EDP AVS

Generated by Doxygen 1.8.14

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 ObstacleDetection Class Reference

ObstacleDetection object controls the activation patterns of all ObstacleSensor objects and outputs detected obstacle grid references to the navigation system.

```
#include <ObstacleDetection.h>
```

Public Member Functions

- ObstacleDetection ()

  Default constructor - not used currently.
- ObstacleDetection (ObstacleSensor ∗frontSensorPtr, ObstacleSensor ∗leftSensorPtr, ObstacleSensor ∗rightSensorPtr, Navigator ∗navPtr)
- void detectAllSensors ()
- void odsToNavTestObstacles ()

  Filled with dummy obstacles to be outputted to navigation system for testing of OD-NM interface.

Private Member Functions

- uint8_t detectLeftSensor ()
- uint8_t detectRightSensor ()
- uint8_t detectFrontSensor ()

Private Attributes

- ObstacleSensor ∗ frontSensorPtr_

  Pointer to front sensor.
- ObstacleSensor ∗ leftSensorPtr_

  Pointer to left sensor.
- ObstacleSensor ∗ rightSensorPtr_

  Pointer to right sensor.
- Navigator ∗ navPtr_

  Pointer to the navigation system object.

Static Private Attributes

- static const uint8_t iterations = 5

  Number of times the sensors are activated to average distance of detection.

3.1.1 Detailed Description

ObstacleDetection object controls the activation patterns of all ObstacleSensor objects and outputs detected obstacle grid references to the navigation system.

A single ObstacleDetection object is used per AVS unit, and it contains all of the details of the specific respective ObstacleSensor objects. The ObstacleDetection object has at its disposal a number of ObstacleSensor activation patterns based on the needs of the particular environment and AVS configuration. INPUTS: Requires information detailing ObstacleSensor objects OUTPUTS TO: Grid reference to navigation system

3.1.2 Constructor & Destructor Documentation

3.1.2.1 ObstacleDetection() [1/2]

```
ObstacleDetection::ObstacleDetection ( )
```

Default constructor - not used currently.

3.1.2.2 ObstacleDetection() [2/2]

```
ObstacleDetection::ObstacleDetection (
              ObstacleSensor * frontSensorPtr,
              ObstacleSensor * leftSensorPtr,
              ObstacleSensor * rightSensorPtr,
              Navigator * navPtr )
```

Constructor which takes in three ObstacleSensor object pointers and navigator object pointer used to activate in different sequences as required

Parameters

∗*frontSensorPtr*	Pointer to front sensor Obstacle Sensor Object
∗*leftSensororPtr*	Pointer to left sensor Obstacle Sensor Object
∗*rightSensorPtr*	Pointer to right sensor Obstacle Sensor Object
∗*navPtr*	Pointer to Navigator object in order to call addObstacle function and pass obstalce details to navigation system

3.1.3 Member Function Documentation

3.1.3.1 detectAllSensors()

```
void ObstacleDetection::detectAllSensors ( )
```

Standard detection function which activates all sensors and converts any obstacles to grid references. Will be in loop function so constantly firing.

3.1.3.2 detectFrontSensor()

```
uint8_t ObstacleDetection::detectFrontSensor ( )  [private]
```

Fire front sensor only

Returns

0 or 1 for legal obstacle distance detected

3.1.3.3 detectLeftSensor()

```
uint8_t ObstacleDetection::detectLeftSensor ( )  [private]
```

Fire left sensor only

Returns

0 or 1 for legal obstacle distance detected

3.1.3.4 detectRightSensor()

```
uint8_t ObstacleDetection::detectRightSensor ( )  [private]
```

Fire right sensor only

Returns

0 or 1 for legal obstacle distance detected

3.1.3.5 odsToNavTestObstacles()

```
void ObstacleDetection::odsToNavTestObstacles ( )
```

Filled with dummy obstacles to be outputted to navigation system for testing of OD-NM interface.

3.1.4 Member Data Documentation

3.1.4.1 frontSensorPtr_

```
ObstacleSensor* ObstacleDetection::frontSensorPtr_ [private]
```

Pointer to front sensor.

