My Project

Generated by Doxygen 1.12.0

Chapter 1

Vehicle Telemetry and Diagnostic System Simulation Project

1.1 Project Description:

A customer has a car and wants to measure its telemetries and control its system. This vehicle is equipped with state-of-the-art sensors and systems designed to ensure safe and efficient operation under various driving conditions.

1.2 Sensors:

SpeedSensor: Simulates the vehicle's speed. RadarSensor: Tracks Radar distance for obstacle detection. TemperatureSensor: Monitors the engine's temperature. Battery: Display the battery's charge level and temperature.

1.3 ECUs:

AdaptiveCruiseControl: changes speed or break when temperature or obstacale detection happens.

1.4 Diagnostics:

Regularly check all monitored parameters, identify any potential issues (such as low fuel, excessive speed, overheating, low battery charge, or unsafe proximity to another vehicle)

1.5 Logger:

Single instance logger to be used among all the classes for logging

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

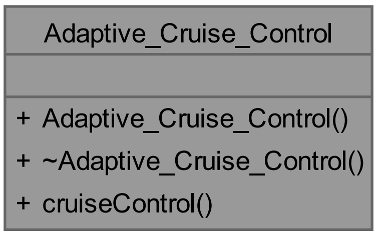
Class Documentation

5.1 Adaptive_Cruise_Control Class Reference

This class is for Adaptive_Cruise_Control.

```
#include <Adaptive_Cruise_Control.hpp>
```

Collaboration diagram for Adaptive_Cruise_Control:



Public Member Functions

- Adaptive_Cruise_Control (Speed_Sensor ∗speed_sensor, Radar_Sensor ∗radar_sensor, Temp_Sensor ∗temp_sensor)

  Construct a new Adaptive_Cruise_Control::Adaptive_Cruise_Control object.
- ∼Adaptive_Cruise_Control ()

  Destroy the Adaptive_Cruise_Control::Adaptive_Cruise_Control object.
- void cruiseControl ()

  Function to control the car.

5.1.1 Detailed Description

This class is for Adaptive_Cruise_Control.

It controls the speed or brake when the car temperature is high or an obstacale detection happened

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Adaptive_Cruise_Control()

```
Adaptive_Cruise_Control::Adaptive_Cruise_Control (
            Speed_Sensor * speed_sensor,
            Radar_Sensor * radar_sensor,
            Temp_Sensor * temp_sensor)
```

Construct a new Adaptive_Cruise_Control::Adaptive_Cruise_Control object.

Parameterized Constructor

Parameters

speed_sensor	
radar_sensor	
temp_sensor	

< Assign object of the singleton logger

5.1.2.2 ~Adaptive_Cruise_Control()

```
Adaptive_Cruise_Control::~Adaptive_Cruise_Control ()
```

Destroy the Adaptive_Cruise_Control::Adaptive_Cruise_Control object.

Default Destructor

5.1.3 Member Function Documentation

5.1.3.1 cruiseControl()

```
void Adaptive_Cruise_Control::cruiseControl ()
```

Function to control the car.

This function automatically brake the car if an object is detected at a distance smaller than the SAFE_DISTANCE. Slow down the speed if its temperature is higher than SAFE_TEMPERATURE. Stop the car if its temperature is higher than THRESHOLD_TEMPERATURE

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

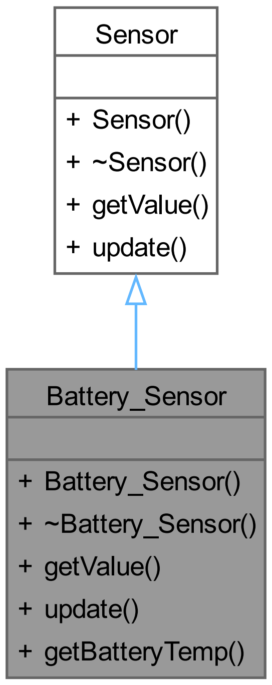
- Adaptive_Cruise_Control.hpp
- Adaptive_Cruise_Control.cpp

5.2 Battery_Sensor Class Reference

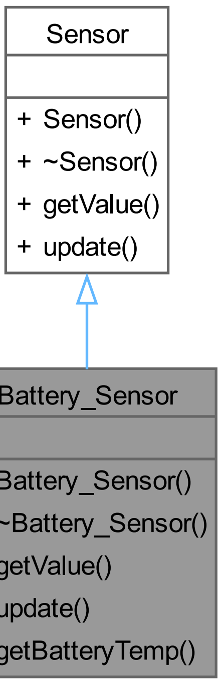
This class is for the car battery.

```
#include <Battery_Sensor.hpp>
```

Inheritance diagram for Battery_Sensor:



Collaboration diagram for Battery_Sensor:



Public Member Functions

- Battery_Sensor ()
- ∼Battery_Sensor ()
- unsigned int getValue () const override

  Virtual function to get the temperature value.
- void update () override

  Virtual function to update the temperature value.
- unsigned int getBatteryTemp () const

  Virtual function to get the charge value.

Public Member Functions inherited from Sensor

- Sensor ()
- ∼Sensor ()

5.2.1 Detailed Description

This class is for the car battery.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Battery_Sensor()

```
Battery_Sensor::Battery_Sensor ()  [inline]
```

< Default Constructor Default Destructor

5.2.2.2 ∼Battery_Sensor()

```
Battery_Sensor::~Battery_Sensor ()  [inline]
```

Setters and Getters Get the temperature value

5.2.3 Member Function Documentation

5.2.3.1 getBatteryTemp()

```
unsigned int Battery_Sensor::getBatteryTemp () const
```

Virtual function to get the charge value.

Returns

unsigned int: The battery's temperature

5.2.3.2 getValue()

```
unsigned int Battery_Sensor::getValue () const  [override], [virtual]
```

Virtual function to get the temperature value.

Update the temperature value

Returns

unsigned int: The battery-charge level

Implements Sensor.

5.2.3.3 update()

```
void Battery_Sensor::update ()  [override], [virtual]
```

Virtual function to update the temperature value.

Get the charge value

Implements Sensor.

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

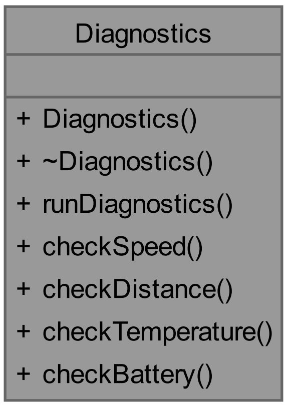
- Battery_Sensor.hpp
- Battery_Sensor.cpp

5.3 Diagnostics Class Reference

This class is for car diagnostics.

```
#include <Diagnostics.hpp>
```

Collaboration diagram for Diagnostics:



Public Member Functions

- Diagnostics (Speed_Sensor ∗speed, Radar_Sensor ∗radar, Temp_Sensor ∗temp, Battery_Sensor ∗batt)

  Construct a new Diagnostics:: Diagnostics object and initializes the members values with the given arguments.
- ∼Diagnostics ()

  Destroy the Diagnostics:: Diagnostics object.
- void runDiagnostics ()

  Function to run all car diagnostics.
- bool checkSpeed ()

  Function to check speed.
- bool checkDistance ()

  Function to check the distance between the car and front objects.
- bool checkTemperature ()

  Function to check the car temperature.
- bool checkBattery ()

  Function to check battery-charge level.

5.3.1 Detailed Description

This class is for car diagnostics.

It runs car diagnostics which checks car speed, temperature, safe distance, and battery level

5.3.2 Constructor & Destructor Documentation

5.3.2.1 Diagnostics()

```
Diagnostics::Diagnostics (
            Speed_Sensor * speed,
            Radar_Sensor * radar,
            Temp_Sensor * temp,
            Battery_Sensor * batt)
```

Construct a new Diagnostics:: Diagnostics object and initializes the members values with the given arguments.

