

My Project

Generated by Doxygen 1.12.0

1 Vehicle Telemetry and Diagnostic System Simulation Project	1
1.1 Project Description:	1
1.2 Sensors:	1
1.3 ECUs:	1
1.4 Diagnostics:	1
1.5 Logger:	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Class Documentation	9
5.1 Adaptive_Cruise_Control Class Reference	9
5.1.1 Detailed Description	9
5.1.2 Constructor & Destructor Documentation	10
5.1.2.1 Adaptive_Cruise_Control()	10
5.1.2.2 ~Adaptive_Cruise_Control()	10
5.1.3 Member Function Documentation	10
5.1.3.1 cruiseControl()	10
5.2 Battery_Sensor Class Reference	11
5.2.1 Detailed Description	12
5.2.2 Constructor & Destructor Documentation	13
5.2.2.1 Battery_Sensor()	13
5.2.2.2 ~Battery_Sensor()	13
5.2.3 Member Function Documentation	13
5.2.3.1 getBatteryTemp()	13
5.2.3.2 getValue()	13
5.2.3.3 update()	13
5.3 Diagnostics Class Reference	14
5.3.1 Detailed Description	14
5.3.2 Constructor & Destructor Documentation	15
5.3.2.1 Diagnostics()	15
5.3.2.2 ~Diagnostics()	15
5.3.3 Member Function Documentation	15
5.3.3.1 checkBattery()	15
5.3.3.2 checkDistance()	16
5.3.3.3 checkSpeed()	16
5.3.3.4 checkTemperature()	16
5.3.3.5 runDiagnostics()	16

5.4 Logger Class Reference	17
5.4.1 Detailed Description	17
5.4.2 Constructor & Destructor Documentation	17
5.4.2.1 ~Logger()	17
5.4.3 Member Function Documentation	18
5.4.3.1 getInstance()	18
5.4.3.2 log()	18
5.5 Radar_Sensor Class Reference	19
5.5.1 Detailed Description	20
5.5.2 Constructor & Destructor Documentation	21
5.5.2.1 Radar_Sensor()	21
5.5.2.2 ~Radar_Sensor()	21
5.5.3 Member Function Documentation	21
5.5.3.1 getValue()	21
5.5.3.2 update()	21
5.6 Sensor Class Reference	22
5.6.1 Detailed Description	23
5.6.2 Constructor & Destructor Documentation	23
5.6.2.1 Sensor()	23
5.6.2.2 ~Sensor()	23
5.6.3 Member Function Documentation	23
5.6.3.1 getValue()	23
5.6.3.2 update()	23
5.7 Speed_Sensor Class Reference	24
5.7.1 Detailed Description	25
5.7.2 Constructor & Destructor Documentation	26
5.7.2.1 Speed_Sensor()	26
5.7.2.2 ~Speed_Sensor()	26
5.7.3 Member Function Documentation	26
5.7.3.1 controlSpeed()	26
5.7.3.2 getValue()	26
5.7.3.3 update()	27
5.8 Temp_Sensor Class Reference	27
5.8.1 Detailed Description	28
5.8.2 Constructor & Destructor Documentation	29
5.8.2.1 Temp_Sensor()	29
5.8.2.2 ~Temp_Sensor()	29
5.8.3 Member Function Documentation	29
5.8.3.1 getValue()	29
5.8.3.2 update()	29
5.9 Vehicle Class Reference	30
5.9.1 Detailed Description	30

5.9.2 Constructor & Destructor Documentation	31
5.9.2.1 Vehicle()	31
5.9.2.2 ~Vehicle()	31
5.9.3 Member Function Documentation	31
5.9.3.1 controlVehicle()	31
5.9.3.2 displayDashboard()	31
5.9.3.3 runDiagnostics()	31
5.9.3.4 updateSensors()	31
6 File Documentation	33
6.1 Adaptive_Cruise_Control.cpp File Reference	33
6.1.1 Detailed Description	33
6.2 Adaptive_Cruise_Control.hpp File Reference	34
6.2.1 Detailed Description	35
6.3 Adaptive_Cruise_Control.hpp	35
6.4 Battery_Sensor.cpp File Reference	36
6.4.1 Detailed Description	36
6.5 Battery_Sensor.hpp File Reference	37
6.5.1 Detailed Description	38
6.6 Battery_Sensor.hpp	38
6.7 Diagnostics.cpp File Reference	39
6.7.1 Detailed Description	39
6.8 Diagnostics.hpp File Reference	40
6.8.1 Detailed Description	41
6.9 Diagnostics.hpp	41
6.10 Logger.cpp File Reference	42
6.10.1 Detailed Description	42
6.11 Logger.hpp File Reference	43
6.11.1 Detailed Description	44
6.12 Logger.hpp	44
6.13 Radar_Sensor.cpp File Reference	45
6.13.1 Detailed Description	45
6.14 Radar_Sensor.hpp File Reference	46
6.14.1 Detailed Description	47
6.15 Radar_Sensor.hpp	47
6.16 Sensor.hpp File Reference	48
6.16.1 Detailed Description	48
6.17 Sensor.hpp	49
6.18 Speed_Sensor.cpp File Reference	49
6.18.1 Detailed Description	50
6.19 Speed_Sensor.hpp File Reference	50
6.19.1 Detailed Description	51

6.20 Speed_Sensor.hpp	52
6.21 Temp_Sensor.cpp File Reference	52
6.21.1 Detailed Description	53
6.22 Temp_Sensor.hpp File Reference	53
6.22.1 Detailed Description	54
6.23 Temp_Sensor.hpp	55
6.24 Vehicle.cpp File Reference	55
6.24.1 Detailed Description	55
6.25 Vehicle.hpp File Reference	56
6.25.1 Detailed Description	57
6.26 Vehicle.hpp	57
Index	59

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:

Adaptive_Cruise_Control	9
Diagnostics	14
Logger	17
Sensor	22
Battery_Sensor	11
Radar_Sensor	19
Speed_Sensor	24
Temp_Sensor	27
Vehicle	30