3.1.4.2 iterations

```
const uint8_t ObstacleDetection::iterations = 5 [static], [private]
```

Number of times the sensors are activated to average distance of detection.

3.1.4.3 leftSensorPtr_

```
ObstacleSensor* ObstacleDetection::leftSensorPtr_ [private]
```

Pointer to left sensor.

3.1.4.4 navPtr_

```
Navigator* ObstacleDetection::navPtr_ [private]
```

Pointer to the navigation system object.

3.1.4.5 rightSensorPtr_

```
ObstacleSensor* ObstacleDetection::rightSensorPtr_ [private]
```

Pointer to right sensor.

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

- D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAutoCar/bot↩ Main/ObstacleDetection.h
- D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAutoCar/bot↩ Main/ObstacleDetection.cpp

3.2 ObstacleSensor Class Reference

ObstacleSensor object contains all angles and distances/locations of a particular sensor allowing for distance calculations from the pozyx locator to be conducted.

```
#include <ObstacleSensor.h>
```

Public Member Functions

- ObstacleSensor ()

  Default constructor - not used.
- ObstacleSensor (uint8_t triggerPin, uint8_t echoPin, float offsetX, float offsetY, float sensorAngle)
- uint8_t activateSensor (uint8_t iterations)
- void printDistance (String sensorName)

Static Public Member Functions

- static void calculateSoundCm (uint8_t dhtPin)
- static void printSound (float temp, float hum)
- static void updateOdsData (float x, float y, float heading)

Public Attributes

- float offsetX_

  Locations and directions on vehicle relative to Posyx sensor or center or some other reference.
- float offsetY_
- float sensorAngle_
- float sensorGridAngle_

  Angle of sensor relative to the grid (i.e. sensorAngle_ + heading_)
- int8_t unitVect_ [2]

  Unit vector of vector from sensor to detected obstacle (i.e xComp + yComp)
- float distance_

  Distance from sensor to object detected.
- float objXDist_

  Distances from pozyx to object detected.
- float objYDist_
- float objX_

  Grid coordinates at location object detected.
- float objY_
- int gridX_

  Grid reference of obstacle detected converted from coordinates above.
- int gridY_
- NewPing sonar_

  NewPing object causes HC-SR04 sensor pulses and enable.

Static Public Attributes

- static int8_t bias_ = 17

  Bias for grid reference conversion.

Private Attributes

- uint8_t triggerPin_

  Pin definitions.
- uint8_t echoPin_
- const unsigned int maxDistance_ = 400

  Maximum distance of sensor (i.e. 400cm)
- unsigned long duration_

  Stores First HC-SR04 pulse duration value.
- float objGridRef_ [2]

  objGridRef_[0] = x coordinate, objGridRef_[1] = y coordinate

Static Private Attributes

- static float soundcm_ = 0.0343f

  Stores calculated speed of sound in cm/ms - defaults to speed at 20 degrees Celcius.
- static float heading_

  AVS pozyx heading data.
- static float xPos_

  AVS pozyx x position.
- static float yPos_

  AVS pozyx y position.

3.2.1 Detailed Description

ObstacleSensor object contains all angles and distances/locations of a particular sensor allowing for distance calculations from the pozyx locator to be conducted.

The ObstacleSensor object represents a single sensor on the AVS and as such multiple ObstacleSensor objects will exist in the botMain and ObstacleDetection modules. The ObstacleSensor class controls all of the sensor calculations to determine the distance to an obstacle and its grid reference. Once grid reference is determined the ObstacleDetection object will confirm its accuracy and legality. INPUTS: Still require pozyx heading angle (interface between sensor and pozyx) Humidity and temperature details from DHT-22 for improved accuracy Interfaces with HC-SR04 sensors via NewPing class OUTPUTS TO: Objects used by ObstacleDetection class to output grid references to nav.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 ObstacleSensor() [1/2]

```
ObstacleSensor::ObstacleSensor ( )
```

Default constructor - not used.