Parameterized Constructor

Parameters

speed	Pointer to the Speed_Sensor object
radar	Pointer to the Radar_Sensor object
temp	Pointer to the Temp_Sensor object
batt	Pointer to the Battery_Sensor object

< Singleton logger

5.3.2.2 ∼Diagnostics()

```
Diagnostics::∼Diagnostics ()
```

Destroy the Diagnostics:: Diagnostics object.

Default Destructor

5.3.3 Member Function Documentation

5.3.3.1 checkBattery()

```
bool Diagnostics::checkBattery ()
```

Function to check battery-charge level.

It gives alert to the driver if the battery level is higher than THRESHOLD_LEVEL

Returns

true

false

5.3.3.2 checkDistance()

```
bool Diagnostics::checkDistance ()
```

Function to check the distance between the car and front objects.

It gives alert to the driver if the distance is lower than SAFE_DISTANCE

Returns

true: If the distance is in the safe range

false: If the distance lower than the SAFE_DISTANCE

5.3.3.3 checkSpeed()

```
bool Diagnostics::checkSpeed ()
```

Function to check speed.

It gives alert to the driver if the speed is higher than SAFE_SPEED

Returns

true: If the speed is in the safe range

false: If the speed exceeds the SAFE_SPEED

5.3.3.4 checkTemperature()

```
bool Diagnostics::checkTemperature ()
```

Function to check the car temperature.

It gives alert to the driver if the temperature is higher than SAFE_TEMPERATURE

Returns

true: If the car temperature is in the safe range

false: If the car temperature is higher than SAFE_TEMPERATURE

5.3.3.5 runDiagnostics()

```
void Diagnostics::runDiagnostics ()
```

Function to run all car diagnostics.

It checks the values of all car sensors and give alert to the driver if upnormal event happened

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

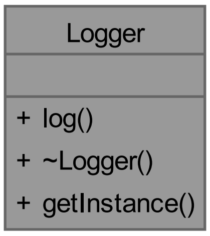
- Diagnostics.hpp
- Diagnostics.cpp

5.4 Logger Class Reference

This class is for logging messages.

```
#include <Logger.hpp>
```

Collaboration diagram for Logger:



Public Member Functions

- void log (const std::string &message, LogLevel level, OutputDestination dest=FILE_ONLY)

    Function to log messages in the log file only or in console too.
- ∼Logger ()

    Destroy the Logger:: Logger object.

Static Public Member Functions

- static Logger ∗ getInstance ()

    Static function to get the singleton object.

5.4.1 Detailed Description

This class is for logging messages.

This class is designed as singleton Pattern

5.4.2 Constructor & Destructor Documentation

5.4.2.1 ∼Logger()

```
Logger::∼Logger ()
```

Destroy the Logger:: Logger object.

Destructor

5.4.3 Member Function Documentation

5.4.3.1 getInstance()

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

Static function to get the singleton object.

Public method

Returns

Logger* : pointer to the static object of Logger class

5.4.3.2 log()

```
void Logger::log (
            const std::string & message,
            LogLevel level,
            OutputDestination dest = FILE_ONLY)
```

Function to log messages in the log file only or in console too.

Public method

Parameters

message	String of the message content
level	Level of the log message
dest	Destination to log the message

Get current time and write it in the log file

Add log level

Add the log message

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

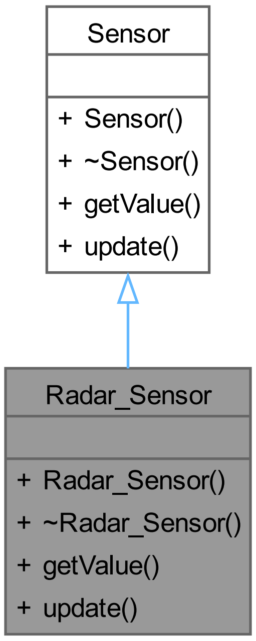
- Logger.hpp
- Logger.cpp

5.5 Radar_Sensor Class Reference

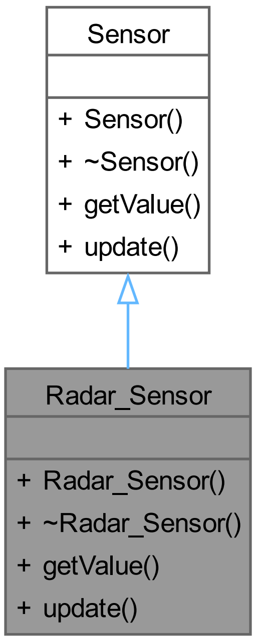
This class is for the radar sensor.

```
#include <Radar_Sensor.hpp>
```

Inheritance diagram for Radar_Sensor:



Collaboration diagram for Radar_Sensor:



Public Member Functions

- Radar_Sensor ()

    Construct a new Radar_Sensor::Radar_Sensor object and initializes the distance value with zero.
- ∼Radar_Sensor ()

    Destroy the Radar_Sensor::Radar_Sensor object.
- unsigned int getValue () const override

    Virtual function to get the distance between the car and front object.
- void update () override

    Virtual function to update the distance value.

Public Member Functions inherited from Sensor

- Sensor ()
- ∼Sensor ()

5.5.1 Detailed Description

This class is for the radar sensor.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 Radar_Sensor()

```
Radar_Sensor::Radar_Sensor ()
```

Construct a new Radar_Sensor::Radar_Sensor object and initializes the distance value with zero.

Member Intialize Constructor

5.5.2.2 ∼Radar_Sensor()

```
Radar_Sensor::∼Radar_Sensor ()
```

Destroy the Radar_Sensor::Radar_Sensor object.

Default Destructor

5.5.3 Member Function Documentation

5.5.3.1 getValue()

```
unsigned int Radar_Sensor::getValue () const  [override], [virtual]
```

Virtual function to get the distance between the car and front object.

Returns

unsigned int: The distance between the car and front object

Implements Sensor.

5.5.3.2 update()

```
void Radar_Sensor::update ()  [override], [virtual]
```

Virtual function to update the distance value.

Implements Sensor.

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

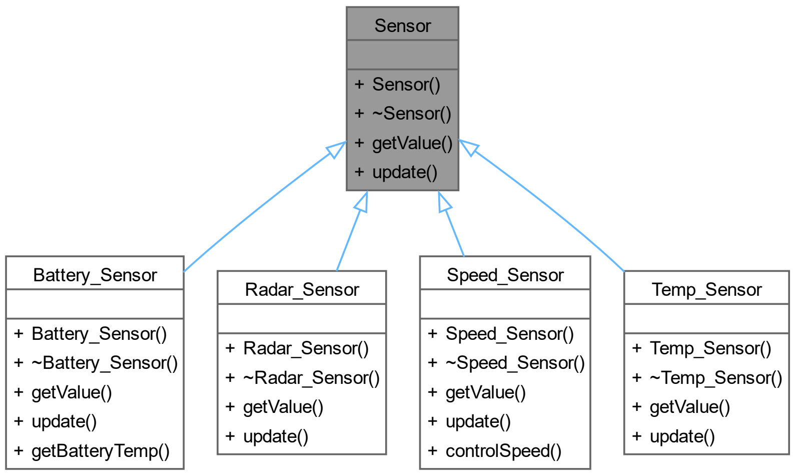
- Radar_Sensor.hpp
- Radar_Sensor.cpp

5.6 Sensor Class Reference

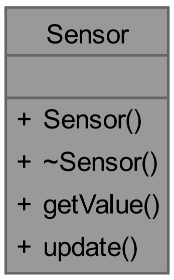
This is an interface Class provides the main methods inherited in all sensors.

