

EDP AVS

Generated by Doxygen 1.8.14



# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class List . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	Navigator Class Reference . . . . .	5
3.1.1	Constructor & Destructor Documentation . . . . .	5
3.1.1.1	Navigator() . . . . .	5
3.1.2	Member Function Documentation . . . . .	6
3.1.2.1	addObstacle() . . . . .	6
3.1.2.2	convertToArray() . . . . .	7
3.1.2.3	createMap() . . . . .	7
3.1.2.4	printMap() . . . . .	7
3.1.2.5	testMap() . . . . .	7
3.1.2.6	testObstacleData() . . . . .	8
3.1.3	Member Data Documentation . . . . .	8
3.1.3.1	grid_ . . . . .	8
3.2	NewPing Class Reference . . . . .	8
3.2.1	Constructor & Destructor Documentation . . . . .	9
3.2.1.1	NewPing() . . . . .	9
3.2.2	Member Function Documentation . . . . .	9
3.2.2.1	check_timer() . . . . .	9

3.2.2.2	<a href="#">convert_cm()</a>	10
3.2.2.3	<a href="#">convert_in()</a>	10
3.2.2.4	<a href="#">ping()</a>	10
3.2.2.5	<a href="#">ping_cm()</a>	10
3.2.2.6	<a href="#">ping_in()</a>	10
3.2.2.7	<a href="#">ping_median()</a>	10
3.2.2.8	<a href="#">ping_timer()</a>	10
3.2.2.9	<a href="#">ping_trigger()</a>	11
3.2.2.10	<a href="#">ping_trigger_timer()</a>	11
3.2.2.11	<a href="#">ping_wait_timer()</a>	11
3.2.2.12	<a href="#">set_max_distance()</a>	11
3.2.2.13	<a href="#">timer_ms()</a>	11
3.2.2.14	<a href="#">timer_ms_cntdown()</a>	11
3.2.2.15	<a href="#">timer_setup()</a>	11
3.2.2.16	<a href="#">timer_stop()</a>	12
3.2.2.17	<a href="#">timer_us()</a>	12
3.2.3	<a href="#">Member Data Documentation</a>	12
3.2.3.1	<a href="#">_echoBit</a>	12
3.2.3.2	<a href="#">_echoInput</a>	12
3.2.3.3	<a href="#">_max_time</a>	12
3.2.3.4	<a href="#">_maxEchoTime</a>	12
3.2.3.5	<a href="#">_triggerBit</a>	12
3.2.3.6	<a href="#">_triggerMode</a>	13
3.2.3.7	<a href="#">_triggerOutput</a>	13
3.2.3.8	<a href="#">ping_result</a>	13
3.3	<a href="#">ObstacleDetection Class Reference</a>	13
3.3.1	<a href="#">Detailed Description</a>	14
3.3.2	<a href="#">Constructor &amp; Destructor Documentation</a>	14
3.3.2.1	<a href="#">ObstacleDetection() [1/2]</a>	14
3.3.2.2	<a href="#">ObstacleDetection() [2/2]</a>	14

3.3.3	Member Function Documentation . . . . .	15
3.3.3.1	detectAllSensors() . . . . .	15
3.3.3.2	detectFrontSensor() . . . . .	15
3.3.3.3	detectLeftSensor() . . . . .	15
3.3.3.4	detectRightSensor() . . . . .	16
3.3.3.5	odsToNavTestObstacles() . . . . .	16
3.3.4	Member Data Documentation . . . . .	16
3.3.4.1	frontSensorPtr_ . . . . .	16
3.3.4.2	iterations . . . . .	16
3.3.4.3	leftSensorPtr_ . . . . .	16
3.3.4.4	navPtr_ . . . . .	17
3.3.4.5	rightSensorPtr_ . . . . .	17
3.4	ObstacleSensor Class Reference . . . . .	17
3.4.1	Detailed Description . . . . .	19
3.4.2	Constructor & Destructor Documentation . . . . .	19
3.4.2.1	ObstacleSensor() [1/2] . . . . .	19
3.4.2.2	ObstacleSensor() [2/2] . . . . .	19
3.4.3	Member Function Documentation . . . . .	19
3.4.3.1	activateSensor() . . . . .	19
3.4.3.2	calculateSoundCm() . . . . .	20
3.4.3.3	printDistance() . . . . .	20
3.4.3.4	printSound() . . . . .	20
3.4.3.5	updateOdsData() . . . . .	21
3.4.4	Member Data Documentation . . . . .	21
3.4.4.1	bias_ . . . . .	21
3.4.4.2	distance_ . . . . .	21
3.4.4.3	duration_ . . . . .	21
3.4.4.4	echoPin_ . . . . .	22
3.4.4.5	gridX_ . . . . .	22
3.4.4.6	gridY_ . . . . .	22

3.4.4.7	heading_ . . . . .	22
3.4.4.8	maxDistance_ . . . . .	22
3.4.4.9	objGridRef_ . . . . .	22
3.4.4.10	objX_ . . . . .	23
3.4.4.11	objXDist_ . . . . .	23
3.4.4.12	objY_ . . . . .	23
3.4.4.13	objYDist_ . . . . .	23
3.4.4.14	offsetX_ . . . . .	23
3.4.4.15	offsetY_ . . . . .	23
3.4.4.16	sensorAngle_ . . . . .	23
3.4.4.17	sensorGridAngle_ . . . . .	24
3.4.4.18	sonar_ . . . . .	24
3.4.4.19	soundcm_ . . . . .	24
3.4.4.20	triggerPin_ . . . . .	24
3.4.4.21	unitVect_ . . . . .	24
3.4.4.22	xPos_ . . . . .	24
3.4.4.23	yPos_ . . . . .	24
<b>4</b>	<b>File Documentation</b>	<b>25</b>
4.1	D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAuto↔ Car/botMain/botMain.ino File Reference . . . . .	25
4.1.1	Macro Definition Documentation . . . . .	26
4.1.1.1	BACKWARD . . . . .	27
4.1.1.2	DIAG_BACK_LEFT . . . . .	27
4.1.1.3	DIAG_FOR_LEFT . . . . .	27
4.1.1.4	DIAG_FOR_RIGHT . . . . .	27
4.1.1.5	FORWARD . . . . .	27
4.1.1.6	LEFT . . . . .	27
4.1.1.7	RIGHT . . . . .	27
4.1.2	Function Documentation . . . . .	27
4.1.2.1	loop() . . . . .	28