3.2.2.2 ObstacleSensor() [2/2]

```
ObstacleSensor::ObstacleSensor (
            uint8_t triggerPin,
            uint8_t echoPin,
            float offsetX,
            float offsetY,
            float sensorAngle )
```

Constructor reads in pin details as well as relative to car location and direction details

Parameters

triggerPin	Arduino pin number of sensor trigger
echoPin	Arduino pin number of sensor echo
offsetX	Left or right offset from pozyx placement on AVS chassis
offsetY	Forward or backward offset from pozyx placement on AVS chassis
sensorAngle	Angle sensor is placed at relative to forward direction of AVS chassis

3.2.3 Member Function Documentation

3.2.3.1 activateSensor()

```
uint8_t ObstacleSensor::activateSensor (
            uint8_t iterations )
```

Calculates x and y components to obstacle found by this sensor averaged iterations times by taking into account the heading of the vehicle, the offset of the sensor relative to the pozyx tag, the angle of the sensor relative to the vehicle and the distance the sensor calculates relative to itself. Returns 0 if no obstacle detected, 1 if obstacle detected

Parameters

iterations	Number of iterations sensor measures distance to calculate average over

Returns

0 or 1 for legal obstacle distance detected

3.2.3.2 calculateSoundCm()

```
void ObstacleSensor::calculateSoundCm (
             uint8_t dhtPin )  [static]
```

Calculates the speed of sound based on humidity and temperature found by DHT22 sensor with paramater dhtPin which is the arduino pin assigned to be the data input of the dht. Will be used in setup function so only accessed once at the start. This could be changed if conditions are expected to vary throughout use time.

Parameters

dhtPin	Arduino connected pin number of DHT sensor

3.2.3.3 printDistance()

```
void ObstacleSensor::printDistance (
             String sensorName )
```

Prints on serial monitor the detected sensor distance, and converted x and y components relative to pozyx system

Parameters

sensorName	Name of sensor to be printed

3.2.3.4 printSound()

```
void ObstacleSensor::printSound (
             float temp,
             float hum )  [static]
```

Parameters

temp	Calculated temperature
hum	Calcualted humidity

3.2.3.5 updateOdsData()

```
static void ObstacleSensor::updateOdsData (
             float x,
             float y,
             float heading )  [inline], [static]
```

Updates all sensor data. Will be in loop function so constnatly firing

Parameters

x	Current X location of pozyx
y	Current Y location of pozyx
heading	Current heading of pozyx

3.2.4 Member Data Documentation

3.2.4.1 bias_

```
int8_t ObstacleSensor::bias_ = 17  [static]
```

Bias for grid reference conversion.

3.2.4.2 distance_

```
float ObstacleSensor::distance_
```

Distance from sensor to object detected.

3.2.4.3 duration_

```
unsigned long ObstacleSensor::duration_  [private]
```

Stores First HC-SR04 pulse duration value.

3.2.4.4 echoPin_

```
uint8_t ObstacleSensor::echoPin_  [private]
```

3.2.4.5 gridX_

```
int ObstacleSensor::gridX_
```

Grid reference of obstacle detected converted from coordinates above.

3.2.4.6 gridY_

```
int ObstacleSensor::gridY_
```

3.2.4.7 heading_

```
float ObstacleSensor::heading_  [static], [private]
```

AVS pozyx heading data.

3.2.4.8 maxDistance_

```
const unsigned int ObstacleSensor::maxDistance_ = 400  [private]
```

Maximum distance of sensor (i.e. 400cm)

3.2.4.9 objGridRef_

```
float ObstacleSensor::objGridRef_[2]  [private]
```

objGridRef_[0] = x coordinate, objGridRef_[1] = y coordinate

3.2.4.10 objX_

```
float ObstacleSensor::objX_
```

Grid coordinates at location object detected.

3.2.4.11 objXDist_

```
float ObstacleSensor::objXDist_
```

Distances from pozyx to object detected.

3.2.4.12 objY_

```
float ObstacleSensor::objY_
```

3.2.4.13 objYDist_

```
float ObstacleSensor::objYDist_
```

3.2.4.14 offsetX_

```
float ObstacleSensor::offsetX_
```

Locations and directions on vehicle relative to Posyx sensor or center or some other reference.

