptr< ECU > > Subscribed_ECUs

Static Protected Attributes

- static std::atomic< int > total_sensor_count {0}

3.9.1 Detailed Description

Abstract base class for all sensors.

This class implements the observer pattern and serves as a base for different sensor types, defining the common interface that they must adhere to.

3.9.2 Member Function Documentation

3.9.2.1 AttachECU()

```
virtual void Sensor::AttachECU (
            std::weak_ptr< ECU > E )  [pure virtual]
```

Attach an ECU to the sensor for notifications.

Parameters

E	A weak pointer to the ECU to be attached.

Implemented in TemperatureSensor, SpeedSensor, BatteryLevelSensor, and RadarSensor.

3.9.2.2 DeAttachECU()

```
virtual void Sensor::DeAttachECU (
            std::weak_ptr< ECU > E )  [pure virtual]
```

Detach an ECU from the sensor.

Parameters

E	A weak pointer to the ECU to be detached.

Implemented in TemperatureSensor, SpeedSensor, BatteryLevelSensor, and RadarSensor.

3.9.2.3 getRandomData()

```
virtual double Sensor::getRandomData ( )  [pure virtual]
```

Generate random sensor data.

Returns

A double representing the random sensor data.

3.9.2.4 GetSensorData()

```
virtual double Sensor::GetSensorData ( )  [pure virtual]
```

Getter for the current sensor data value.

Returns

A double representing the current sensor data.

Implemented in SpeedSensor, TemperatureSensor, BatteryLevelSensor, and RadarSensor.

3.9.2.5 getSensorID()

```
virtual int Sensor::getSensorID ( ) const  [pure virtual]
```

Get the unique identifier for the sensor.

Returns

An integer representing the sensor ID.

Implemented in TemperatureSensor, BatteryLevelSensor, SpeedSensor, and RadarSensor.

3.9.2.6 getTotalSensorsCount()

```
virtual int Sensor::getTotalSensorsCount ( )  [pure virtual]
```

Get the total count of sensor instances.

Returns

An integer representing the total number of sensors.

Implemented in TemperatureSensor, SpeedSensor, BatteryLevelSensor, and RadarSensor.

3.9.2.7 getType()

```
virtual std::string Sensor::getType ( )  [pure virtual]
```

Get the type of the sensor.

Returns

A string representing the sensor type.

Implemented in SpeedSensor, TemperatureSensor, BatteryLevelSensor, and RadarSensor.

3.9.2.8 updateECU()

```
virtual void Sensor::updateECU (
            std::weak_ptr< ECU > E )  [pure virtual]
```

Update the specified ECU with the latest data.

Parameters

E	A weak pointer to the ECU to be updated.

Implements SObserver.

Implemented in TemperatureSensor, SpeedSensor, BatteryLevelSensor, and RadarSensor.

3.9.3 Member Data Documentation

3.9.3.1 Subscribed_ECUs

```
std::vector<std::weak_ptr<ECU> > Sensor::Subscribed_ECUs  [protected]
```

List of subscribed ECUs

3.9.3.2 total_sensor_count

```
std::atomic< int > Sensor::total_sensor_count {0}  [static], [protected]
```

Atomic count of total sensor instances

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

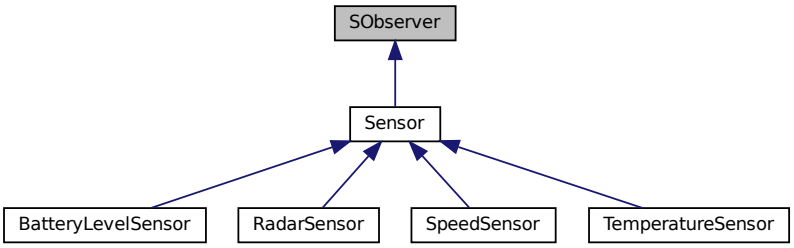
- Sensors/Sensor.hpp
- Sensors/SpeedSensor.cpp

3.10 SObserver Class Reference

This interface defines the basic functionality for all sensor types, including speed, temperature, radar, and battery level.

```
#include <Sensor.hpp>
```

Inheritance diagram for SObserver:



Public Member Functions

- virtual void updateECU (std::weak_ptr< ECU > E)=0

  Update a subscribed ECU with the latest sensor data.