Chapter 3

Class Index

3.1 Class List

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

Adaptive_Cruise_Control	
This class is for Adaptive_Cruise_Control	9
Battery_Sensor	
This class is for the car battery	11
Diagnostics	
This class is for car diagnostics	14
Logger	
This class is for logging messages	17
Radar_Sensor	
This class is for the radar sensor	19
Sensor	
This is an interface Class provides the main methods inherited in all sensors	22
Speed_Sensor	
This class is for the car's speed sensor	24
Temp_Sensor	
This class is for the car's temperature sensor	27
Vehicle	
This is the Vehicle Class	30

Chapter 4

File Index

4.1 File List

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

Adaptive_Cruise_Control.cpp	
This is a cpp file for Adaptive_Cruise_Control class	33
Adaptive_Cruise_Control.hpp	
This is a header file for Adaptive_Cruise_Control class	34
Battery_Sensor.cpp	
This is a cpp file for Battery_Sensor class	36
Battery_Sensor.hpp	
This is a header file for Battery_Sensor class	37
Diagnostics.cpp	
This is a cpp file for Diagnostics class	39
Diagnostics.hpp	
This is a header file for Diagnostics class	40
Logger.cpp	
This is a cpp file for Logger class	42
Logger.hpp	
This is a header file for Logger class	43
Radar_Sensor.cpp	
This is a cpp file for Radar_Sensor class	45
Radar_Sensor.hpp	
This is a header file for Radar_Sensor class	46
Sensor.hpp	
This is a header file for Sensor class	48
Speed_Sensor.cpp	
This is a cpp file for Speed_Sensor class	49
Speed_Sensor.hpp	
This is a header file for Speed_Sensor class	50
Temp_Sensor.cpp	
This is a cpp file for Temp_Sensor class	52
Temp_Sensor.hpp	
This is a header file for Temp_Sensor class	53
Vehicle.cpp	
This is a header file for Vehicle class	55
Vehicle.hpp	
This is a header file for Vehicle class	56

Chapter 5

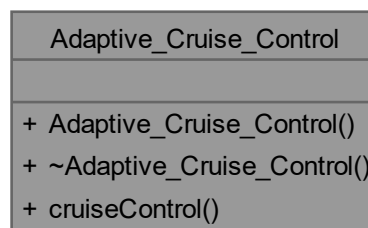
Class Documentation

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

<i>speed_sensor</i>	
<i>radar_sensor</i>	
<i>temp_sensor</i>	

< 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:

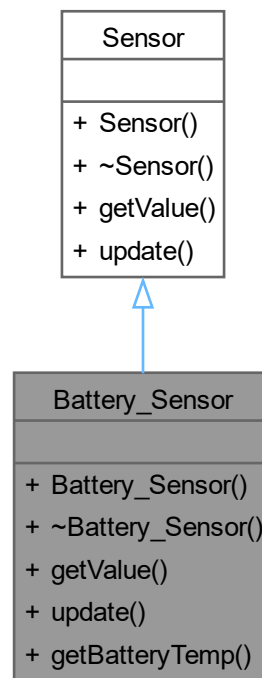
- [Adaptive_Cruise_Control.hpp](#)
- [Adaptive_Cruise_Control.cpp](#)

5.2 Battery_Sensor Class Reference

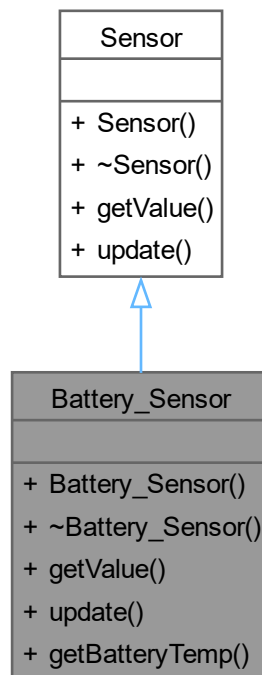
This class is for the car battery.

```
#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:

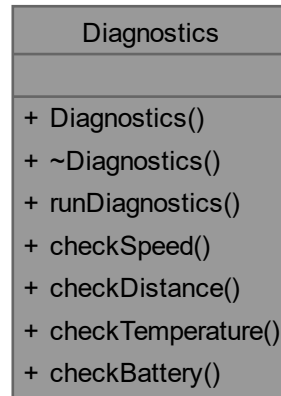
- [Battery_Sensor.hpp](#)
- [Battery_Sensor.cpp](#)

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

<i>speed</i>	Pointer to the Speed_Sensor object
<i>radar</i>	Pointer to the Radar_Sensor object
<i>temp</i>	Pointer to the Temp_Sensor object
<i>batt</i>	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:

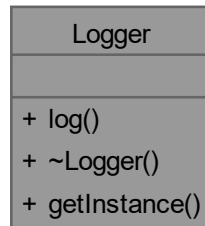
- [Diagnostics.hpp](#)
- [Diagnostics.cpp](#)

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

<i>message</i>	String of the message content
<i>level</i>	Level of the log message
<i>dest</i>	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:

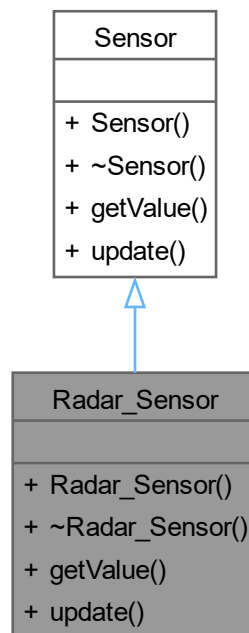
- [Logger.hpp](#)
- [Logger.cpp](#)

5.5 Radar_Sensor Class Reference

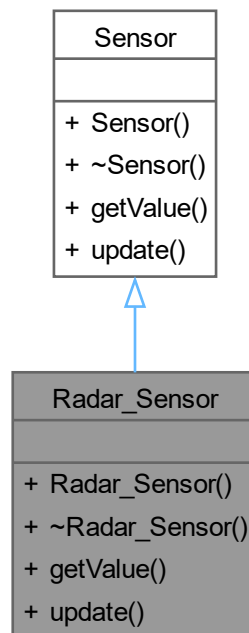
This class is for the radar sensor.