4.1.2.2	setup()	28
4.1.2.3	testBlueToothGrid()	28
4.1.3	Variable Documentation	28
4.1.3.1	avsHeading_	28
4.1.3.2	avsX_	28
4.1.3.3	avsY_	28
4.1.3.4	dhtPin	29
4.1.3.5	frontEchoPin	29
4.1.3.6	frontSensor	29
4.1.3.7	frontsensorAngle	29
4.1.3.8	frontTriggerPin	29
4.1.3.9	frontXOffset	29
4.1.3.10	frontYOffset	30
4.1.3.11	leftEchoPin	30
4.1.3.12	leftSensor	30
4.1.3.13	leftsensorAngle	30
4.1.3.14	leftTriggerPin	30
4.1.3.15	leftXOffset	30
4.1.3.16	leftYOffset	31
4.1.3.17	nav	31
4.1.3.18	rightEchoPin	31
4.1.3.19	rightSensor	31
4.1.3.20	rightsensorAngle	31
4.1.3.21	rightTriggerPin	31
4.1.3.22	rightXOffset	32
4.1.3.23	rightYOffset	32
4.2	D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAuto↔ Car/botMain/common.h File Reference	32
4.2.1	Variable Documentation	32
4.2.1.1	DATA	32
4.2.1.2	ELEMENT_VALUE	32

4.2.1.3	ELEMENT_XPOS	32
4.2.1.4	ELEMENT_YPOS	33
4.2.1.5	HEIGHT	33
4.2.1.6	WIDTH	33
4.3	D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAuto↔ Car/botMain/Navigator.cpp File Reference	33
4.4	D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAuto↔ Car/botMain/Navigator.h File Reference	33
4.4.1	Variable Documentation	34
4.4.1.1	DATA	34
4.4.1.2	ELEMENT_VALUE	34
4.4.1.3	ELEMENT_XPOS	34
4.4.1.4	ELEMENT_YPOS	34
4.4.1.5	HEIGHT	34
4.4.1.6	WIDTH	34
4.5	D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAuto↔ Car/botMain/NewPing.cpp File Reference	34
4.5.1	Function Documentation	35
4.5.1.1	ISR()	35
4.5.2	Variable Documentation	35
4.5.2.1	_ms_cnt	35
4.5.2.2	_ms_cnt_reset	35
4.5.2.3	intFunc	35
4.5.2.4	intFunc2	35
4.6	D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAuto↔ Car/botMain/NewPing.h File Reference	36
4.6.1	Macro Definition Documentation	36
4.6.1.1	DO_BITWISE	36
4.6.1.2	ECHO_TIMER_FREQ	36
4.6.1.3	MAX_SENSOR_DELAY	37
4.6.1.4	MAX_SENSOR_DISTANCE	37
4.6.1.5	NewPingConvert	37



4.6.1.6	NO_ECHO . . . . .	37
4.6.1.7	ONE_PIN_ENABLED . . . . .	37
4.6.1.8	PING_MEDIAN_DELAY . . . . .	37
4.6.1.9	PING_OVERHEAD [1/2] . . . . .	37
4.6.1.10	PING_OVERHEAD [2/2] . . . . .	38
4.6.1.11	PING_TIMER_OVERHEAD [1/2] . . . . .	38
4.6.1.12	PING_TIMER_OVERHEAD [2/2] . . . . .	38
4.6.1.13	ROUNDING_ENABLED . . . . .	38
4.6.1.14	TIMER_ENABLED [1/2] . . . . .	38
4.6.1.15	TIMER_ENABLED [2/2] . . . . .	38
4.6.1.16	URM37_ENABLED . . . . .	38
4.6.1.17	US_ROUNDTRIP_CM [1/2] . . . . .	38
4.6.1.18	US_ROUNDTRIP_CM [2/2] . . . . .	39
4.6.1.19	US_ROUNDTRIP_IN [1/2] . . . . .	39
4.6.1.20	US_ROUNDTRIP_IN [2/2] . . . . .	39
4.7	D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAuto↔ Car/botMain/ObstacleDetection.cpp File Reference . . . . .	39
4.8	D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAuto↔ Car/botMain/ObstacleDetection.h File Reference . . . . .	39
4.9	D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAuto↔ Car/botMain/ObstacleSensor.cpp File Reference . . . . .	39
4.10	D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAuto↔ Car/botMain/ObstacleSensor.h File Reference . . . . .	39
<b>Index</b>		<b>41</b>



# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">Navigator</a>	5
<a href="#">NewPing</a>	8
<a href="#">ObstacleDetection</a>	13
<a href="#">ObstacleSensor</a>	17



## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAutoCar/bot↔ Main/ <a href="#">botMain.ino</a> . . . . .	25
D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAutoCar/bot↔ Main/ <a href="#">common.h</a> . . . . .	32
D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAutoCar/bot↔ Main/ <a href="#">Navigator.cpp</a> . . . . .	33
D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAutoCar/bot↔ Main/ <a href="#">Navigator.h</a> . . . . .	33
D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAutoCar/bot↔ Main/ <a href="#">NewPing.cpp</a> . . . . .	34
D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAutoCar/bot↔ Main/ <a href="#">NewPing.h</a> . . . . .	36
D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAutoCar/bot↔ Main/ <a href="#">ObstacleDetection.cpp</a> . . . . .	39
D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAutoCar/bot↔ Main/ <a href="#">ObstacleDetection.h</a> . . . . .	39
D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAutoCar/bot↔ Main/ <a href="#">ObstacleSensor.cpp</a> . . . . .	39
D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAutoCar/bot↔ Main/ <a href="#">ObstacleSensor.h</a> . . . . .	39



## Chapter 3

# Class Documentation

### 3.1 Navigator Class Reference

```
#include <Navigator.h>
```

#### Public Member Functions

- [Navigator](#) ()
- void [addObstacle](#) (int xPos, int yPos)
- void [testMap](#) ()
- void [printMap](#) ()
- void [testObstacleData](#) ()

#### Public Attributes

- int [grid\\_](#) [[HEIGHT](#)+2][[WIDTH](#)+2][[DATA](#)]

#### Private Member Functions

- void [createMap](#) ()
- int [convertToArray](#) (int coord)

#### 3.1.1 Constructor & Destructor Documentation

##### 3.1.1.1 Navigator()

```
Navigator::Navigator ( )
```

## 3.1.2 Member Function Documentation

### 3.1.2.1 addObstacle()

```
void Navigator::addObstacle (
    int xPos,
    int yPos )
```

Adds an obstacle to the map



**Parameters**

<i>xPos</i>	Cartesian x-coordinate of the obstacle grid square
<i>yPos</i>	Cartesian y-coordinate of the obstacle grid square