3.10.1 Detailed Description

This interface defines the basic functionality for all sensor types, including speed, temperature, radar, and battery level.

It declares several virtual functions that must be overridden by derived sensor classes to implement specific behaviors.

Observer interface for the Observer design pattern.

The sensor acts as an observer for a list of subscribed ECUs, notifying them of updates when sensor data changes.

3.10.2 Member Function Documentation

3.10.2.1 updateECU()

```
virtual void SObserver::updateECU (
            std::weak_ptr< ECU > E )  [pure virtual]
```

Update a subscribed ECU with the latest sensor data.

Parameters

E	A weak pointer to the ECU that will be updated.

Implemented in Sensor, TemperatureSensor, SpeedSensor, BatteryLevelSensor, and RadarSensor.

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

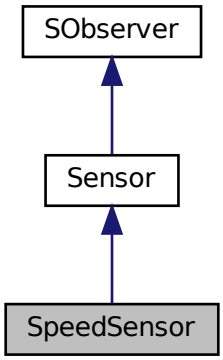
- Sensors/Sensor.hpp

3.11 SpeedSensor Class Reference

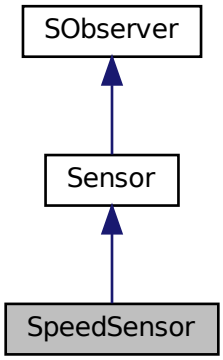
Represents a speed sensor that derives from the Sensor interface.

```
#include <SpeedSensor.hpp>
```

Inheritance diagram for SpeedSensor:



Collaboration diagram for SpeedSensor:



Public Member Functions

- SpeedSensor ()

  Constructs a SpeedSensor object.
- ~SpeedSensor ()

  Destroys the SpeedSensor object.
- int getSensorID () const

  Gets the unique identifier for the speed sensor.
- double GetSensorData () override

  Retrieves the current speed sensor data.
- void sensorRead () override

Mimics reading sensor data by generating a random speed value.

- int getSensorCount () const

  Retrieves the total count of speed sensors.
- void PrintInfo () override

  Prints information about the speed sensor.
- std::string getType () override

  Gets the type of the sensor.
- **SpeedSensor** (const SpeedSensor &)=delete
- SpeedSensor & **operator=** (const SpeedSensor &)=delete
- **SpeedSensor** (SpeedSensor &&)=delete
- SpeedSensor & **operator=** (SpeedSensor &&)=delete
- void AttachECU (std::weak_ptr< ECU > E) override

  Attaches an ECU to the speed sensor for notifications.
- void DeAttachECU (std::weak_ptr< ECU > E) override

  Detaches an ECU from the speed sensor.
- void updateECU (std::weak_ptr< ECU > E) override

  Updates the specified ECU with the latest speed data.
- void NotifyAllECUs () override

  Notifies all subscribed ECUs of the data update.
- virtual int getTotalSensorsCount () override

  Gets the total count of sensor instances.

Additional Inherited Members

3.11.1 Detailed Description

Represents a speed sensor that derives from the Sensor interface.

This class implements the specific functionality for a speed sensor, including reading speed data and managing its subscribed ECUs.

3.11.2 Constructor & Destructor Documentation

3.11.2.1 SpeedSensor()

```
SpeedSensor::SpeedSensor ( )
```

Constructs a SpeedSensor object.

Constructs a SpeedSensor object and initializes it.

Increments the sensor count and logs the sensor information.

3.11.3 Member Function Documentation

3.11.3.1 AttachECU()

```
void SpeedSensor::AttachECU (
            std::weak_ptr< ECU > Ecu )  [override], [virtual]
```

Attaches an ECU to the speed sensor for notifications.

Parameters

E	A weak pointer to the ECU to be attached.
Ecu	A weak pointer to the ECU to be attached.

Implements Sensor.

3.11.3.2 DeAttachECU()

```
void SpeedSensor::DeAttachECU (
            std::weak_ptr< ECU > Ecu )  [override], [virtual]
```

Detaches an ECU from the speed sensor.

Parameters

E	A weak pointer to the ECU to be detached.
Ecu	A weak pointer to the ECU to be detached.

Implements Sensor.

3.11.3.3 getSensorCount()

```
int SpeedSensor::getSensorCount ( ) const
```

Retrieves the total count of speed sensors.