```
#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 Initialize 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:

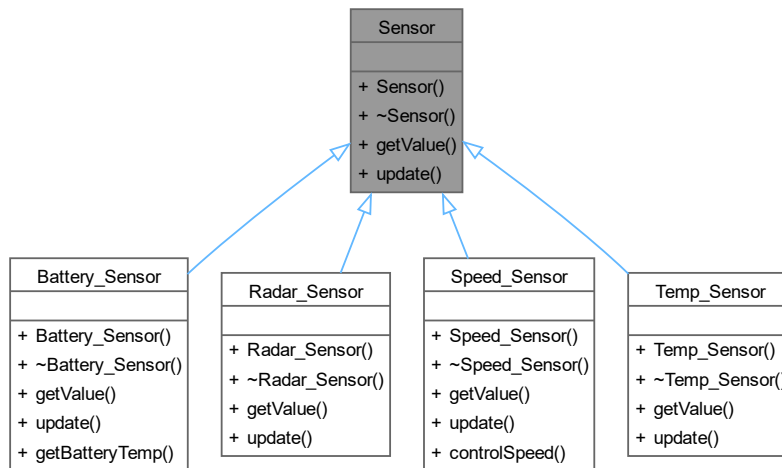
- [Radar_Sensor.hpp](#)
- [Radar_Sensor.cpp](#)

5.6 Sensor Class Reference

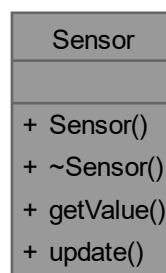
This is an interface Class provides the main methods inherited in all sensors.

```
#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:

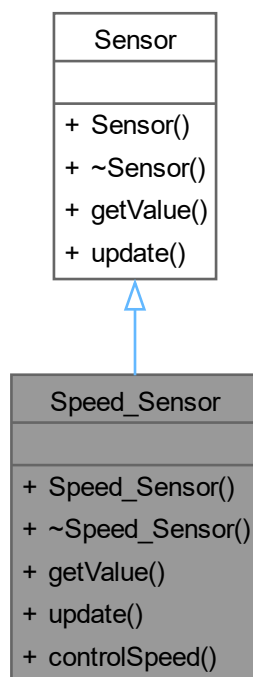
- [Sensor.hpp](#)

5.7 Speed_Sensor Class Reference

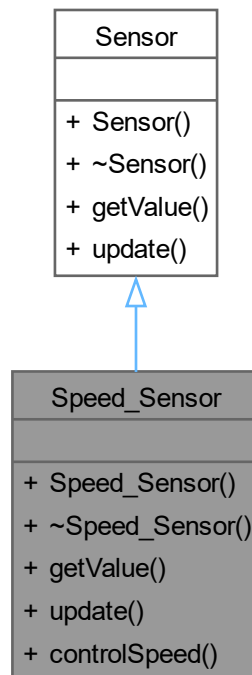
This class is for the car's speed sensor.

```
#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

<i>speed</i>	: 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:

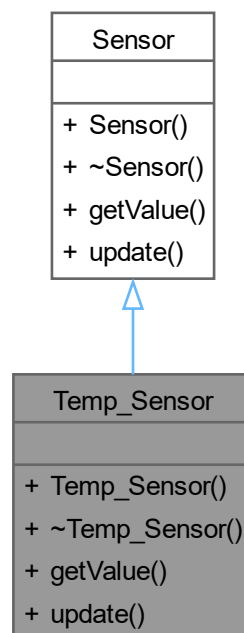
- [Speed_Sensor.hpp](#)
- [Speed_Sensor.cpp](#)

5.8 Temp_Sensor Class Reference

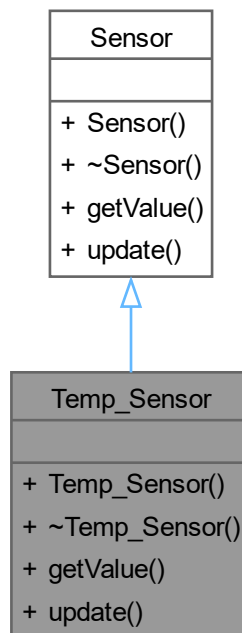
This class is for the car's temperature sensor.

```
#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 celsius.
- [~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:

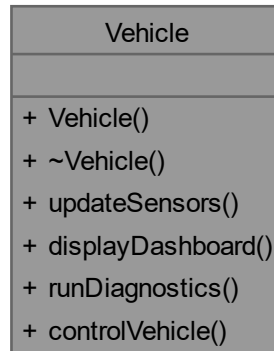
- [Temp_Sensor.hpp](#)
- [Temp_Sensor.cpp](#)

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

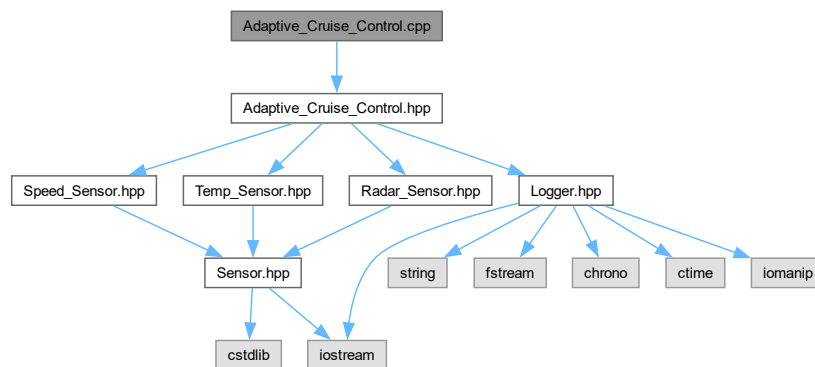
File Documentation

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 occurred

Version

0.1

Date

2024-09-27

Copyright

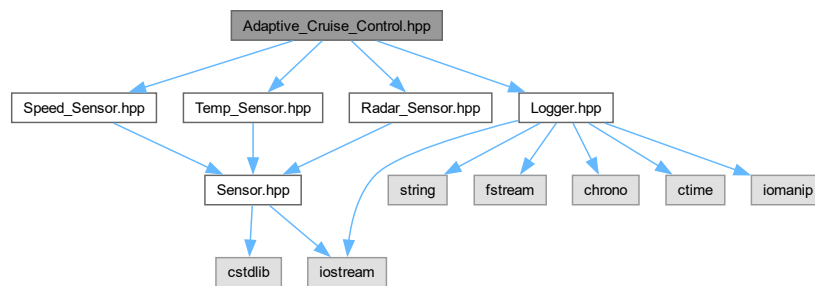
Copyright (c) 2024

6.2 Adaptive_Cruise_Control.hpp File Reference

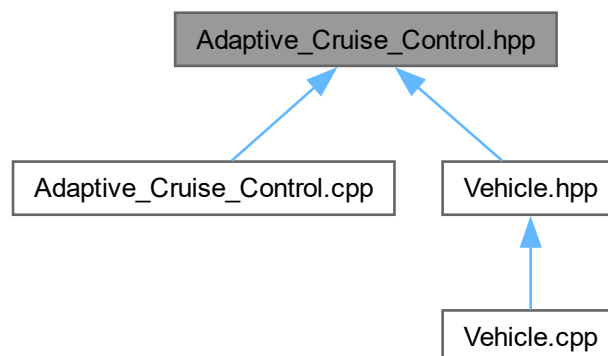
This is a header file for [Adaptive_Cruise_Control](#) class.