3.2.4.15 offsetY_

```
float ObstacleSensor::offsetY_
```

3.2.4.16 sensorAngle_

```
float ObstacleSensor::sensorAngle_
```

3.2.4.17 sensorGridAngle_

```
float ObstacleSensor::sensorGridAngle_
```

Angle of sensor relative to the grid (i.e. sensorAngle_ + heading_)

3.2.4.18 sonar_

```
NewPing ObstacleSensor::sonar_
```

NewPing object causes HC-SR04 sensor pulses and enable.

3.2.4.19 soundcm_

```
float ObstacleSensor::soundcm_ = 0.0343f  [static], [private]
```

Stores calculated speed of sound in cm/ms - defaults to speed at 20 degrees Celcius.

3.2.4.20 triggerPin_

```
uint8_t ObstacleSensor::triggerPin_  [private]
```

Pin definitions.

3.2.4.21 unitVect_

```
int8_t ObstacleSensor::unitVect_[2]
```

Unit vector of vector from sensor to detected obstacle (i.e xComp + yComp)

3.2.4.22 xPos_

```
float ObstacleSensor::xPos_  [static], [private]
```

AVS pozyx x position.

3.2.4.23 yPos_

```
float ObstacleSensor::yPos_  [static], [private]
```

AVS pozyx y position.

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

- D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAutoCar/botMain/ObstacleSensor.h
- D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAutoCar/botMain/ObstacleSensor.cpp

Chapter 4

File Documentation

4.1 D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_↩ edpAutoCar/botMain/botMain.ino File Reference

Top level file for the AVS which integrates all sub-systems on the arduino.

```
#include "ObstacleSensor.h"
#include "ObstacleDetection.h"
```

Macros

- #define LEFT 270∗(PI/180);

  Left direction relative to front of AVS in radians.
- #define FORWARD 0∗(PI/180);

  Forward direction relative to front of AVS in radians.
- #define RIGHT 90∗(PI/180);

  Right direction relative to front of AVS in radians.
- #define BACKWARD 180∗(PI/180);

  Backward direction relative to front of AVS in radians.
- #define DIAG_FOR_RIGHT 45∗(PI/180);

  Forward right diagonal direction relative to front of AVS in radians.
- #define DIAG_FOR_LEFT 315∗(PI/180);

  Forward diagonal left direction relative to front of AVS in radians.
- #define DIAG_BACK_LEFT 225∗(PI/180);

  Backward diagonal left direction relative to front of AVS in radians.
- #define DIAG_BACK_RIGHT 135∗(PI/180);

  Backward diagonal right direction relative to front of AVS in radians.

Functions

- void setup ()
- void loop ()
- void testBlueToothGrid ()

  fake function - to be deleted - just there to see if nav.grid_ array can be accessed from here

Variables

- float avsHeading_ = 0∗(PI/180)

  Heading - to come from pozyx locatio system.
- float avsX_ = 25
- float avsY_ = 0
- Navigator nav
- const uint8_t frontTriggerPin = 17

  Front sensor arduino trigger pin number.
- const uint8_t frontEchoPin = frontTriggerPin

  Front sensor arduino echo pin number.
- const float frontXOffset = 0

  Front sensor x offset on chassis from pozyx location.
- const float frontYOffset = 0

  Front sensor y offset on chassis from pozyx location.
- const float frontsensorAngle = FORWARD

  Front sensor angle relative to forward of chassis.
- const uint8_t leftTriggerPin = 15

  Left sensor arduino trigger pin number.
- const uint8_t leftEchoPin = leftTriggerPin

  Left sensor arduino echo pin number.
- const float leftXOffset = 0

  Left sensor x offset on chassis from pozyx location.
- const float leftYOffset = 0

  Left sensor y offset on chassis from pozyx location.
- const float leftsensorAngle = FORWARD

  Left sensor angle relative to forward of chassis.
- const uint8_t rightTriggerPin = 19

  Right sensor arduino trigger pin number.
- const uint8_t rightEchoPin = rightTriggerPin

  Right sensor arduino echo pin number.
- const float rightXOffset = 0

  Right sensor x offset on chassis from pozyx location.
- const float rightYOffset = 0

  Right sensor y offset on chassis from pozyx location.
- const float rightsensorAngle = FORWARD