Gets the count of speed sensors currently active.

Returns

An integer representing the total number of speed sensors.

The total count of speed sensors.

3.11.3.4 GetSensorData()

```
double SpeedSensor::GetSensorData ( )  [override], [virtual]
```

Retrieves the current speed sensor data.

Retrieves the current sensor data (speed).

Returns

A double representing the current speed.

A double representing the current speed value.

Implements Sensor.

3.11.3.5 getSensorID()

```
int SpeedSensor::getSensorID ( ) const  [virtual]
```

Gets the unique identifier for the speed sensor.

Gets the ID of the sensor.

Returns

An integer representing the sensor ID.

The unique identifier for the speed sensor.

Implements Sensor.

3.11.3.6 getTotalSensorsCount()

```
int SpeedSensor::getTotalSensorsCount ( )  [override], [virtual]
```

Gets the total count of sensor instances.

Returns

An integer representing the total number of sensors.

Implements Sensor.

3.11.3.7 getType()

```
std::string SpeedSensor::getType ( )  [override], [virtual]
```

Gets the type of the sensor.

Returns

A string representing the sensor type.

A string representing the type of the speed sensor.

Implements Sensor.

3.11.3.8 NotifyAllECUs()

```
void SpeedSensor::NotifyAllECUs ( )  [override], [virtual]
```

Notifies all subscribed ECUs of the data update.

Notifies all subscribed ECUs with the latest speed data.

Implements Sensor.

3.11.3.9 PrintInfo()

```
void SpeedSensor::PrintInfo ( )  [override], [virtual]
```

Prints information about the speed sensor.

Logs information about the new speed sensor created.

Implements Sensor.

3.11.3.10 sensorRead()

```
void SpeedSensor::sensorRead ( )  [override], [virtual]
```

Mimics reading sensor data by generating a random speed value.

Reads the sensor data by generating a random speed value.

Implements Sensor.

3.11.3.11 updateECU()

```
void SpeedSensor::updateECU (
            std::weak_ptr< ECU > E )  [override], [virtual]
```

Updates the specified ECU with the latest speed data.

Parameters

E	A weak pointer to the ECU to be updated.

Implements Sensor.

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

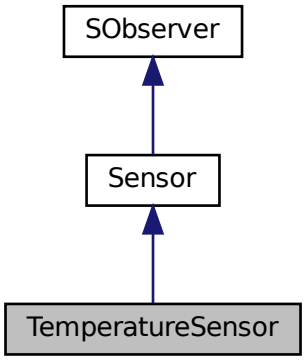
- Sensors/SpeedSensor.hpp
- Sensors/SpeedSensor.cpp

3.12 TemperatureSensor Class Reference

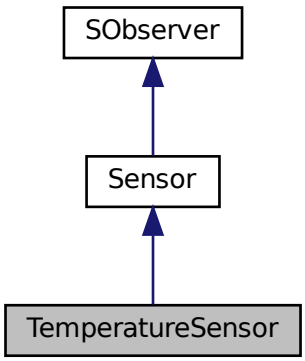
Represents a temperature sensor that inherits from the Sensor base class.

`#include <TemperatureSensor.hpp>`

Inheritance diagram for TemperatureSensor:



Collaboration diagram for TemperatureSensor:



Public Member Functions

- TemperatureSensor ()

  Default constructor for the TemperatureSensor class.
- ~TemperatureSensor ()

  Destructor for the TemperatureSensor class.
- int getSensorID () const override

  Retrieves the unique sensor ID.
- double GetSensorData () override

  Retrieves the temperature data.
- void sensorRead () override

  Reads the temperature sensor data by generating a random value.
- int getSensorCount () const

  Retrieves the count of existing temperature sensors.
- void PrintInfo () override

  Prints information about the temperature sensor.
- std::string getType () override

  Retrieves the type of the sensor.
- TemperatureSensor (const TemperatureSensor &)=delete

  Deleted copy constructor.
- TemperatureSensor & operator= (const TemperatureSensor &)=delete

  Deleted assignment operator.
- TemperatureSensor (TemperatureSensor &&)=delete

  Deleted move constructor.
- TemperatureSensor & operator= (TemperatureSensor &&)=delete

  Deleted move assignment operator.
- void AttachECU (std::weak_ptr< ECU > E) override

  Attaches an ECU to the temperature sensor.
- void DeAttachECU (std::weak_ptr< ECU > E) override

  Detaches an ECU from the temperature sensor.
- void updateECU (std::weak_ptr< ECU > E) override

  Updates the attached ECU with the latest sensor data.
- void NotifyAllECUs () override

  Notifies all attached ECUs with the latest sensor data.
- virtual int getTotalSensorsCount () override

  Retrieves the total number of temperature sensors created.