```
#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 occurred

Version

0.1

Date

2024-09-27

Copyright

Copyright (c) 2024

6.3 Adaptive_Cruise_Control.hpp

[Go to the documentation of this file.](#)

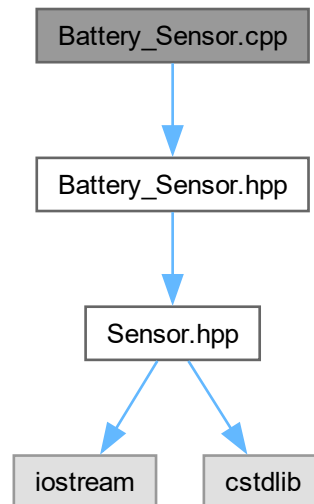
```
00001 #ifndef __ADAPTIVE_CRUISE_CONTROL__H__
00002 #define __ADAPTIVE_CRUISE_CONTROL__H__
00003
00016 #include "Speed_Sensor.hpp"
00017 #include "Temp_Sensor.hpp"
00018 #include "Radar_Sensor.hpp"
00019 #include "Logger.hpp"
00020
00025 #define SLOW_CAR (unsigned int)50
00026 #define STOP_CAR (unsigned int)0
00027
00033 class Adaptive_Cruise_Control
00034 {
00035 private:
00036     Speed_Sensor *speed_sensor;
00037     Radar_Sensor *radar_sensor;
00038     Temp_Sensor *temp_sensor;
00039     Logger *logger;
00040 public:
00041     /* Parameterized Constructor */
00042     Adaptive_Cruise_Control(Speed_Sensor *speed_sensor, Radar_Sensor *radar_sensor, Temp_Sensor
00043                             *temp_sensor) ;
00044     /* Default Destructor */
00045     ~Adaptive_Cruise_Control();
00046     /* Methods */
00047     void cruiseControl();
00048 };
00049
00050 #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

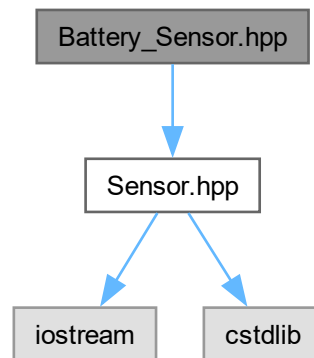
Copyright (c) 2024

6.5 Battery_Sensor.hpp File Reference

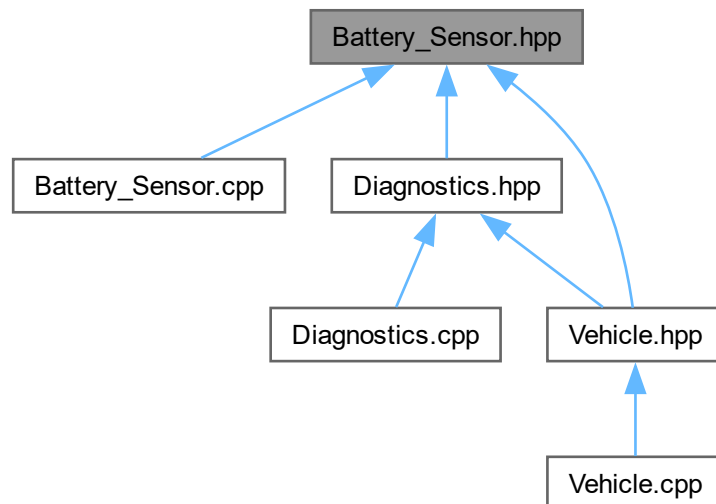
This is a header file for [Battery_Sensor](#) class.

```
#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.](#)