**3.1.2.2 convertToArray()**

```
int Navigator::convertToArray (
    int coord ) [private]
```

Converts y grid coordinate to y array coordinate

**Parameters**

<i>coord</i>	Grid coordinate
--------------	-----------------

**Returns**

Array coordinate

**3.1.2.3 createMap()**

```
void Navigator::createMap ( ) [private]
```

Creates a map that with a given length, height and a boundary wall of obstacles assigning an initial weighting of -1

**Parameters**

<i>height</i>	Height of the grid to be traversed
<i>length</i>	Length of the grid to be traversed

**3.1.2.4 printMap()**

```
void Navigator::printMap ( )
```

Prints the grid with the associated priority value for each grid square

**3.1.2.5 testMap()**

```
void Navigator::testMap ( )
```

Tests the [createMap\(\)](#) function with a length of 10 and height of 10. Tests the [prioritiseMap\(\)](#) function with a target location at 9, 9

### 3.1.2.6 testObstacleData()

```
void Navigator::testObstacleData ( )
```

Adds obstacles from hardcoded obstacle position data

## 3.1.3 Member Data Documentation

### 3.1.3.1 grid\_

```
int Navigator::grid_[HEIGHT+2][WIDTH+2][DATA]
```

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/[Navigator.h](#)
- D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git\_edpAutoCar/bot↵ Main/[Navigator.cpp](#)

## 3.2 NewPing Class Reference

```
#include <NewPing.h>
```

### Public Member Functions

- [NewPing](#) (uint8\_t trigger\_pin, uint8\_t echo\_pin, unsigned int max\_cm\_distance=[MAX\\_SENSOR\\_DISTANCE](#))
- unsigned int [ping](#) (unsigned int max\_cm\_distance=0)
- unsigned long [ping\\_cm](#) (unsigned int max\_cm\_distance=0)
- unsigned long [ping\\_in](#) (unsigned int max\_cm\_distance=0)
- unsigned long [ping\\_median](#) (uint8\_t it=5, unsigned int max\_cm\_distance=0)
- void [ping\\_timer](#) (void(\*userFunc)(void), unsigned int max\_cm\_distance=0)
- boolean [check\\_timer](#) ()

### Static Public Member Functions

- static unsigned int [convert\\_cm](#) (unsigned int echoTime)
- static unsigned int [convert\\_in](#) (unsigned int echoTime)
- static void [timer\\_us](#) (unsigned int frequency, void(\*userFunc)(void))
- static void [timer\\_ms](#) (unsigned long frequency, void(\*userFunc)(void))
- static void [timer\\_stop](#) ()

### Public Attributes

- unsigned long [ping\\_result](#)

### Private Member Functions

- boolean [ping\\_trigger](#) ()
- void [set\\_max\\_distance](#) (unsigned int max\_cm\_distance)
- boolean [ping\\_trigger\\_timer](#) (unsigned int trigger\_delay)
- boolean [ping\\_wait\\_timer](#) ()

### Static Private Member Functions

- static void [timer\\_setup](#) ()
- static void [timer\\_ms\\_cntdown](#) ()

### Private Attributes

- uint8\_t [\\_triggerBit](#)
- uint8\_t [\\_echoBit](#)
- volatile uint8\_t \* [\\_triggerOutput](#)
- volatile uint8\_t \* [\\_echoInput](#)
- volatile uint8\_t \* [\\_triggerMode](#)
- unsigned int [\\_maxEchoTime](#)
- unsigned long [\\_max\\_time](#)

## 3.2.1 Constructor & Destructor Documentation

### 3.2.1.1 NewPing()

```
NewPing::NewPing (
    uint8_t trigger_pin,
    uint8_t echo_pin,
    unsigned int max_cm_distance = MAX\_SENSOR\_DISTANCE )
```

## 3.2.2 Member Function Documentation

### 3.2.2.1 check\_timer()

```
boolean NewPing::check_timer ( )
```

### 3.2.2.2 convert\_cm()

```
unsigned int NewPing::convert_cm (
    unsigned int echoTime ) [static]
```

### 3.2.2.3 convert\_in()

```
unsigned int NewPing::convert_in (
    unsigned int echoTime ) [static]
```

### 3.2.2.4 ping()

```
unsigned int NewPing::ping (
    unsigned int max_cm_distance = 0 )
```

### 3.2.2.5 ping\_cm()

```
unsigned long NewPing::ping_cm (
    unsigned int max_cm_distance = 0 )
```

### 3.2.2.6 ping\_in()

```
unsigned long NewPing::ping_in (
    unsigned int max_cm_distance = 0 )
```

### 3.2.2.7 ping\_median()

```
unsigned long NewPing::ping_median (
    uint8_t it = 5,
    unsigned int max_cm_distance = 0 )
```

### 3.2.2.8 ping\_timer()

```
void NewPing::ping_timer (
    void(*) (void) userFunc,
    unsigned int max_cm_distance = 0 )
```

### 3.2.2.9 ping\_trigger()

```
boolean NewPing::ping_trigger ( ) [private]
```

### 3.2.2.10 ping\_trigger\_timer()

```
boolean NewPing::ping_trigger_timer (
    unsigned int trigger_delay ) [private]
```

### 3.2.2.11 ping\_wait\_timer()

```
boolean NewPing::ping_wait_timer ( ) [private]
```

### 3.2.2.12 set\_max\_distance()

```
void NewPing::set_max_distance (
    unsigned int max_cm_distance ) [private]
```

### 3.2.2.13 timer\_ms()

```
void NewPing::timer_ms (
    unsigned long frequency,
    void(*) (void) userFunc ) [static]
```

### 3.2.2.14 timer\_ms\_cntdown()

```
void NewPing::timer_ms_cntdown ( ) [static], [private]
```

### 3.2.2.15 timer\_setup()

```
void NewPing::timer_setup ( ) [static], [private]
```

### 3.2.2.16 timer\_stop()

```
void NewPing::timer_stop ( ) [static]
```

### 3.2.2.17 timer\_us()

```
void NewPing::timer_us (
    unsigned int frequency,
    void(*) (void) userFunc ) [static]
```