Additional Inherited Members

3.12.1 Detailed Description

Represents a temperature sensor that inherits from the Sensor base class.

The TemperatureSensor class simulates a physical temperature sensor. It provides functionalities for obtaining temperature readings, attaching/detaching ECUs, and notifying subscribed ECUs about updates.

3.12.2 Constructor & Destructor Documentation

3.12.2.1 TemperatureSensor()

```
TemperatureSensor::TemperatureSensor ( )
```

Default constructor for the TemperatureSensor class.

Default constructor for the TemperatureSensor class. Initializes sensor ID and temperature, logs sensor creation, and updates the total sensor count.

3.12.2.2 ~TemperatureSensor()

```
TemperatureSensor::~TemperatureSensor ( )
```

Destructor for the TemperatureSensor class.

Destructor for the TemperatureSensor class. Decreases the sensor count and logs the destruction of the sensor.

3.12.3 Member Function Documentation

3.12.3.1 AttachECU()

```
void TemperatureSensor::AttachECU (
            std::weak_ptr< ECU > Ecu )  [override], [virtual]
```

Attaches an ECU to the temperature sensor.

Parameters

E	A weak pointer to the ECU to be attached.
Ecu	A weak pointer to the ECU to be attached.

Implements Sensor.

3.12.3.2 DeAttachECU()

```
void TemperatureSensor::DeAttachECU (
            std::weak_ptr< ECU > Ecu )  [override], [virtual]
```

Detaches an ECU from the temperature sensor.

Parameters

E	A weak pointer to the ECU to be detached.
Ecu	A weak pointer to the ECU to be detached.

Implements Sensor.

3.12.3.3 getSensorCount()

```
int TemperatureSensor::getSensorCount ( ) const
```

Retrieves the count of existing temperature sensors.

Gets the count of existing temperature sensors.

Returns

The number of temperature sensors created.

3.12.3.4 GetSensorData()

```
double TemperatureSensor::GetSensorData ( )  [override], [virtual]
```

Retrieves the temperature data.

Gets the sensor data.

Returns

The current temperature reading.

Implements Sensor.

3.12.3.5 getSensorID()

```
int TemperatureSensor::getSensorID ( ) const  [override], [virtual]
```

Retrieves the unique sensor ID.

Returns

The ID of the sensor.

Implements Sensor.

3.12.3.6 getTotalSensorsCount()

```
int TemperatureSensor::getTotalSensorsCount ( )  [override], [virtual]
```

Retrieves the total number of temperature sensors created.

Gets the total number of sensors created.

Returns

The total sensor count.

The total sensor count from the base Sensor class.

Implements Sensor.

3.12.3.7 getType()

```
std::string TemperatureSensor::getType ( )  [override], [virtual]
```

Retrieves the type of the sensor.

Gets the type of the sensor.

Returns

The type of the sensor as a string.

A string representing the type of the sensor.

Implements Sensor.

3.12.3.8 operator=() [1/2]

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

Deleted assignment operator.

Returns

A reference to the current instance.

3.12.3.9 operator=() [2/2]

```
TemperatureSensor& TemperatureSensor::operator= (
            TemperatureSensor &&  )  [delete]
```

Deleted move assignment operator.

Returns

A reference to the current instance.

3.12.3.10 PrintInfo()

```
void TemperatureSensor::PrintInfo ( )  [override], [virtual]
```

Prints information about the temperature sensor.

Prints information about the temperature sensor. Logs the sensor's type, ID, and count.

Implements Sensor.

3.12.3.11 updateECU()

```
void TemperatureSensor::updateECU (
            std::weak_ptr< ECU > E )  [override], [virtual]
```

Updates the attached ECU with the latest sensor data.

Parameters

E	A weak pointer to the ECU to be updated.

Implements Sensor.

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

- Sensors/TemperatureSensor.hpp
- Sensors/TemperatureSensor.cpp

Index