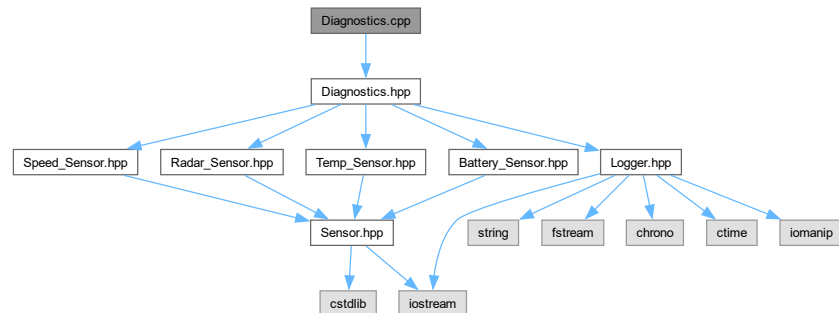
```
00001 #ifndef __BATTERY_SENSOR_H__
00002 #define __BATTERY_SENSOR_H__
00003
00016 #include "Sensor.hpp"
00017
00022 #define THRESHOLD_LEVEL 20
00023
00028 class Battery_Sensor : public Sensor {
00029 private:
00030     unsigned int temperature_;    // from 0 to 50 celesius
00031     unsigned int charge_;        // from 0% to 100%
00032 public:
00034     Battery_Sensor() : temperature_(0), charge_(0) {}
00036     ~Battery_Sensor() {}
00039     unsigned int getValue() const override;
00041     void update() override;
00043     unsigned int getBatteryTemp() const ;
00044 };
00045
00046 #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

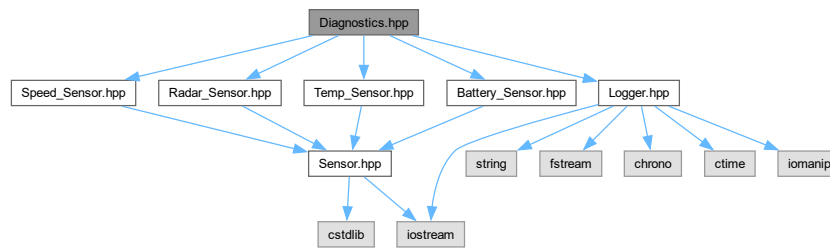
Copyright (c) 2024

6.8 Diagnostics.hpp File Reference

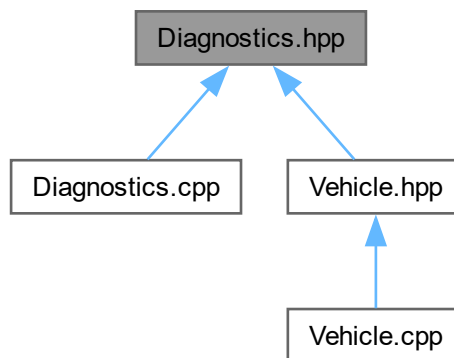
This is a header file for [Diagnostics](#) class.

```
#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.](#)

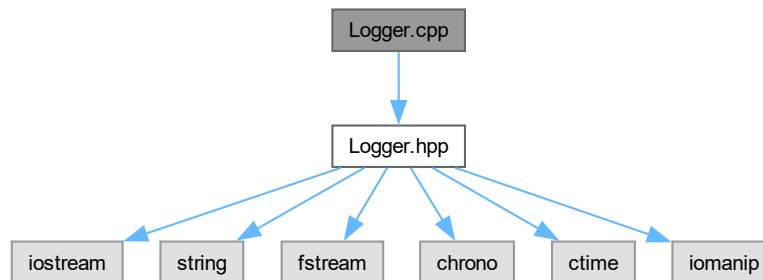
```
00001 #ifndef __DIAGNOSTICS_H__
00002 #define __DIAGNOSTICS_H__
00003
00016 #include "Speed_Sensor.hpp"
00017 #include "Radar_Sensor.hpp"
00018 #include "Temp_Sensor.hpp"
00019 #include "Battery_Sensor.hpp"
00020 #include "Logger.hpp"
00021
00027 class Diagnostics
00028 {
00029 private:
00030     /* Pointers to sensors for monitoring */
00036     Speed_Sensor* speedSensor;
00042     Radar_Sensor* radarSensor;
00048     Temp_Sensor* tempSensor;
00054     Battery_Sensor* battery;
00060     Logger* logger; /* Singleton logger */
00061 public:
00062     /* Parameterized Constructor */
00063     Diagnostics(Speed_Sensor* speed, Radar_Sensor* radar, Temp_Sensor* temp, Battery_Sensor* batt);
00064     /* Default Destructor */
00065     ~Diagnostics();
00066     /* Method to run car diagnostics */
00067     void runDiagnostics();
00068     /* Check Speed */
00069     bool checkSpeed();
00070     /* Check Distance */
00071     bool checkDistance();
00072     /* Check Temperature */
00073     bool checkTemperature();
00074     /* Check Battery */
00075     bool checkBattery();
00076 };
00077
00078 #endif
00079
00080
```

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

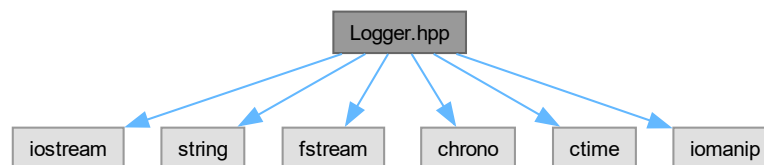
Copyright (c) 2024

6.11 Logger.hpp File Reference

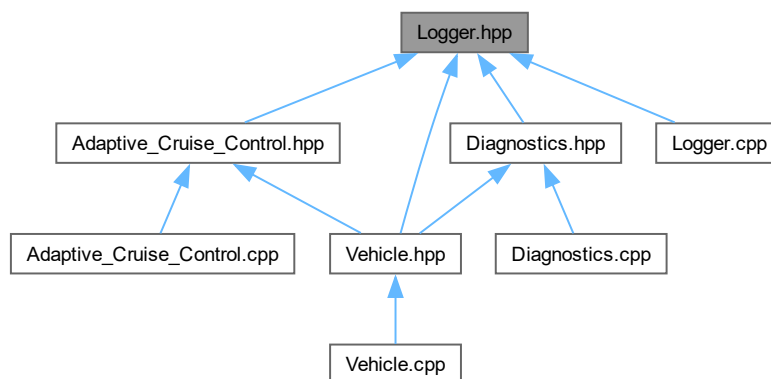
This is a header file for [Logger](#) class.

```
#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.](#)

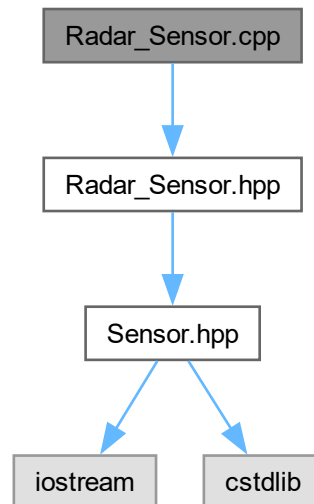
```
00001 #ifndef __LOGGER_H__
00002 #define __LOGGER_H__
00003
00016 #include <iostream>
00017 #include <string>
00018 #include <fstream> /* std::ofstream */
00019 #include <chrono> /* std::chrono::system_clock */
00020 #include <ctime> /* time() */
00021 #include <iomanip> /* std::put_time */
00022
00027 enum LogLevel { DEBUG, INFO, WARNING, ERROR, CRITICAL };
00028
00033 enum OutputDestination { FILE_ONLY, CONSOLE_ONLY, BOTH };
00034
00040 class Logger
00041 {
00042 private:
00048     static Logger *instance_;
00054     std::ofstream logFile_;
00056     /* Private constructor to prevent direct instantiation */
00057     Logger() ;
00058     /* Prevent copy constructor and assignment operator to be used */
00059     Logger(const Logger&) = delete;
00060     Logger& operator=(const Logger&) = delete;
00061 public:
00062     /* Static method to get the singleton instance */
00063     static Logger* getInstance();
00064     /* Public method to log messages */
00065     void log(const std::string& message, LogLevel level, OutputDestination dest = FILE_ONLY);
00066     /* Destructor */
00067     ~Logger();
00068 };
00069
00070 #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