  Right sensor angle relative to forward of chassis.
- const uint8_t dhtPin = 14

  DHT22 Sensor arduino pin number.
- ObstacleSensor frontSensor (frontTriggerPin, frontEchoPin, frontXOffset, frontYOffset, frontsensorAngle)

  Front sensor ObstacleSensor object initialisation - used to detect obstacles at front of AVS.
- ObstacleSensor leftSensor (leftTriggerPin, leftEchoPin, leftXOffset, leftYOffset, leftsensorAngle)

  Left sensor ObstacleSensor object initialisation - used to detect obstacles at left of AVS.
- ObstacleSensor rightSensor (rightTriggerPin, rightEchoPin, rightXOffset, rightYOffset, rightsensorAngle)

  Right sensor ObstacleSensor object initialisation - used to detect obstacles at right of AVS.

4.1.1 Detailed Description

Top level file for the AVS which integrates all sub-systems on the arduino.

The botMain file will contain the initial setup and loop functions along with all sub-system object initialisations including parameter setting. Ideally all of the sub-systems shall implement separate classes to improve modularity and readability of this top file. Requires specific libraries be downloaded using the Include Library function explained as per INSTRUCTIONS below.

4.1.2 Macro Definition Documentation

4.1.2.1 BACKWARD

```
#define BACKWARD 180*(PI/180);
```

Backward direction relative to front of AVS in radians.

4.1.2.2 DIAG_BACK_LEFT

```
#define DIAG_BACK_LEFT 225*(PI/180);
```

Backward diagonal left direction relative to front of AVS in radians.

4.1.2.3 DIAG_BACK_RIGHT

```
#define DIAG_BACK_RIGHT 135*(PI/180);
```

Backward diagonal right direction relative to front of AVS in radians.

4.1.2.4 DIAG_FOR_LEFT

```
#define DIAG_FOR_LEFT 315*(PI/180);
```

Forward diagonal left direction relative to front of AVS in radians.

4.1.2.5 DIAG_FOR_RIGHT

```
#define DIAG_FOR_RIGHT 45*(PI/180);
```

Forward right diagonal direction relative to front of AVS in radians.

4.1.2.6 FORWARD

```
#define FORWARD 0*(PI/180);
```

Forward direction relative to front of AVS in radians.

4.1.2.7 LEFT

```
#define LEFT 270*(PI/180);
```

Left direction relative to front of AVS in radians.

4.1.2.8 RIGHT

```
#define RIGHT 90*(PI/180);
```

Right direction relative to front of AVS in radians.

4.1.3 Function Documentation

4.1.3.1 loop()

```
void loop ( )
```

4.1.3.2 setup()

```
void setup ( )
```

4.1.3.3 testBlueToothGrid()

```
void testBlueToothGrid ( )
```

fake function - to be deleted - just there to see if nav.grid_ array can be accessed from here

4.1.4 Variable Documentation

4.1.4.1 avsHeading_

```
float avsHeading_ = 0*(PI/180)
```

Heading - to come from pozyx locatio system.

4.1.4.2 avsX_

```
float avsX_ = 25
```

4.1.4.3 avsY_

```
float avsY_ = 0
```

4.1.4.4 dhtPin

```
const uint8_t dhtPin = 14
```

DHT22 Sensor arduino pin number.

4.1.4.5 frontEchoPin

```
const uint8_t frontEchoPin = frontTriggerPin
```

Front sensor arduino echo pin number.

4.1.4.6 frontSensor

```
ObstacleDetection ods & frontSensor
```

Front sensor ObstacleSensor object initialisation - used to detect obstacles at front of AVS.

Obstacle detection system object initialisation - used to activate ObstacleSensor objects as required and pass on information to the navigation system

4.1.4.7 frontsensorAngle

```
const float frontsensorAngle = FORWARD
```

Front sensor angle relative to forward of chassis.

4.1.4.8 frontTriggerPin

```
const uint8_t frontTriggerPin = 17
```

Front sensor arduino trigger pin number.

4.1.4.9 frontXOffset

```
const float frontXOffset = 0
```

Front sensor x offset on chassis from pozyx location.