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

Inheritance diagram for Sensor:



Collaboration diagram for Sensor:



Public Member Functions

- Sensor ()
- ∼Sensor ()
- virtual unsigned int getValue () const =0
- virtual void update ()=0

5.6.1 Detailed Description

This is an interface Class provides the main methods inherited in all sensors.

5.6.2 Constructor & Destructor Documentation

5.6.2.1 Sensor()

```
Sensor::Sensor ()  [inline]
```

< Default Constructor Default Destructor

5.6.2.2 ∼Sensor()

```
Sensor::∼Sensor ()  [inline]
```

Virtual functions

5.6.3 Member Function Documentation

5.6.3.1 getValue()

```
virtual unsigned int Sensor::getValue () const  [pure virtual]
```

Implemented in Battery_Sensor, Radar_Sensor, Speed_Sensor, and Temp_Sensor.

5.6.3.2 update()

```
virtual void Sensor::update ()  [pure virtual]
```

Implemented in Battery_Sensor, Radar_Sensor, Speed_Sensor, and Temp_Sensor.

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

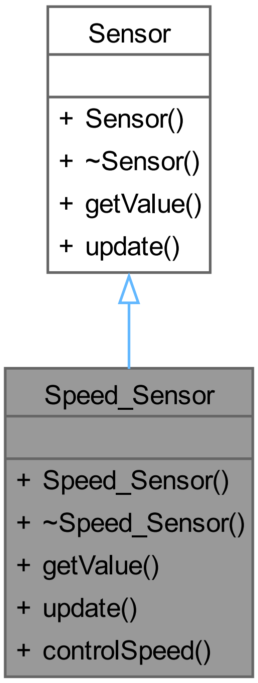
- Sensor.hpp

5.7 Speed_Sensor Class Reference

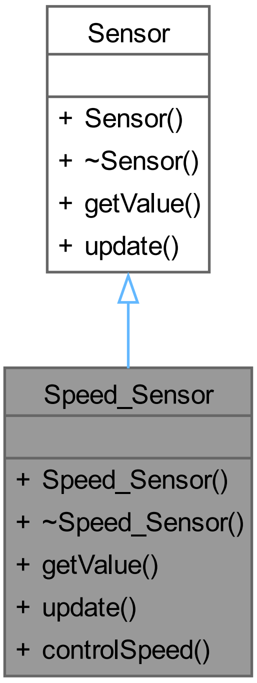
This class is for the car's speed sensor.

```
#include <Speed_Sensor.hpp>
```

Inheritance diagram for Speed_Sensor:



Collaboration diagram for Speed_Sensor:



Public Member Functions

- Speed_Sensor ()

    Construct a new Speed_Sensor::Speed_Sensor object and initializes the speed value with zero.
- ∼Speed_Sensor ()

    Destroy the Speed_Sensor::Speed_Sensor object.
- unsigned int getValue () const override

    Virtual function to get the speed value.
- void update () override

    Virtual function to update the speed in range 0 to 250 Km/h.
- void controlSpeed (unsigned int speed)

    Function to control the speed value.

Public Member Functions inherited from Sensor

- Sensor ()
- ∼Sensor ()

5.7.1 Detailed Description

This class is for the car's speed sensor.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 Speed_Sensor()

```
Speed_Sensor::Speed_Sensor ()
```

Construct a new Speed_Sensor::Speed_Sensor object and initializes the speed value with zero.

Member Initialize Constructor

5.7.2.2 ∼Speed_Sensor()

```
Speed_Sensor::∼Speed_Sensor ()
```

Destroy the Speed_Sensor::Speed_Sensor object.

Default Destructor

5.7.3 Member Function Documentation

5.7.3.1 controlSpeed()

```
void Speed_Sensor::controlSpeed (
            unsigned int speed)
```

Function to control the speed value.

Parameters

speed	: The new speed of the car

5.7.3.2 getValue()

```
unsigned int Speed_Sensor::getValue () const  [override], [virtual]
```

Virtual function to get the speed value.

Returns

unsigned int : The car speed

Implements Sensor.

5.7.3.3 update()

```
void Speed_Sensor::update ()  [override], [virtual]
```

Virtual function to update the speed in range 0 to 250 Km/h.

Implements Sensor.

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

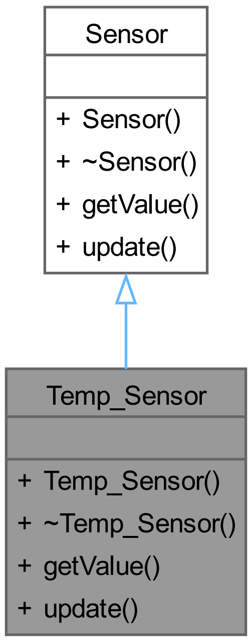
- Speed_Sensor.hpp
- Speed_Sensor.cpp

5.8 Temp_Sensor Class Reference

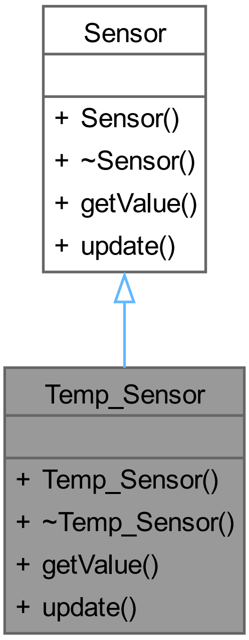
This class is for the car's temperature sensor.

`#include <Temp_Sensor.hpp>`

Inheritance diagram for Temp_Sensor:



Collaboration diagram for Temp_Sensor:



Public Member Functions

- Temp_Sensor ()

    Construct a new Temp_Sensor::Temp_Sensor object and initializes the temperature value with 12 celisius.
- ∼Temp_Sensor ()

    Destroy the Temp_Sensor::Temp_Sensor object.
- unsigned int getValue () const override

    Virtual function to get the temperature value.
- void update () override

    Virtual function to update the temperature value.

Public Member Functions inherited from Sensor

- Sensor ()
- ∼Sensor ()

5.8.1 Detailed Description

This class is for the car's temperature sensor.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 Temp_Sensor()

```
Temp_Sensor::Temp_Sensor ()
```

Construct a new Temp_Sensor::Temp_Sensor object and initializes the temperature value with 12 celisius.

Member Initialize Constructor

5.8.2.2 ∼Temp_Sensor()

```
Temp_Sensor::∼Temp_Sensor ()
```

Destroy the Temp_Sensor::Temp_Sensor object.

Default Destructor

5.8.3 Member Function Documentation

5.8.3.1 getValue()

```
unsigned int Temp_Sensor::getValue () const  [override], [virtual]
```

Virtual function to get the temperature value.

Returns

unsigned int : The car temperature

Implements Sensor.

5.8.3.2 update()

```
void Temp_Sensor::update ()  [override], [virtual]
```

Virtual function to update the temperature value.

Implements Sensor.

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

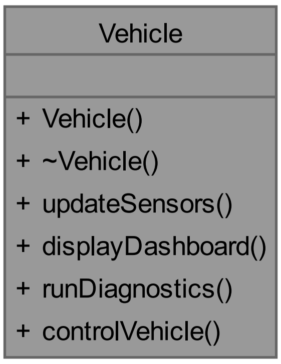
- Temp_Sensor.hpp
- Temp_Sensor.cpp

5.9 Vehicle Class Reference

This is the Vehicle Class.

```
#include <Vehicle.hpp>
```

Collaboration diagram for Vehicle:



Public Member Functions

- Vehicle ()

  Construct a new Vehicle:: Vehicle object.
- ∼Vehicle ()

  Destroy the Vehicle:: Vehicle object.
- void updateSensors ()

  Function to update readings of all the sensors.
- void displayDashboard ()

  Function to display the vehicle's status on the dashboard.
- void runDiagnostics ()

  Function to run diagnostics to check vehicle status.
- void controlVehicle ()

  Function to control the vehicle behavior (e.g., Adaptive Cruise Control)

5.9.1 Detailed Description

This is the Vehicle Class.