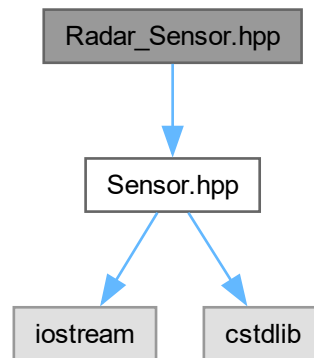
Copyright (c) 2024

6.14 Radar_Sensor.hpp File Reference

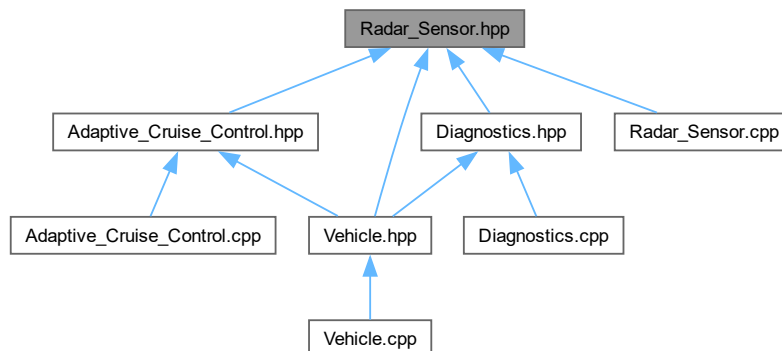
This is a header file for [Radar_Sensor](#) class.

```
#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.](#)

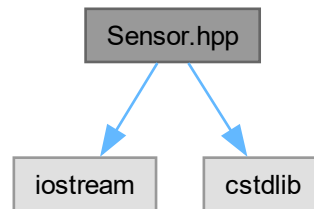
```
00001 #ifndef __RADAR_SENSOR__H__
00002 #define __RADAR_SENSOR__H__
00003
00017 #include "Sensor.hpp"
00018
00023 #define SAFE_DISTANCE (unsigned int)5
00024
00029 class Radar_Sensor : public Sensor {
00030 private:
00032     unsigned int distance_;
00033 public:
00034     /* Member Intialize Constructor */
00035     Radar_Sensor();
00036     /* Default Destructor */
00037     ~Radar_Sensor();
00038     /* Setters and Getters */
00039     /* Get the distance value */
00040     unsigned int getValue() const override;
00041     /* Update the distance value */
00042     void update() override;
00043 };
00044
00045
00046 #endif
```

6.16 Sensor.hpp File Reference

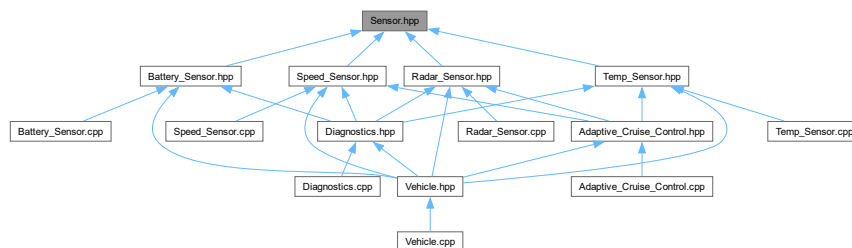
This is a header file for [Sensor](#) class.

```
#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.](#)

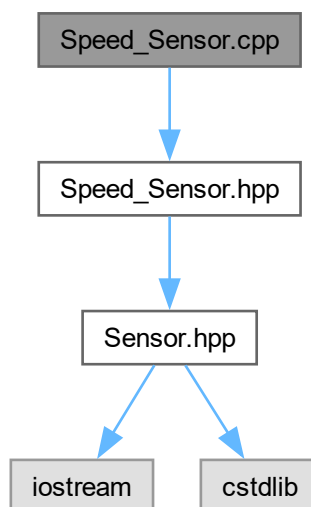
```
00001 #ifndef __SENSOR__H__
00002 #define __SENSOR__H__
00015 #include <iostream>
00016 #include <cstdlib> // For rand() and srand()
00017 using namespace std;
00018
00019 /* Interface Class for Inheritance */
00024 class Sensor
00025 {
00026 public:
00028     Sensor() {}
00030     ~Sensor() {}
00032     virtual unsigned int getValue() const = 0;
00033     virtual void update() = 0;
00034 };
00035
00036 #endif
00037
```

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

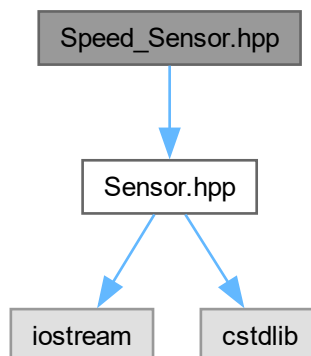
Copyright (c) 2024

6.19 Speed_Sensor.hpp File Reference

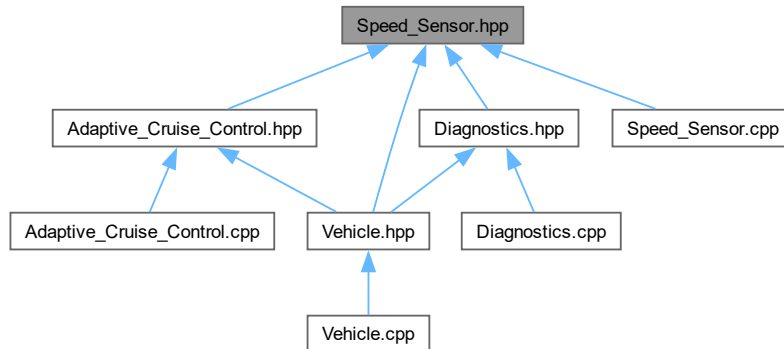
This is a header file for [Speed_Sensor](#) class.

```
#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.](#)