4.1.4.10 frontYOffset

```
const float frontYOffset = 0
```

Front sensor y offset on chassis from pozyx location.

4.1.4.11 leftEchoPin

```
const uint8_t leftEchoPin = leftTriggerPin
```

Left sensor arduino echo pin number.

4.1.4.12 leftSensor

```
ObstacleSensor leftSensor(leftTriggerPin, leftEchoPin, leftXOffset, leftYOffset, leftsensorAngle)
```

Left sensor ObstacleSensor object initialisation - used to detect obstacles at left of AVS.

4.1.4.13 leftsensorAngle

```
const float leftsensorAngle = FORWARD
```

Left sensor angle relative to forward of chassis.

4.1.4.14 leftTriggerPin

```
const uint8_t leftTriggerPin = 15
```

Left sensor arduino trigger pin number.

4.1.4.15 leftXOffset

```
const float leftXOffset = 0
```

Left sensor x offset on chassis from pozyx location.

4.1.4.16 leftYOffset

```
const float leftYOffset = 0
```

Left sensor y offset on chassis from pozyx location.

4.1.4.17 nav

```
Navigator nav
```

4.1.4.18 rightEchoPin

```
const uint8_t rightEchoPin = rightTriggerPin
```

Right sensor arduino echo pin number.

4.1.4.19 rightSensor

```
ObstacleSensor rightSensor(rightTriggerPin, rightEchoPin, rightXOffset, rightYOffset, rightsensorAngle)
```

Right sensor ObstacleSensor object initialisation - used to detect obstacles at right of AVS.

4.1.4.20 rightsensorAngle

```
const float rightsensorAngle = FORWARD
```

Right sensor angle relative to forward of chassis.

4.1.4.21 rightTriggerPin

```
const uint8_t rightTriggerPin = 19
```

Right sensor arduino trigger pin number.

4.1.4.22 rightXOffset

```
const float rightXOffset = 0
```

Right sensor x offset on chassis from pozyx location.

4.1.4.23 rightYOffset

```
const float rightYOffset = 0
```

Right sensor y offset on chassis from pozyx location.

4.2 D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_↩ edpAutoCar/botMain/common.h File Reference

Variables

- static const int HEIGHT = 10

  Contains variables common to multiple classes allowing easy sharing of this information rather than passing the variables or doubling up on them.
- static const int WIDTH = 10
- static const int DATA = 3
- const int ELEMENT_XPOS = 0
- const int ELEMENT_YPOS = 1
- const int ELEMENT_VALUE = 2

4.2.1 Variable Documentation

4.2.1.1 DATA

```
const int DATA = 3  [static]
```

4.2.1.2 ELEMENT_VALUE

```
const int ELEMENT_VALUE = 2
```

4.2.1.3 ELEMENT_XPOS

```
const int ELEMENT_XPOS = 0
```

4.2.1.4 ELEMENT_YPOS

```
const int ELEMENT_YPOS = 1
```

4.2.1.5 HEIGHT

```
const int HEIGHT = 10  [static]
```

Contains variables common to multiple classes allowing easy sharing of this information rather than passing the variables or doubling up on them.

4.2.1.6 WIDTH

```
const int WIDTH = 10  [static]
```

4.3 D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_↩ edpAutoCar/botMain/ObstacleDetection.cpp File Reference

```
#include "ObstacleDetection.h"
```

4.4 D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_↩ edpAutoCar/botMain/ObstacleDetection.h File Reference

```
#include "ObstacleSensor.h"
#include <stdint.h>
#include "Navigator.h"
```

Classes

- class ObstacleDetection

  ObstacleDetection object controls the activation patterns of all ObstacleSensor objects and outputs detected obstacle grid references to the navigation system.

4.5 D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_↩ edpAutoCar/botMain/ObstacleSensor.cpp File Reference

```
#include "ObstacleSensor.h"
#include "DHT.h"
```

4.6 D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_↩ edpAutoCar/botMain/ObstacleSensor.h File Reference

```
#include "NewPing.h"
```

Classes

- class ObstacleSensor

  ObstacleSensor object contains all angles and distances/locations of a particular sensor allowing for distance calculations from the pozyx locator to be conducted.

Index