## 3.2.3 Member Data Documentation

### 3.2.3.1 \_echoBit

```
uint8_t NewPing::_echoBit [private]
```

### 3.2.3.2 \_echoInput

```
volatile uint8_t* NewPing::_echoInput [private]
```

### 3.2.3.3 \_max\_time

```
unsigned long NewPing::_max_time [private]
```

### 3.2.3.4 \_maxEchoTime

```
unsigned int NewPing::_maxEchoTime [private]
```

### 3.2.3.5 \_triggerBit

```
uint8_t NewPing::_triggerBit [private]
```

3.2.3.6 `_triggerMode`

```
volatile uint8_t* NewPing::_triggerMode [private]
```

3.2.3.7 `_triggerOutput`

```
volatile uint8_t* NewPing::_triggerOutput [private]
```

3.2.3.8 `ping_result`

```
unsigned long NewPing::ping_result
```

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/[NewPing.h](#)
- D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git\_edpAutoCar/bot↔ Main/[NewPing.cpp](#)

## 3.3 ObstacleDetection Class Reference

```
#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.*

### 3.3.1 Detailed Description

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

INPUTS: Requires information detailing [ObstacleSensor](#) objects

OUTPUTS TO: Grid reference to navigation system

### 3.3.2 Constructor & Destructor Documentation

#### 3.3.2.1 [ObstacleDetection\(\)](#) [1/2]

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

Default constructor - not used currently.

#### 3.3.2.2 [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

<i>*frontSensorPtr</i>	Pointer to front sensor Obstacle Sensor Object
<i>*leftSensororPtr</i>	Pointer to left sensor Obstacle Sensor Object
<i>*rightSensorPtr</i>	Pointer to right sensor Obstacle Sensor Object
<i>*navPtr</i>	Pointer to <a href="#">Navigator</a> object in order to call addObstacle function and pass obstacle details to navigation system

## 3.3.3 Member Function Documentation

## 3.3.3.1 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.

## 3.3.3.2 detectFrontSensor()

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

Fire front sensor only

## Returns

0 or 1 for legal obstacle distance detected

## 3.3.3.3 detectLeftSensor()

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

Fire left sensor only

## Returns

0 or 1 for legal obstacle distance detected

#### 3.3.3.4 detectRightSensor()

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

Fire right sensor only

##### Returns

0 or 1 for legal obstacle distance detected

#### 3.3.3.5 odsToNavTestObstacles()

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

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

### 3.3.4 Member Data Documentation

#### 3.3.4.1 frontSensorPtr\_

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

Pointer to front sensor.

#### 3.3.4.2 iterations

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

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

#### 3.3.4.3 leftSensorPtr\_

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

Pointer to left sensor.

#### 3.3.4.4 navPtr\_

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

Pointer to the navigation system object.

#### 3.3.4.5 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](#)

## 3.4 ObstacleSensor Class Reference

```
#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.*

### 3.4.1 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. 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.

### 3.4.2 Constructor & Destructor Documentation

#### 3.4.2.1 ObstacleSensor() [1/2]

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

Default constructor - not used.

#### 3.4.2.2 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

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

### 3.4.3 Member Function Documentation

#### 3.4.3.1 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

<i>iterations</i>	Number of iterations sensor measures distance to calculate average over
-------------------	---

#### Returns

0 or 1 for legal obstacle distance detected

#### 3.4.3.2 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

<i>dhtPin</i>	Arduino connected pin number of DHT sensor
---------------	--

#### 3.4.3.3 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

<i>sensorName</i>	Name of sensor to be printed
-------------------	------------------------------

#### 3.4.3.4 printSound()

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

## Parameters

<i>temp</i>	Calculated temperature
<i>hum</i>	Calculated humidity

## 3.4.3.5 updateOdsData()

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

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

## Parameters

<i>x</i>	Current X location of pozyx
<i>y</i>	Current Y location of pozyx
<i>heading</i>	Current heading of pozyx

## 3.4.4 Member Data Documentation

## 3.4.4.1 bias\_

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

Bias for grid reference conversion.

## 3.4.4.2 distance\_

```
float ObstacleSensor::distance_
```

Distance from sensor to object detected.

## 3.4.4.3 duration\_

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

Stores First HC-SR04 pulse duration value.

#### 3.4.4.4 echoPin\_

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

#### 3.4.4.5 gridX\_

```
int ObstacleSensor::gridX_
```

Grid reference of obstacle detected converted from coordinates above.

#### 3.4.4.6 gridY\_

```
int ObstacleSensor::gridY_
```

#### 3.4.4.7 heading\_

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

AVS pozyx heading data.

#### 3.4.4.8 maxDistance\_

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

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

#### 3.4.4.9 objGridRef\_

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

objGridRef\_[0] = x coordinate, objGridRef\_[1] = y coordinate



**3.4.4.10 objX\_**

```
float ObstacleSensor::objX_
```

Grid coordinates at location object detected.

**3.4.4.11 objXDist\_**

```
float ObstacleSensor::objXDist_
```

Distances from pozyx to object detected.

**3.4.4.12 objY\_**

```
float ObstacleSensor::objY_
```

**3.4.4.13 objYDist\_**

```
float ObstacleSensor::objYDist_
```

**3.4.4.14 offsetX\_**

```
float ObstacleSensor::offsetX_
```

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

**3.4.4.15 offsetY\_**

```
float ObstacleSensor::offsetY_
```

**3.4.4.16 sensorAngle\_**

```
float ObstacleSensor::sensorAngle_
```

#### 3.4.4.17 sensorGridAngle\_

```
float ObstacleSensor::sensorGridAngle_
```

Angle of sensor relative to the grid (i.e. sensorAngle\_ + heading\_)

#### 3.4.4.18 sonar\_

```
NewPing ObstacleSensor::sonar_
```

[NewPing](#) object causes HC-SR04 sensor pulses and enable.

#### 3.4.4.19 soundcm\_

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

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

#### 3.4.4.20 triggerPin\_

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

Pin definitions.

#### 3.4.4.21 unitVect\_

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

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

#### 3.4.4.22 xPos\_

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

AVS pozyx x position.