It has all sensors and objects as private members

5.9.2 Constructor & Destructor Documentation

5.9.2.1 Vehicle()

```
Vehicle::Vehicle ()
```

Construct a new Vehicle:: Vehicle object.

It calls all object constructors and initialize the logger object. Delegation Constructor.

5.9.2.2 ∼Vehicle()

```
Vehicle::∼Vehicle ()
```

Destroy the Vehicle:: Vehicle object.

Default Destructor

5.9.3 Member Function Documentation

5.9.3.1 controlVehicle()

```
void Vehicle::controlVehicle ()
```

Function to control the vehicle behavior (e.g., Adaptive Cruise Control)

It calls cruiseControl() method in Adaptive_Cruise_Control class

5.9.3.2 displayDashboard()

```
void Vehicle::displayDashboard ()
```

Function to display the vehicle's status on the dashboard.

< Log dashboard display

5.9.3.3 runDiagnostics()

```
void Vehicle::runDiagnostics ()
```

Function to run diagnostics to check vehicle status.

It calls runDiagnostics() method in Diagnostics class

5.9.3.4 updateSensors()

```
void Vehicle::updateSensors ()
```

Function to update readings of all the sensors.

It logs all readings in the log file

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

- Vehicle.hpp
- Vehicle.cpp

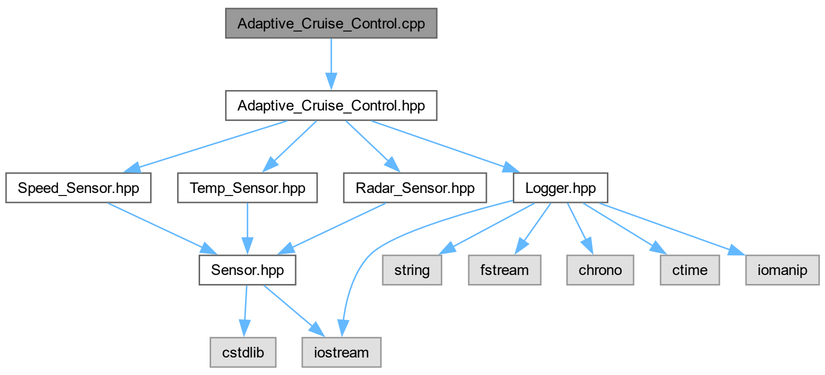
Chapter 6

File Documentation

6.1 Adaptive_Cruise_Control.cpp File Reference

This is a cpp file for Adaptive_Cruise_Control class.

`#include "Adaptive_Cruise_Control.hpp"`
Include dependency graph for Adaptive_Cruise_Control.cpp:



6.1.1 Detailed Description

This is a cpp file for Adaptive_Cruise_Control class.

Author

Abdelrahman Abdelhalem (`abdohalem305@gmail.com`)

This class is for controlling the car if unexpected event occured

Version

0.1

Date

2024-09-27

Copyright

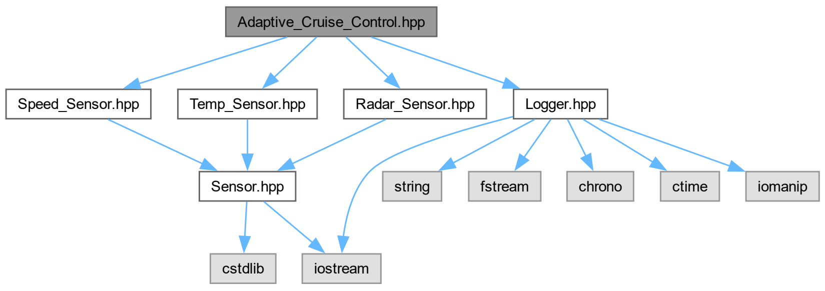
Copyright (c) 2024

6.2 Adaptive_Cruise_Control.hpp File Reference

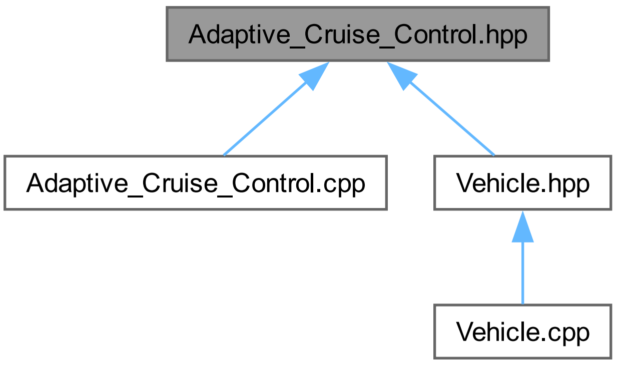
This is a header file for Adaptive_Cruise_Control class.

```
#include "Speed_Sensor.hpp"
#include "Temp_Sensor.hpp"
#include "Radar_Sensor.hpp"
#include "Logger.hpp"
```

Include dependency graph for Adaptive_Cruise_Control.hpp:



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



Classes

- class Adaptive_Cruise_Control

  This class is for Adaptive_Cruise_Control.

Macros

- #define **SLOW_CAR** (unsigned int)50

  Macros for Adaptive_Cruise_Control orders.
- #define **STOP_CAR** (unsigned int)0

6.2.1 Detailed Description

This is a header file for Adaptive_Cruise_Control class.

Author

Abdelrahman Abdelhalem (abdohalem305@gmail.com)

This class is for controlling the car if unexpected event occured

Version

0.1

Date

2024-09-27

Copyright

Copyright (c) 2024

6.3 Adaptive_Cruise_Control.hpp

Go to the documentation of this file.

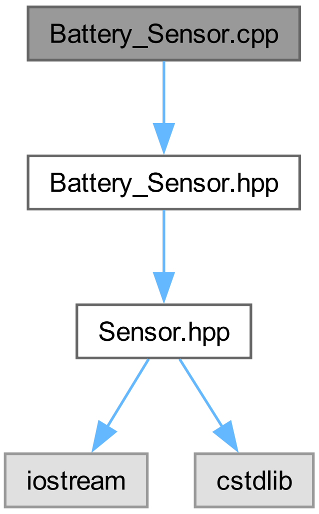
```
#ifndef __ADAPTIVE_CRUISE_CONTROL__H__
#define __ADAPTIVE_CRUISE_CONTROL__H__

#include "Speed_Sensor.hpp"
#include "Temp_Sensor.hpp"
#include "Radar_Sensor.hpp"
#include "Logger.hpp"

#define SLOW_CAR                (unsigned int)50
#define STOP_CAR                (unsigned int)0

class Adaptive_Cruise_Control
{
private:
    Speed_Sensor *speed_sensor;
    Radar_Sensor *radar_sensor;
    Temp_Sensor *temp_sensor;
    Logger *logger;
public:
    /* Parameterized Constructor */
    Adaptive_Cruise_Control(Speed_Sensor *speed_sensor, Radar_Sensor *radar_sensor, Temp_Sensor
*temp_sensor) ;
    /* Default Destructor */
    ~Adaptive_Cruise_Control();
    /* Methods */
    void cruiseControl();
};


#endif
```

6.4 Battery_Sensor.cpp File Reference

This is a cpp file for Battery_Sensor class.

```
#include "Battery_Sensor.hpp"
```

Include dependency graph for Battery_Sensor.cpp:



6.4.1 Detailed Description

This is a cpp file for Battery_Sensor class.