```

00001 #ifndef __SENSORS__H__
00002 #define __SENSORS__H__
00003
00016 #include "Sensor.hpp"
00017
00022 #define SAFE_SPEED 150
00023
00028 class Speed_Sensor : public Sensor {
00029 private:
00030     unsigned int speed_;    // from 0 to 250 km/m
00031 public:
00032     /* Member Initialize Constructor */
00033     Speed_Sensor();
00034     /* Default Destructor */
00035     ~Speed_Sensor();
00036     /* Setters and Getters */
00037     /* Get the speed value */
00038     unsigned int getValue() const override;
00039     /* Update the speed value */
00040     void update() override;
00041     /* Control the speed value */
00042     void controlSpeed(unsigned int speed) ;
00043 };
00044
00045
00046 #endif
00047

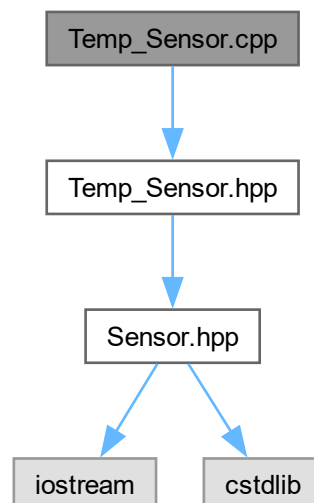
```

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

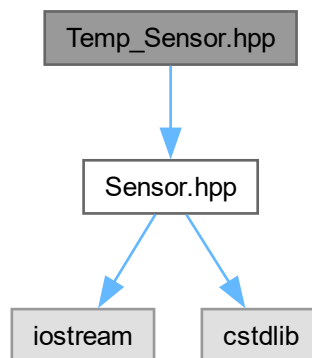
Copyright (c) 2024

6.22 Temp_Sensor.hpp File Reference

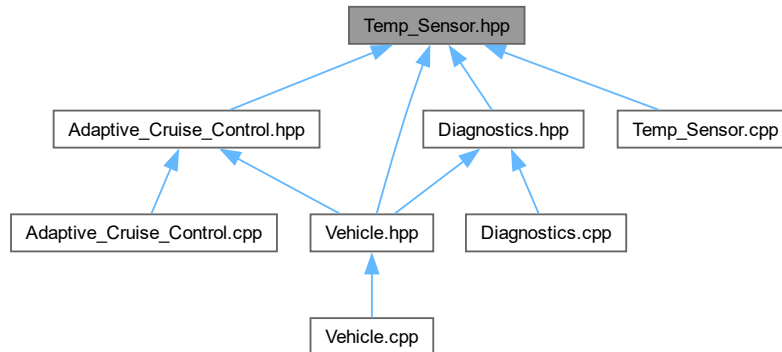
This is a header file for [Temp_Sensor](#) class.

```
#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.](#)

```

00001 #ifndef __TEMP_SENSOR_H__
00002 #define __TEMP_SENSOR_H__
00015 #include "Sensor.hpp"
00016
00021 #define THRESHOLD_TEMPERATURE (unsigned int)90
00022 #define SAFE_TEMPERATURE (unsigned int)70
00023
00028 class Temp_Sensor : public Sensor {
00029 private:
00030     unsigned int temperature_;
00031 public:
00032     /* Member Initialize Constructor */
00033     Temp_Sensor();
00034     /* Default Destructor */
00035     ~Temp_Sensor();
00036     /* Setters and Getters */
00037     /* Get the temperature value */
00038     unsigned int getValue() const override;
00039     /* Update the temperature value */
00040     void update() override;
00041 };
00042
00043 #endif

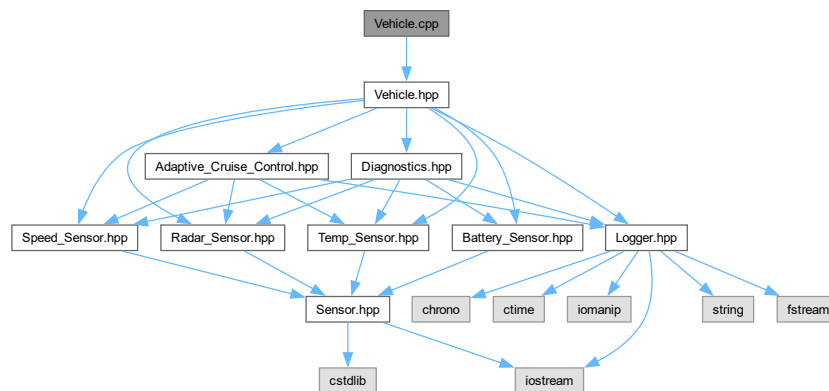
```

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

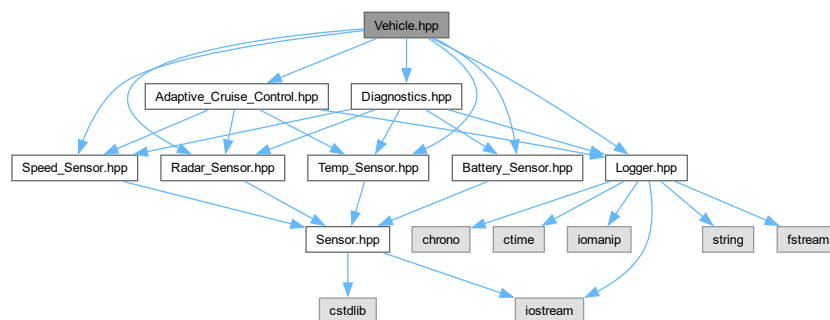
Copyright (c) 2024

6.25 Vehicle.hpp File Reference

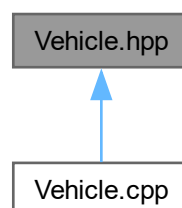
This is a header file for [Vehicle](#) class.

```
#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.](#)