#### 3.4.4.23 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/bot↔ Main/[ObstacleSensor.h](#)
- D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git\_edpAutoCar/bot↔ Main/[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

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

#### Macros

- #define LEFT  $270 \times (\text{PI}/180)$ ;
- #define FORWARD  $0 \times (\text{PI}/180)$ ;
- #define RIGHT  $90 \times (\text{PI}/180)$ ;
- #define BACKWARD  $180 \times (\text{PI}/180)$ ;
- #define DIAG\_FOR\_RIGHT  $45 \times (\text{PI}/180)$ ;
- #define DIAG\_FOR\_LEFT  $315 \times (\text{PI}/180)$ ;
- #define DIAG\_BACK\_LEFT  $225 \times (\text{PI}/180)$ ;

#### 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_` =  $0 \times (\text{PI}/180)$
- float `avsX_` = 0
- 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` = 10  
*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` = -5  
*Left sensor x offset on chassis from pozyx location.*
- const float `leftYOffset` = 2  
*Left sensor y offset on chassis from pozyx location.*
- const float `leftsensorAngle` = LEFT  
*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` = 5  
*Right sensor x offset on chassis from pozyx location.*
- const float `rightYOffset` = 2  
*Right sensor y offset on chassis from pozyx location.*
- const float `rightsensorAngle` = RIGHT  
*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 Macro Definition Documentation

#### 4.1.1.1 BACKWARD

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

#### 4.1.1.2 DIAG\_BACK\_LEFT

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

#### 4.1.1.3 DIAG\_FOR\_LEFT

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

#### 4.1.1.4 DIAG\_FOR\_RIGHT

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

#### 4.1.1.5 FORWARD

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

#### 4.1.1.6 LEFT

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

#### 4.1.1.7 RIGHT

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

### 4.1.2 Function Documentation

#### 4.1.2.1 loop()

```
void loop ( )
```

#### 4.1.2.2 setup()

```
void setup ( )
```

#### 4.1.2.3 testBlueToothGrid()

```
void testBlueToothGrid ( )
```

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

### 4.1.3 Variable Documentation

#### 4.1.3.1 avHeading\_

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

\*\*\*\*\* [botMain.ino](#) - Main .ino file for arduino (for now, someone else may have a better one) \*\*\*\*\* // \*\*\*\*\* INS<sub>↔</sub>TRUCTIONS: \*\*\*\*\* // \*\*\*\*\* 1. Ensure all libraries are installed by navigating Sketch > Include Library > Manage Libraries \*\*\*\*\* // \*\*\*\*\* and searching for the following libraries: (i) DHT \*\*\*\*\* // \*\*\*\*\* (ii) Adafruit Unified Sensor \*\*\*\*\* // \*\*\*\*\* (iii) Adafruit AM2315 \*\*\*\*\* // \*\*\*\*\* 2.

#### 4.1.3.2 avX\_

```
float avX_ = 0
```

#### 4.1.3.3 avY\_

```
float avY_ = 0
```

#### 4.1.3.4 dhtPin

```
const uint8_t dhtPin = 14
```

DHT22 Sensor arduino pin number.

#### 4.1.3.5 frontEchoPin

```
const uint8_t frontEchoPin = frontTriggerPin
```

Front sensor arduino echo pin number.

#### 4.1.3.6 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

#### 4.1.3.7 frontsensorAngle

```
const float frontsensorAngle = FORWARD
```

Front sensor angle relative to forward of chassis.

#### 4.1.3.8 frontTriggerPin

```
const uint8_t frontTriggerPin = 17
```

Front sensor arduino trigger pin number.

#### 4.1.3.9 frontXOffset

```
const float frontXOffset = 0
```

Front sensor x offset on chassis from pozix location.

#### 4.1.3.10 frontYOffset

```
const float frontYOffset = 10
```

Front sensor y offset on chassis from pozyx location.

#### 4.1.3.11 leftEchoPin

```
const uint8_t leftEchoPin = leftTriggerPin
```

Left sensor arduino echo pin number.

#### 4.1.3.12 leftSensor

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

Left sensor [ObstacleSensor](#) object initialisation - used to detect obstacles at left of AVS.

#### 4.1.3.13 leftsensorAngle

```
const float leftsensorAngle = LEFT
```

Left sensor angle relative to forward of chassis.

#### 4.1.3.14 leftTriggerPin

```
const uint8_t leftTriggerPin = 15
```

Left sensor arduino trigger pin number.

#### 4.1.3.15 leftXOffset

```
const float leftXOffset = -5
```

Left sensor x offset on chassis from pozyx location.



#### 4.1.3.16 leftYOffset

```
const float leftYOffset = 2
```

Left sensor y offset on chassis from pozyx location.

#### 4.1.3.17 nav

```
Navigator nav
```

#### 4.1.3.18 rightEchoPin

```
const uint8_t rightEchoPin = rightTriggerPin
```

Right sensor arduino echo pin number.

#### 4.1.3.19 rightSensor

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

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

#### 4.1.3.20 rightsensorAngle

```
const float rightsensorAngle = RIGHT
```

Right sensor angle relative to forward of chassis.

#### 4.1.3.21 rightTriggerPin

```
const uint8_t rightTriggerPin = 19
```

Right sensor arduino trigger pin number.

#### 4.1.3.22 rightXOffset

```
const float rightXOffset = 5
```

Right sensor x offset on chassis from pozyx location.

#### 4.1.3.23 rightYOffset

```
const float rightYOffset = 2
```

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
- 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]
```

#### 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/Navigator.cpp File Reference

```
#include "Navigator.h"
```

### 4.4 D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git\_edpAutoCar/botMain/Navigator.h File Reference

```
#include <Arduino.h>
#include "common.h"
```

#### Classes

- class [Navigator](#)

#### Variables

- const int [HEIGHT](#)
- const int [WIDTH](#)
- const int [DATA](#)
- const int [ELEMENT\\_XPOS](#)
- const int [ELEMENT\\_YPOS](#)
- const int [ELEMENT\\_VALUE](#)

#### 4.4.1 Variable Documentation

##### 4.4.1.1 DATA

```
const int DATA
```

##### 4.4.1.2 ELEMENT\_VALUE

```
const int ELEMENT_VALUE
```

##### 4.4.1.3 ELEMENT\_XPOS

```
const int ELEMENT_XPOS
```

##### 4.4.1.4 ELEMENT\_YPOS

```
const int ELEMENT_YPOS
```

##### 4.4.1.5 HEIGHT

```
const int HEIGHT
```

##### 4.4.1.6 WIDTH

```
const int WIDTH
```

#### 4.5 D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git\_↵ edpAutoCar/botMain/NewPing.cpp File Reference

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

## Functions

- [ISR](#) (TIMER2\_COMPA\_vect)

## Variables

- void(\* [intFunc](#) )()
- void(\* [intFunc2](#) )()
- unsigned long [\\_ms\\_cnt\\_reset](#)
- volatile unsigned long [\\_ms\\_cnt](#)

### 4.5.1 Function Documentation

#### 4.5.1.1 ISR()