Author

Abdelrahman Abdelhalem (`abdohalem305@gmail.com`)

This class provides the main functionality of the car battery

Version

0.1

Date

2024-09-27

Copyright

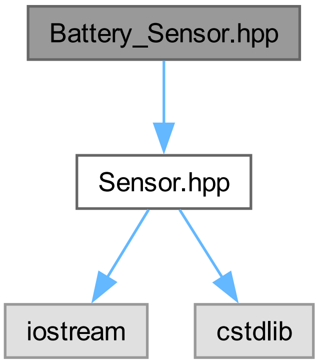
Copyright (c) 2024

6.5 Battery_Sensor.hpp File Reference

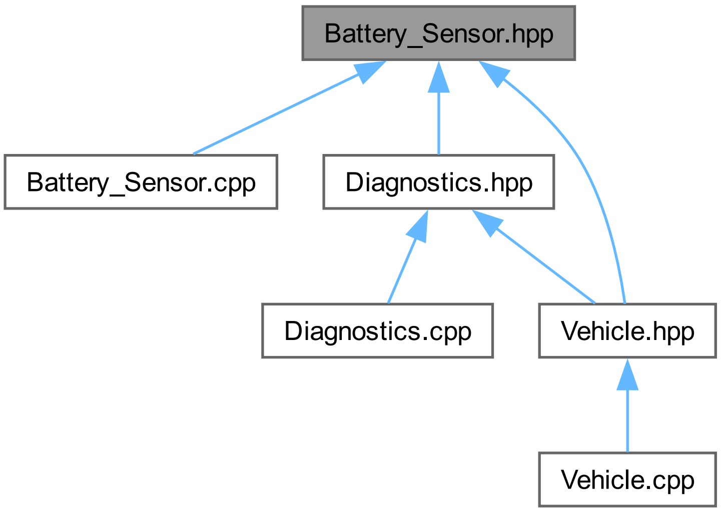
This is a header file for Battery_Sensor class.

```
#include "Sensor.hpp"
```

Include dependency graph for Battery_Sensor.hpp:



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



Classes

- class Battery_Sensor

  This class is for the car battery.

Macros

- #define **THRESHOLD_LEVEL** 20

  Macro for the battery threshold level.

6.5.1 Detailed Description

This is a header file for Battery_Sensor class.

Author

Abdelrahman Abdelhalem (abdohalem305@gmail.com)

This class provides the main functionality of the car battery

Version

0.1

Date

2024-09-27

Copyright

Copyright (c) 2024

6.6 Battery_Sensor.hpp

Go to the documentation of this file.

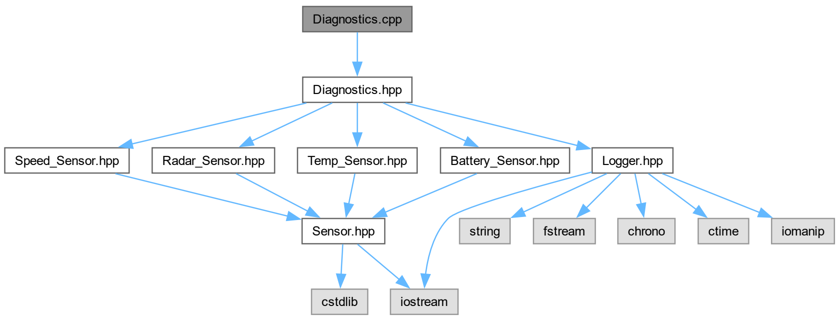
```
#ifndef __BATTERY_SENSOR__H__
#define __BATTERY_SENSOR__H__

#include "Sensor.hpp"

#define THRESHOLD_LEVEL 20

class Battery_Sensor : public Sensor {
private:
    unsigned int temperature_;      // from 0 to 50 celesius
    unsigned int charge_;           // from 0% to 100%
public:
    Battery_Sensor() : temperature_(0), charge_(0) {}
    ~Battery_Sensor() {}
    unsigned int getValue() const override;
    void update() override;
    unsigned int getBatteryTemp() const ;
};

#endif
```

6.7 Diagnostics.cpp File Reference

This is a cpp file for Diagnostics class.

`#include "Diagnostics.hpp"`

Include dependency graph for Diagnostics.cpp:



6.7.1 Detailed Description

This is a cpp file for Diagnostics class.

Author

Abdelrahman Abdelhalem (`abdohalem305@gmail.com`)

This class is for running and checking all the car diagnostics

Version

0.1

Date

2024-09-27

Copyright

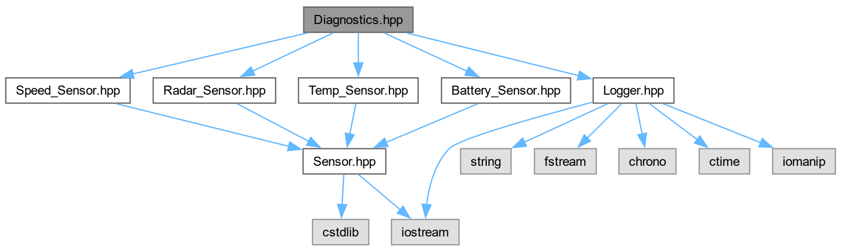
Copyright (c) 2024

6.8 Diagnostics.hpp File Reference

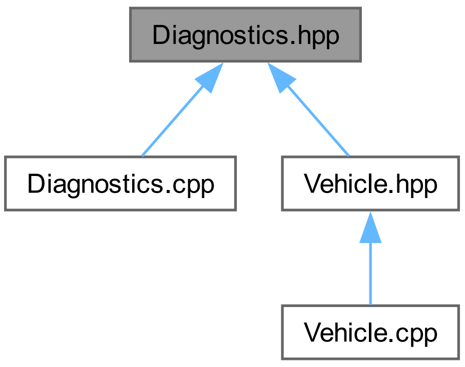
This is a header file for Diagnostics class.

```
#include "Speed_Sensor.hpp"
#include "Radar_Sensor.hpp"
#include "Temp_Sensor.hpp"
#include "Battery_Sensor.hpp"
#include "Logger.hpp"
```

Include dependency graph for Diagnostics.hpp:



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



Classes

- class Diagnostics

  This class is for car diagnostics.

6.8.1 Detailed Description

This is a header file for Diagnostics class.

Author

Abdelrahman Abdelhalem (abdohalem305@gmail.com)

This class is for running and checking all the car diagnostics

Version

0.1

Date

2024-09-27

Copyright

Copyright (c) 2024

6.9 Diagnostics.hpp

Go to the documentation of this file.