```
00001 #ifndef __VEHICLE__H__
00002 #define __VEHICLE__H__
00003
00017 #include "Speed_Sensor.hpp"
00018 #include "Temp_Sensor.hpp"
00019 #include "Radar_Sensor.hpp"
00020 #include "Battery_Sensor.hpp"
00021 #include "Adaptive_Cruise_Control.hpp"
00022 #include "Diagnostics.hpp"
00023 #include "Logger.hpp"
00024
00025 /* Vehicle Class */
00031 class Vehicle {
00032     /* Sensors */
00038     Speed_Sensor speed_sensor;
00044     Radar_Sensor radar_sensor;
00050     Temp_Sensor temp_sensor;
00056     Battery_Sensor battery;
00057     /* ECUs and Diagnostics */
00063     Adaptive_Cruise_Control cruise_control;
00069     Diagnostics diagnostics;
00075     Logger *logger;
00076 public:
00077     /* Delegation Constructor */
00078     Vehicle();
00079     /* Default Destructor */
00080     ~Vehicle();
00081     /* Update sensor readings */
00082     void updateSensors();
00083     /* Display the vehicle's current status */
00084     void displayDashboard();
00085     /* Run diagnostics to check vehicle status */
00086     void runDiagnostics();
00087     /* Control the vehicle behavior (e.g., Adaptive Cruise Control) */
00088     void controlVehicle();
00089 };
00090
00091 #endif
```


Index

- ~Adaptive_Cruise_Control
 - Adaptive_Cruise_Control, [10](#)
- ~Battery_Sensor
 - Battery_Sensor, [13](#)
- ~Diagnostics
 - Diagnostics, [15](#)
- ~Logger
 - Logger, [17](#)
- ~Radar_Sensor
 - Radar_Sensor, [21](#)
- ~Sensor
 - Sensor, [23](#)
- ~Speed_Sensor
 - Speed_Sensor, [26](#)
- ~Temp_Sensor
 - Temp_Sensor, [29](#)
- ~Vehicle
 - Vehicle, [31](#)
- Adaptive_Cruise_Control, [9](#)
 - ~Adaptive_Cruise_Control, [10](#)
 - Adaptive_Cruise_Control, [10](#)
 - cruiseControl, [10](#)
- Adaptive_Cruise_Control.cpp, [33](#)
- Adaptive_Cruise_Control.hpp, [34](#)
- Battery_Sensor, [11](#)
 - ~Battery_Sensor, [13](#)
 - Battery_Sensor, [13](#)
 - getBatteryTemp, [13](#)
 - getValue, [13](#)
 - update, [13](#)
- Battery_Sensor.cpp, [36](#)
- Battery_Sensor.hpp, [37](#)
- checkBattery
 - Diagnostics, [15](#)
- checkDistance
 - Diagnostics, [15](#)
- checkSpeed
 - Diagnostics, [16](#)
- checkTemperature
 - Diagnostics, [16](#)
- controlSpeed
 - Speed_Sensor, [26](#)
- controlVehicle
 - Vehicle, [31](#)
- cruiseControl
 - Adaptive_Cruise_Control, [10](#)
- Diagnostics, [14](#)
 - ~Diagnostics, [15](#)
 - checkBattery, [15](#)
 - checkDistance, [15](#)
 - checkSpeed, [16](#)
 - checkTemperature, [16](#)
 - Diagnostics, [15](#)
 - runDiagnostics, [16](#)
- Diagnostics.cpp, [39](#)
- Diagnostics.hpp, [40](#)
- displayDashboard
 - Vehicle, [31](#)
- getBatteryTemp
 - Battery_Sensor, [13](#)
- getInstance
 - Logger, [18](#)
- getValue
 - Battery_Sensor, [13](#)
 - Radar_Sensor, [21](#)
 - Sensor, [23](#)
 - Speed_Sensor, [26](#)
 - Temp_Sensor, [29](#)
- log
 - Logger, [18](#)
- Logger, [17](#)
 - ~Logger, [17](#)
 - getInstance, [18](#)
 - log, [18](#)
- Logger.cpp, [42](#)
- Logger.hpp, [43](#)
- Radar_Sensor, [19](#)
 - ~Radar_Sensor, [21](#)
 - getValue, [21](#)
 - Radar_Sensor, [21](#)
 - update, [21](#)
- Radar_Sensor.cpp, [45](#)
- Radar_Sensor.hpp, [46](#)
- runDiagnostics
 - Diagnostics, [16](#)
 - Vehicle, [31](#)
- Sensor, [22](#)
 - ~Sensor, [23](#)
 - getValue, [23](#)
 - Sensor, [23](#)
 - update, [23](#)
- Sensor.hpp, [48](#)
- Speed_Sensor, [24](#)

- [~Speed_Sensor](#), 26
 - [controlSpeed](#), 26
 - [getValue](#), 26
 - [Speed_Sensor](#), 26
 - [update](#), 26
- [Speed_Sensor.cpp](#), 49
- [Speed_Sensor.hpp](#), 50
- [Temp_Sensor](#), 27
 - [~Temp_Sensor](#), 29
 - [getValue](#), 29
 - [Temp_Sensor](#), 29
 - [update](#), 29
- [Temp_Sensor.cpp](#), 52
- [Temp_Sensor.hpp](#), 53
- [update](#)
 - [Battery_Sensor](#), 13
 - [Radar_Sensor](#), 21
 - [Sensor](#), 23
 - [Speed_Sensor](#), 26
 - [Temp_Sensor](#), 29
- [updateSensors](#)
 - [Vehicle](#), 31
- [Vehicle](#), 30
 - [~Vehicle](#), 31
 - [controlVehicle](#), 31
 - [displayDashboard](#), 31
 - [runDiagnostics](#), 31
 - [updateSensors](#), 31
 - [Vehicle](#), 31
- Vehicle Telemetry and Diagnostic System Simulation
 - Project, 1
- [Vehicle.cpp](#), 55
- [Vehicle.hpp](#), 56