```
ISR (
    TIMER2_COMPA_vect )
```

### 4.5.2 Variable Documentation

#### 4.5.2.1 \_ms\_cnt

```
volatile unsigned long _ms_cnt
```

#### 4.5.2.2 \_ms\_cnt\_reset

```
unsigned long _ms_cnt_reset
```

#### 4.5.2.3 intFunc

```
void(* intFunc) ()
```

#### 4.5.2.4 intFunc2

```
void(* intFunc2) ()
```

## 4.6 D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git\_↵ edpAutoCar/botMain/NewPing.h File Reference

```
#include <WProgram.h>
#include <pins_arduino.h>
```

### Classes

- class [NewPing](#)

### Macros

- #define [MAX\\_SENSOR\\_DISTANCE](#) 500
- #define [US\\_ROUNDTRIP\\_CM](#) 57
- #define [US\\_ROUNDTRIP\\_IN](#) 146
- #define [ONE\\_PIN\\_ENABLED](#) true
- #define [ROUNDING\\_ENABLED](#) false
- #define [URM37\\_ENABLED](#) false
- #define [TIMER\\_ENABLED](#) true
- #define [NO\\_ECHO](#) 0
- #define [MAX\\_SENSOR\\_DELAY](#) 5800
- #define [ECHO\\_TIMER\\_FREQ](#) 24
- #define [PING\\_MEDIAN\\_DELAY](#) 29000
- #define [PING\\_OVERHEAD](#) 5
- #define [PING\\_TIMER\\_OVERHEAD](#) 13
- #define [US\\_ROUNDTRIP\\_CM](#) 50
- #define [US\\_ROUNDTRIP\\_IN](#) 127
- #define [NewPingConvert](#)(echoTime, conversionFactor) (max((((unsigned int)echoTime + conversionFactor / 2) / conversionFactor, (echoTime ? 1 : 0)))
- #define [PING\\_OVERHEAD](#) 1
- #define [PING\\_TIMER\\_OVERHEAD](#) 1
- #define [TIMER\\_ENABLED](#) false
- #define [DO\\_BITWISE](#) false

### 4.6.1 Macro Definition Documentation

#### 4.6.1.1 DO\_BITWISE

```
#define DO_BITWISE false
```

#### 4.6.1.2 ECHO\_TIMER\_FREQ

```
#define ECHO_TIMER_FREQ 24
```

#### 4.6.1.3 MAX\_SENSOR\_DELAY

```
#define MAX_SENSOR_DELAY 5800
```

#### 4.6.1.4 MAX\_SENSOR\_DISTANCE

```
#define MAX_SENSOR_DISTANCE 500
```

#### 4.6.1.5 NewPingConvert

```
#define NewPingConvert(  
    echoTime,  
    conversionFactor ) (max(((unsigned int)echoTime + conversionFactor / 2) / conversionFactor, (echoTime ? 1 : 0)))
```

#### 4.6.1.6 NO\_ECHO

```
#define NO_ECHO 0
```

#### 4.6.1.7 ONE\_PIN\_ENABLED

```
#define ONE_PIN_ENABLED true
```

#### 4.6.1.8 PING\_MEDIAN\_DELAY

```
#define PING_MEDIAN_DELAY 29000
```

#### 4.6.1.9 PING\_OVERHEAD [1/2]

```
#define PING_OVERHEAD 5
```

#### 4.6.1.10 PING\_OVERHEAD [2/2]

```
#define PING_OVERHEAD 1
```

#### 4.6.1.11 PING\_TIMER\_OVERHEAD [1/2]

```
#define PING_TIMER_OVERHEAD 13
```

#### 4.6.1.12 PING\_TIMER\_OVERHEAD [2/2]

```
#define PING_TIMER_OVERHEAD 1
```

#### 4.6.1.13 ROUNDING\_ENABLED

```
#define ROUNDING_ENABLED false
```

#### 4.6.1.14 TIMER\_ENABLED [1/2]

```
#define TIMER_ENABLED true
```

#### 4.6.1.15 TIMER\_ENABLED [2/2]

```
#define TIMER_ENABLED false
```

#### 4.6.1.16 URM37\_ENABLED

```
#define URM37_ENABLED false
```

#### 4.6.1.17 US\_ROUNDTRIP\_CM [1/2]

```
#define US_ROUNDTRIP_CM 57
```



#### 4.6.1.18 US\_ROUNDTRIP\_CM [2/2]

```
#define US_ROUNDTRIP_CM 50
```

#### 4.6.1.19 US\_ROUNDTRIP\_IN [1/2]

```
#define US_ROUNDTRIP_IN 146
```

#### 4.6.1.20 US\_ROUNDTRIP\_IN [2/2]