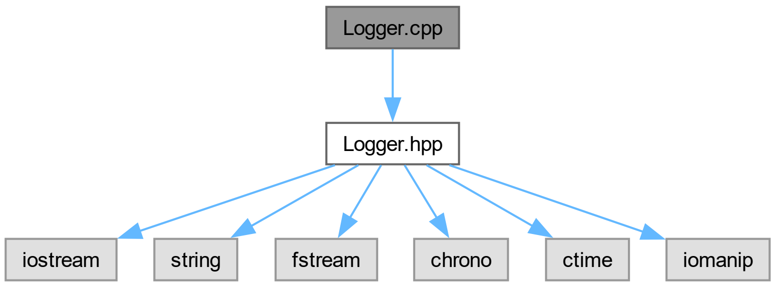
```
#ifndef __DIAGNOSTICS__H__
#define __DIAGNOSTICS__H__

#include "Speed_Sensor.hpp"
#include "Radar_Sensor.hpp"
#include "Temp_Sensor.hpp"
#include "Battery_Sensor.hpp"
#include "Logger.hpp"

class Diagnostics
{
private:
    /* Pointers to sensors for monitoring */
    Speed_Sensor* speedSensor;
    Radar_Sensor* radarSensor;
    Temp_Sensor* tempSensor;
    Battery_Sensor* battery;
    Logger* logger;     /* Singleton logger */
public:
    /* Parameterized Constructor */
    Diagnostics(Speed_Sensor* speed, Radar_Sensor* radar, Temp_Sensor* temp, Battery_Sensor* batt);
    /* Default Destructor */
    ~Diagnostics();
    /* Method to run car diagnostics */
    void runDiagnostics();
    /* Check Speed */
    bool checkSpeed();
    /* Check Distance */
    bool checkDistance();
    /* Check Temperature */
    bool checkTemperature();
    /* Check Battery */
    bool checkBattery();
};

#endif


```

6.10 Logger.cpp File Reference

This is a cpp file for Logger class.

`#include "Logger.hpp"`

Include dependency graph for Logger.cpp:



6.10.1 Detailed Description

This is a cpp file for Logger class.

Author

Abdelrahman Abdelhalem (`abdohalem305@gmail.com`)

This class is for logging all warning and events in a log file

Version

0.1

Date

2024-09-27

Copyright

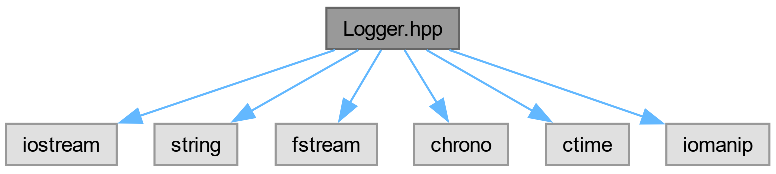
Copyright (c) 2024

6.11 Logger.hpp File Reference

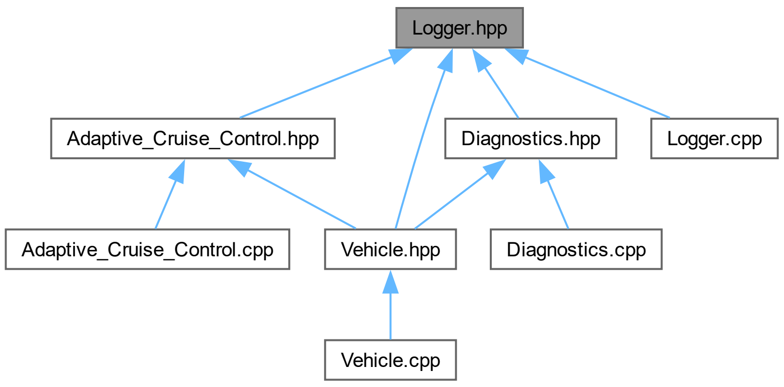
This is a header file for Logger class.

```
#include <iostream>
#include <string>
#include <fstream>
#include <chrono>
#include <ctime>
#include <iomanip>
```

Include dependency graph for Logger.hpp:



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



Classes

- class Logger

  This class is for logging messages.

Enumerations

- enum LogLevel {
  DEBUG , **INFO** , **WARNING** , **ERROR** ,
  CRITICAL }

  Enum to determine the type of log message.
- enum OutputDestination { **FILE_ONLY** , **CONSOLE_ONLY** , **BOTH** }

  Enum to select at which destination the messages will be logged.

6.11.1 Detailed Description

This is a header file for Logger class.

Author

Abdelrahman Abdelhalem (abdohalem305@gmail.com)

This class is for logging all warning and events in a log file

Version

0.1

Date

2024-09-27

Copyright

Copyright (c) 2024

6.12 Logger.hpp

Go to the documentation of this file.

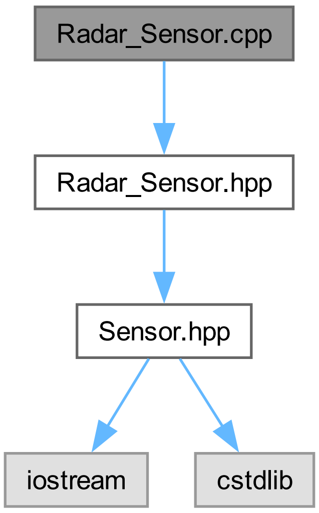
```
#ifndef __LOGGER__H__
#define __LOGGER__H__

#include <iostream>
#include <string>
#include <fstream>  /* std::ofstream */
#include <chrono>   /* std::chrono::system_clock */
#include <ctime>    /* time() */
#include <iomanip>  /* std::put_time */

enum LogLevel { DEBUG, INFO, WARNING, ERROR, CRITICAL };

enum OutputDestination { FILE_ONLY, CONSOLE_ONLY, BOTH };

class Logger
{
private:
    static Logger *instance_;
    std::ofstream logFile_;
    /* Private constructor to prevent direct instantiation */
    Logger() ;
    /* Prevent copy constructor and assignment operator to be used */
    Logger(const Logger&) = delete;
    Logger& operator=(const Logger&) = delete;
public:
    /* Static method to get the singleton instance */
    static Logger* getInstance();
    /* Public method to log messages */
    void log(const std::string& message, LogLevel level, OutputDestination dest = FILE_ONLY);
    /* Destructor */
    ~Logger();
};

#endif
```

6.13 Radar_Sensor.cpp File Reference

This is a cpp file for Radar_Sensor class.

```
#include "Radar_Sensor.hpp"
```

Include dependency graph for Radar_Sensor.cpp:



6.13.1 Detailed Description

This is a cpp file for Radar_Sensor class.

Author

Abdelrahman Abdelhalem (abdohalem305@gmail.com)

This class provides the main functionality of a sensor that measure the distance between the car and front object

Version

0.1

Date

2024-09-27

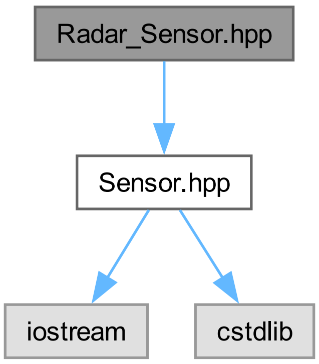
Copyright

Copyright (c) 2024

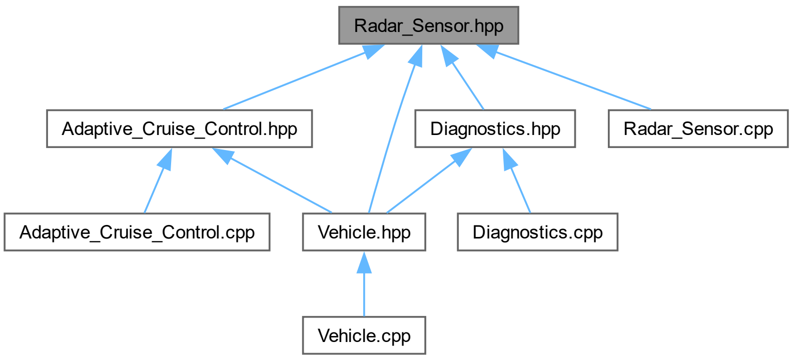
6.14 Radar_Sensor.hpp File Reference

This is a header file for Radar_Sensor class.

`#include "Sensor.hpp"`
Include dependency graph for Radar_Sensor.hpp:



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



Classes

- class Radar_Sensor

  This class is for the radar sensor.

Macros

- #define **SAFE_DISTANCE** (unsigned int)5

  Macro for the distance threshold.