```
#define US_ROUNDTRIP_IN 127
```

### 4.7 D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git↔\_edpAutoCar/botMain/ObstacleDetection.cpp File Reference

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

### 4.8 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](#)

### 4.9 D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git↔\_edpAutoCar/botMain/ObstacleSensor.cpp File Reference

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

### 4.10 D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git↔\_edpAutoCar/botMain/ObstacleSensor.h File Reference

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

#### Classes

- class [ObstacleSensor](#)



# Index

`_echoBit`  
    NewPing, 12  
`_echoInput`  
    NewPing, 12  
`_maxEchoTime`  
    NewPing, 12  
`_max_time`  
    NewPing, 12  
`_ms_cnt`  
    NewPing.cpp, 35  
`_ms_cnt_reset`  
    NewPing.cpp, 35  
`_triggerBit`  
    NewPing, 12  
`_triggerMode`  
    NewPing, 12  
`_triggerOutput`  
    NewPing, 13

`activateSensor`  
    ObstacleSensor, 19  
`addObstacle`  
    Navigator, 6  
`avsHeading_`  
    botMain.ino, 28  
`avsX_`  
    botMain.ino, 28  
`avsY_`  
    botMain.ino, 28

`BACKWARD`  
    botMain.ino, 26

`bias_`  
    ObstacleSensor, 21

`botMain.ino`  
    avsHeading\_, 28  
    avsX\_, 28  
    avsY\_, 28  
    BACKWARD, 26  
    DIAG\_BACK\_LEFT, 27  
    DIAG\_FOR\_LEFT, 27  
    DIAG\_FOR\_RIGHT, 27  
    dhtPin, 28  
    FORWARD, 27  
    frontEchoPin, 29  
    frontSensor, 29  
    frontTriggerPin, 29  
    frontXOffset, 29  
    frontYOffset, 29  
    frontsensorAngle, 29

LEFT, 27  
`leftEchoPin`, 30  
`leftSensor`, 30  
`leftTriggerPin`, 30  
`leftXOffset`, 30  
`leftYOffset`, 30  
`leftsensorAngle`, 30  
`loop`, 27  
`nav`, 31  
RIGHT, 27  
`rightEchoPin`, 31  
`rightSensor`, 31  
`rightTriggerPin`, 31  
`rightXOffset`, 31  
`rightYOffset`, 32  
`rightsensorAngle`, 31  
`setup`, 28  
`testBlueToothGrid`, 28

`calculateSoundCm`  
    ObstacleSensor, 20  
`check_timer`  
    NewPing, 9  
`common.h`  
    DATA, 32  
    ELEMENT\_VALUE, 32  
    ELEMENT\_XPOS, 32  
    ELEMENT\_YPOS, 32  
    HEIGHT, 33  
    WIDTH, 33

`convert_cm`  
    NewPing, 9

`convert_in`  
    NewPing, 10

`convertToArray`  
    Navigator, 7

`createMap`  
    Navigator, 7

D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/↵  
Engineering Design Practise/git\_edpAuto↵  
Car/botMain/Navigator.cpp, 33

D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/↵  
Engineering Design Practise/git\_edpAuto↵  
Car/botMain/Navigator.h, 33

D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/↵  
Engineering Design Practise/git\_edpAuto↵  
Car/botMain/NewPing.cpp, 34

D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/↔  
     Engineering Design Practise/git\_edpAuto↔  
     Car/botMain/NewPing.h, [36](#)  
 D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/↔  
     Engineering Design Practise/git\_edpAuto↔  
     Car/botMain/ObstacleDetection.cpp, [39](#)  
 D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/↔  
     Engineering Design Practise/git\_edpAuto↔  
     Car/botMain/ObstacleDetection.h, [39](#)  
 D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/↔  
     Engineering Design Practise/git\_edpAuto↔  
     Car/botMain/ObstacleSensor.cpp, [39](#)  
 D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/↔  
     Engineering Design Practise/git\_edpAuto↔  
     Car/botMain/ObstacleSensor.h, [39](#)  
 D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/↔  
     Engineering Design Practise/git\_edpAuto↔  
     Car/botMain/botMain.ino, [25](#)  
 D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/↔  
     Engineering Design Practise/git\_edpAuto↔  
     Car/botMain/common.h, [32](#)  
 DATA  
     common.h, [32](#)  
     Navigator.h, [34](#)  
 DIAG\_BACK\_LEFT  
     botMain.ino, [27](#)  
 DIAG\_FOR\_LEFT  
     botMain.ino, [27](#)  
 DIAG\_FOR\_RIGHT  
     botMain.ino, [27](#)  
 DO\_BITWISE  
     NewPing.h, [36](#)  
 detectAllSensors  
     ObstacleDetection, [15](#)  
 detectFrontSensor  
     ObstacleDetection, [15](#)  
 detectLeftSensor  
     ObstacleDetection, [15](#)  
 detectRightSensor  
     ObstacleDetection, [15](#)  
 dhtPin  
     botMain.ino, [28](#)  
 distance\_  
     ObstacleSensor, [21](#)  
 duration\_  
     ObstacleSensor, [21](#)  
 ECHO\_TIMER\_FREQ  
     NewPing.h, [36](#)  
 ELEMENT\_VALUE  
     common.h, [32](#)  
     Navigator.h, [34](#)  
 ELEMENT\_XPOS  
     common.h, [32](#)  
     Navigator.h, [34](#)  
 ELEMENT\_YPOS  
     common.h, [32](#)  
     Navigator.h, [34](#)  
 echoPin\_  
     ObstacleSensor, [21](#)  
 FORWARD  
     botMain.ino, [27](#)  
 frontEchoPin  
     botMain.ino, [29](#)  
 frontSensor  
     botMain.ino, [29](#)  
 frontSensorPtr\_  
     ObstacleDetection, [16](#)  
 frontTriggerPin  
     botMain.ino, [29](#)  
 frontXOffset  
     botMain.ino, [29](#)  
 frontYOffset  
     botMain.ino, [29](#)  
 frontsensorAngle  
     botMain.ino, [29](#)  
 grid\_  
     Navigator, [8](#)  
 gridX\_  
     ObstacleSensor, [22](#)  
 gridY\_  
     ObstacleSensor, [22](#)  
 HEIGHT  
     common.h, [33](#)  
     Navigator.h, [34](#)  
 heading\_  
     ObstacleSensor, [22](#)  
 ISR  
     NewPing.cpp, [35](#)  
 intFunc  
     NewPing.cpp, [35](#)  
 intFunc2  
     NewPing.cpp, [35](#)  
 iterations  
     ObstacleDetection, [16](#)  
 LEFT  
     botMain.ino, [27](#)  
 leftEchoPin  
     botMain.ino, [30](#)  
 leftSensor  
     botMain.ino, [30](#)  
 leftSensorPtr\_  
     ObstacleDetection, [16](#)  
 leftTriggerPin  
     botMain.ino, [30](#)  
 leftXOffset  
     botMain.ino, [30](#)  
 leftYOffset  
     botMain.ino, [30](#)  
 leftsensorAngle  
     botMain.ino, [30](#)  
 loop  
     botMain.ino, [27](#)

- MAX\_SENSOR\_DELAY
  - NewPing.h, [36](#)
- MAX\_SENSOR\_DISTANCE
  - NewPing.h, [37](#)
- maxDistance\_
  - ObstacleSensor, [22](#)
- NO\_ECHO
  - NewPing.h, [37](#)
- nav
  - botMain.ino, [31](#)
- navPtr\_
  - ObstacleDetection, [16](#)
- Navigator, [5](#)
  - addObstacle, [6](#)
  - convertToArray, [7](#)
  - createMap, [7](#)
  - grid\_, [8](#)
  - Navigator, [5](#)
  - printMap, [7](#)
  - testMap, [7](#)
  - testObstacleData, [7](#)
- Navigator.h
  - DATA, [34](#)
  - ELEMENT\_VALUE, [34](#)
  - ELEMENT\_XPOS, [34](#)
  - ELEMENT\_YPOS, [34](#)
  - HEIGHT, [34](#)
  - WIDTH, [34](#)
- NewPing, [8](#)
  - \_echoBit, [12](#)
  - \_echoInput, [12](#)
  - \_maxEchoTime, [12](#)
  - \_max\_time, [12](#)
  - \_triggerBit, [12](#)
  - \_triggerMode, [12](#)
  - \_triggerOutput, [13](#)
  - check\_timer, [9](#)
  - convert\_cm, [9](#)
  - convert\_in, [10](#)
  - NewPing, [9](#)
  - ping, [10](#)
  - ping\_cm, [10](#)
  - ping\_in, [10](#)
  - ping\_median, [10](#)
  - ping\_result, [13](#)
  - ping\_timer, [10](#)
  - ping\_trigger, [10](#)
  - ping\_trigger\_timer, [11](#)
  - ping\_wait\_timer, [11](#)
  - set\_max\_distance, [11](#)
  - timer\_ms, [11](#)
  - timer\_ms\_cntdown, [11](#)
  - timer\_setup, [11](#)
  - timer\_stop, [11](#)
  - timer\_us, [12](#)
- NewPing.cpp
  - \_ms\_cnt, [35](#)
  - \_ms\_cnt\_reset, [35](#)
  - ISR, [35](#)
  - intFunc, [35](#)
  - intFunc2, [35](#)
- NewPing.h
  - DO\_BITWISE, [36](#)
  - ECHO\_TIMER\_FREQ, [36](#)
  - MAX\_SENSOR\_DELAY, [36](#)
  - MAX\_SENSOR\_DISTANCE, [37](#)
  - NO\_ECHO, [37](#)
  - NewPingConvert, [37](#)
  - ONE\_PIN\_ENABLED, [37](#)
  - PING\_MEDIAN\_DELAY, [37](#)
  - PING\_OVERHEAD, [37](#)
  - PING\_TIMER\_OVERHEAD, [38](#)
  - ROUNDING\_ENABLED, [38](#)
  - TIMER\_ENABLED, [38](#)
  - URM37\_ENABLED, [38](#)
  - US\_ROUNDTRIP\_CM, [38](#)
  - US\_ROUNDTRIP\_IN, [39](#)
- NewPingConvert
  - NewPing.h, [37](#)
- ONE\_PIN\_ENABLED
  - NewPing.h, [37](#)
- objGridRef\_
  - ObstacleSensor, [22](#)
- objX\_
  - ObstacleSensor, [22](#)
- objXDist\_
  - ObstacleSensor, [23](#)
- objY\_
  - ObstacleSensor, [23](#)
- objYDist\_
  - ObstacleSensor, [23](#)
- ObstacleDetection, [13](#)
  - detectAllSensors, [15](#)
  - detectFrontSensor, [15](#)
  - detectLeftSensor, [15](#)
  - detectRightSensor, [15](#)
  - frontSensorPtr\_, [16](#)
  - iterations, [16](#)
  - leftSensorPtr\_, [16](#)
  - navPtr\_, [16](#)
  - ObstacleDetection, [14](#)
  - odsToNavTestObstacles, [16](#)
  - rightSensorPtr\_, [17](#)
- ObstacleSensor, [17](#)
  - activateSensor, [19](#)
  - bias\_, [21](#)
  - calculateSoundCm, [20](#)
  - distance\_, [21](#)
  - duration\_, [21](#)
  - echoPin\_, [21](#)
  - gridX\_, [22](#)
  - gridY\_, [22](#)
  - heading\_, [22](#)
  - maxDistance\_, [22](#)
  - objGridRef\_, [22](#)
  - objX\_, [22](#)

- objXDist\_, 23
- objY\_, 23
- objYDist\_, 23
- ObstacleSensor, 19
- offsetX\_, 23
- offsetY\_, 23
- printDistance, 20
- printSound, 20
- sensorAngle\_, 23
- sensorGridAngle\_, 23
- sonar\_, 24
- soundcm\_, 24
- triggerPin\_, 24
- unitVect\_, 24
- updateOdsData, 21
- xPos\_, 24
- yPos\_, 24
- odsToNavTestObstacles
  - ObstacleDetection, 16
- offsetX\_
  - ObstacleSensor, 23
- offsetY\_
  - ObstacleSensor, 23
- PING\_MEDIAN\_DELAY
  - NewPing.h, 37
- PING\_OVERHEAD
  - NewPing.h, 37
- PING\_TIMER\_OVERHEAD
  - NewPing.h, 38
- ping
  - NewPing, 10
- ping\_cm
  - NewPing, 10
- ping\_in
  - NewPing, 10
- ping\_median
  - NewPing, 10
- ping\_result
  - NewPing, 13
- ping\_timer
  - NewPing, 10
- ping\_trigger
  - NewPing, 10
- ping\_trigger\_timer
  - NewPing, 11
- ping\_wait\_timer
  - NewPing, 11
- printDistance
  - ObstacleSensor, 20
- printMap
  - Navigator, 7
- printSound
  - ObstacleSensor, 20
- RIGHT
  - botMain.ino, 27
- ROUNDING\_ENABLED
  - NewPing.h, 38
- rightEchoPin
  - botMain.ino, 31
- rightSensor
  - botMain.ino, 31
- rightSensorPtr\_
  - ObstacleDetection, 17
- rightTriggerPin
  - botMain.ino, 31
- rightXOffset
  - botMain.ino, 31
- rightYOffset
  - botMain.ino, 32
- rightsensorAngle
  - botMain.ino, 31
- sensorAngle\_
  - ObstacleSensor, 23
- sensorGridAngle\_
  - ObstacleSensor, 23
- set\_max\_distance
  - NewPing, 11
- setup
  - botMain.ino, 28
- sonar\_
  - ObstacleSensor, 24
- soundcm\_
  - ObstacleSensor, 24
- TIMER\_ENABLED
  - NewPing.h, 38
- testBlueToothGrid
  - botMain.ino, 28
- testMap
  - Navigator, 7
- testObstacleData
  - Navigator, 7
- timer\_ms
  - NewPing, 11
- timer\_ms\_cntdown
  - NewPing, 11
- timer\_setup
  - NewPing, 11
- timer\_stop
  - NewPing, 11
- timer\_us
  - NewPing, 12
- triggerPin\_
  - ObstacleSensor, 24
- URM37\_ENABLED
  - NewPing.h, 38
- US\_ROUNDTRIP\_CM
  - NewPing.h, 38
- US\_ROUNDTRIP\_IN
  - NewPing.h, 39
- unitVect\_
  - ObstacleSensor, 24
- updateOdsData
  - ObstacleSensor, 21

## WIDTH

- common.h, [33](#)

- Navigator.h, [34](#)

## xPos\_

- ObstacleSensor, [24](#)

## yPos\_

- ObstacleSensor, [24](#)