6.14.1 Detailed Description

This is a header file for Radar_Sensor class.

Author

Abdelrahman Abdelhalem (abdohalem305@gmail.com)

This class provides the main functionality of a sensor that measure the distance between the car and front object

Version

0.1

Date

2024-09-27

Copyright

Copyright (c) 2024

6.15 Radar_Sensor.hpp

Go to the documentation of this file.

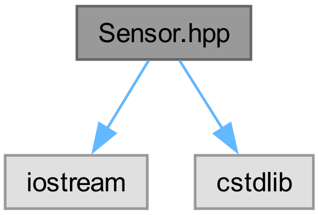
```
#ifndef __RADAR_SENSOR__H__
#define __RADAR_SENSOR__H__

#include "Sensor.hpp"

#define SAFE_DISTANCE           (unsigned int)5

class Radar_Sensor : public Sensor {
private:
    unsigned int distance_;
public:
    /* Member Intialize Constructor */
    Radar_Sensor();
    /* Default Destructor */
    ~Radar_Sensor();
    /* Setters and Getters */
    /* Get the distance value */
    unsigned int getValue() const override;
    /* Update the distance value */
    void update() override;
};


#endif
```

6.16 Sensor.hpp File Reference

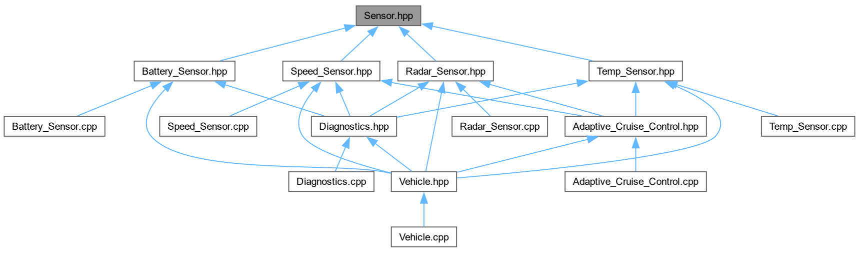
This is a header file for Sensor class.

`#include <iostream>`
`#include <cstdlib>`

Include dependency graph for Sensor.hpp:



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



Classes

- class Sensor

  This is an interface Class provides the main methods inherited in all sensors.

6.16.1 Detailed Description

This is a header file for Sensor class.

Author

Abdelrahman Abdelhalem (`abdohalem305@gmail.com`)

This class provides main methods to be inherited in all sensors

Version

0.1

Date

2024-09-27

Copyright

Copyright (c) 2024

6.17 Sensor.hpp

Go to the documentation of this file.

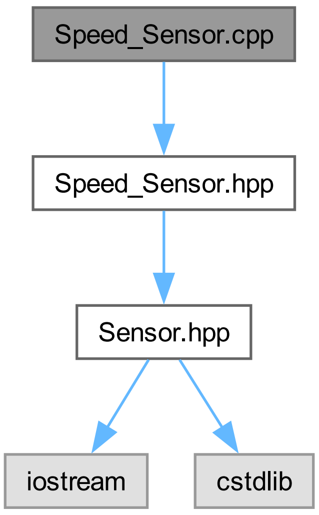
```
#ifndef __SENSOR__H__
#define __SENSOR__H__
#include <iostream>
#include <cstdlib>  // For rand() and srand()
using namespace std;

/* Interface Class for Inheritance */
class Sensor
{
public:
    Sensor() {}
    ~Sensor() {}
    virtual unsigned int getValue() const = 0;
    virtual void update() = 0;
};

#endif

```

6.18 Speed_Sensor.cpp File Reference

This is a cpp file for Speed_Sensor class.

`#include "Speed_Sensor.hpp"`

Include dependency graph for Speed_Sensor.cpp:



6.18.1 Detailed Description

This is a cpp file for Speed_Sensor class.

Author

Abdelrahman Abdelhalem (abdohalem305@gmail.com)

This class provides the main functionality of a sensor that measure the car speed

Version

0.1

Date

2024-09-27

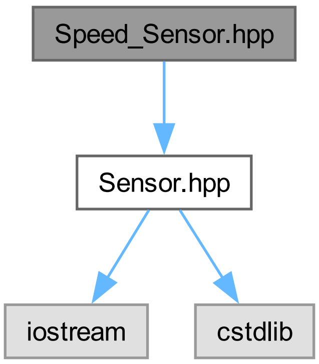
Copyright

Copyright (c) 2024

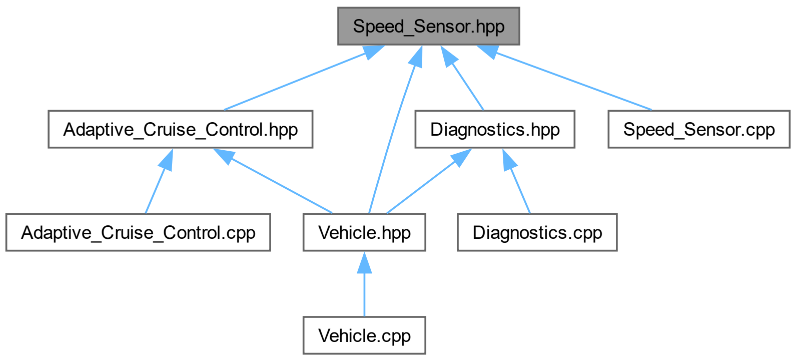
6.19 Speed_Sensor.hpp File Reference

This is a header file for Speed_Sensor class.

```
#include "Sensor.hpp"
```
Include dependency graph for Speed_Sensor.hpp:



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



Classes

- class Speed_Sensor

  This class is for the car's speed sensor.

Macros

- #define **SAFE_SPEED** 150

  Macro for speed threshold.

6.19.1 Detailed Description

This is a header file for Speed_Sensor class.

Author

Abdelrahman Abdelhalem (`abdohalem305@gmail.com`)

This class provides the main functionality of a sensor that measure the car speed

Version

0.1

Date

2024-09-27

Copyright

Copyright (c) 2024

6.20 Speed_Sensor.hpp

Go to the documentation of this file.

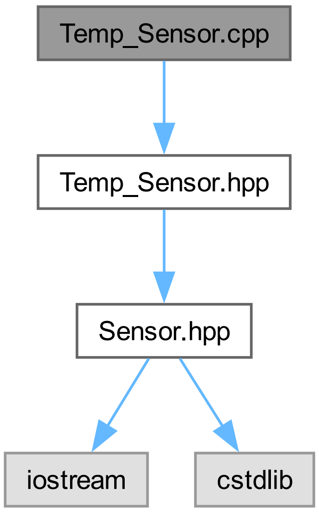
```
#ifndef __SENSORS__H__
#define __SENSORS__H__

#include "Sensor.hpp"

#define SAFE_SPEED 150

class Speed_Sensor : public Sensor {
private:
    unsigned int speed_;    // from 0 to 250 km/m
public:
    /* Member Initialize Constructor */
    Speed_Sensor();
    /* Default Destructor */
    ~Speed_Sensor();
    /* Setters and Getters */
    /* Get the speed value */
    unsigned int getValue() const override;
    /* Update the speed value */
    void update() override;
    /* Control the speed value */
    void controlSpeed(unsigned int speed) ;
};


#endif

```

6.21 Temp_Sensor.cpp File Reference

This is a cpp file for Temp_Sensor class.

```
#include "Temp_Sensor.hpp"
```

Include dependency graph for Temp_Sensor.cpp:



6.21.1 Detailed Description

This is a cpp file for Temp_Sensor class.

Author

Abdelrahman Abdelhalem (`abdohalem305@gmail.com`)

This class provides the main functionality of a temperature sensor

Version

0.1

Date

2024-09-27

Copyright

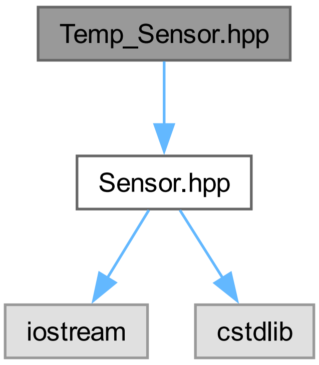
Copyright (c) 2024

6.22 Temp_Sensor.hpp File Reference

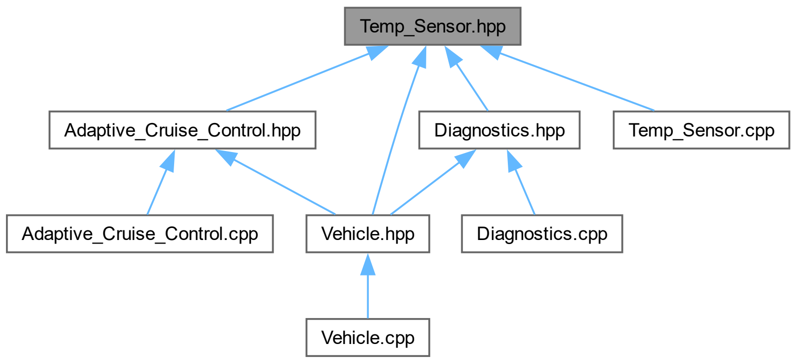
This is a header file for Temp_Sensor class.

```
#include "Sensor.hpp"
```

Include dependency graph for Temp_Sensor.hpp:



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



Classes

- class Temp_Sensor

    This class is for the car's temperature sensor.

Macros

- #define **THRESHOLD_TEMPERATURE** (unsigned int)90

    Macros to the the threshold temperature values.
- #define **SAFE_TEMPERATURE** (unsigned int)70

6.22.1 Detailed Description

This is a header file for Temp_Sensor class.

Author

Abdelrahman Abdelhalem (`abdohalem305@gmail.com`)

This class provides the main functionality of a sensor that measure the car temperature

Version

0.1

Date

2024-09-27

Copyright

Copyright (c) 2024

6.23 Temp_Sensor.hpp

Go to the documentation of this file.

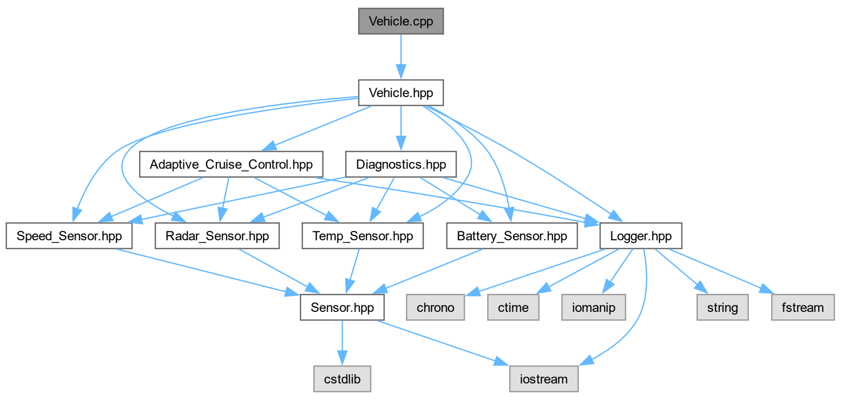
```
#ifndef __TEMP_SENSOR__H__
#define __TEMP_SENSOR__H__
#include "Sensor.hpp"

#define THRESHOLD_TEMPERATURE        (unsigned int)90
#define SAFE_TEMPERATURE             (unsigned int)70

class Temp_Sensor : public Sensor {
private:
    unsigned int temperature_;
public:
    /* Member Initialize Constructor */
    Temp_Sensor();
    /* Default Destructor */
    ~Temp_Sensor();
    /* Setters and Getters */
    /* Get the temperature value */
    unsigned int getValue() const override;
    /* Update the temperature value */
    void update() override;
};

#endif
```

6.24 Vehicle.cpp File Reference

This is a header file for Vehicle class.

`#include "Vehicle.hpp"`

Include dependency graph for Vehicle.cpp:



6.24.1 Detailed Description

This is a header file for Vehicle class.

Author

Abdelrahman Abdelhalem (abdohalem305@gmail.com)

This class is used in the application to update the sensors values, run diagnostics, control the car, and display the dashboard

Version

0.1

Date

2024-09-27

Copyright

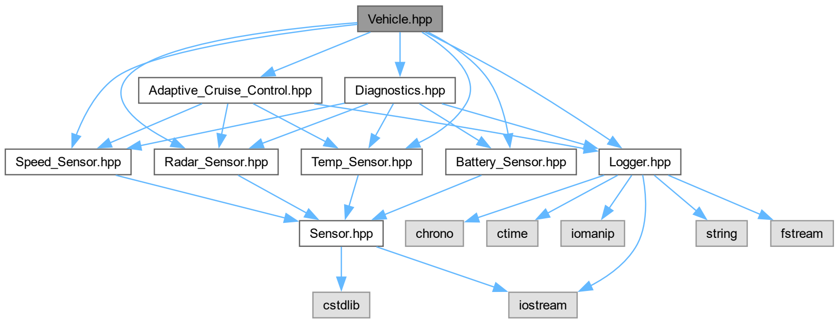
Copyright (c) 2024

6.25 Vehicle.hpp File Reference

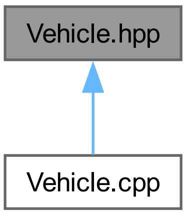
This is a header file for Vehicle class.

```
#include "Speed_Sensor.hpp"
#include "Temp_Sensor.hpp"
#include "Radar_Sensor.hpp"
#include "Battery_Sensor.hpp"
#include "Adaptive_Cruise_Control.hpp"
#include "Diagnostics.hpp"
#include "Logger.hpp"
```

Include dependency graph for Vehicle.hpp:



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



Classes

- class Vehicle

  This is the Vehicle Class.

6.25.1 Detailed Description

This is a header file for Vehicle class.

Author

Abdelrahman Abdelhalem (abdohalem305@gmail.com)

This class is used in the application to update the sensors values, run diagnostics, control the car, and display the dashboard

Version

0.1

Date

2024-09-27

Copyright

Copyright (c) 2024

6.26 Vehicle.hpp

Go to the documentation of this file.

```
#ifndef __VEHICLE__H__
#define __VEHICLE__H__

#include "Speed_Sensor.hpp"
#include "Temp_Sensor.hpp"
#include "Radar_Sensor.hpp"
#include "Battery_Sensor.hpp"
#include "Adaptive_Cruise_Control.hpp"
#include "Diagnostics.hpp"
#include "Logger.hpp"

/* Vehicle Class */
class Vehicle {
    /* Sensors */
    Speed_Sensor speed_sensor;
    Radar_Sensor radar_sensor;
    Temp_Sensor temp_sensor;
    Battery_Sensor battery;
    /* ECUs and Diagnostics */
    Adaptive_Cruise_Control cruise_control;
    Diagnostics diagnostics;
    Logger *logger;
public:
    /* Delegation Constructor */
    Vehicle();
    /* Default Destructor */
    ~Vehicle();
    /* Update sensor readings */
    void updateSensors();
    /* Display the vehicle's current status */
    void displayDashboard();
    /* Run diagnostics to check vehicle status */
    void runDiagnostics();
    /* Control the vehicle behavior (e.g., Adaptive Cruise Control) */
    void controlVehicle();
};

#endif
```

Index