

EDP AVS

Generated by Doxygen 1.8.14

Contents

1	Class Index	1
1.1	Class List	1
2	File Index	3
2.1	File List	3
3	Class Documentation	5
3.1	ObstacleDetection Class Reference	5
3.1.1	Detailed Description	6
3.1.2	Constructor & Destructor Documentation	6
3.1.2.1	ObstacleDetection() [1/2]	6
3.1.2.2	ObstacleDetection() [2/2]	6
3.1.3	Member Function Documentation	7
3.1.3.1	detectAllSensors()	7
3.1.3.2	detectFrontSensor()	7
3.1.3.3	detectLeftSensor()	7
3.1.3.4	detectRightSensor()	7
3.1.3.5	odsToNavTestObstacles()	8
3.1.4	Member Data Documentation	8
3.1.4.1	frontSensorPtr_	8
3.1.4.2	iterations	8
3.1.4.3	leftSensorPtr_	8
3.1.4.4	navPtr_	8
3.1.4.5	rightSensorPtr_	9

3.2	ObstacleSensor Class Reference	9
3.2.1	Detailed Description	10
3.2.2	Constructor & Destructor Documentation	10
3.2.2.1	ObstacleSensor() [1/2]	11
3.2.2.2	ObstacleSensor() [2/2]	11
3.2.3	Member Function Documentation	11
3.2.3.1	activateSensor()	11
3.2.3.2	calculateSoundCm()	12
3.2.3.3	printDistance()	12
3.2.3.4	printSound()	12
3.2.3.5	updateOdsData()	12
3.2.4	Member Data Documentation	13
3.2.4.1	bias_	13
3.2.4.2	distance_	13
3.2.4.3	duration_	13
3.2.4.4	echoPin_	13
3.2.4.5	gridX_	14
3.2.4.6	gridY_	14
3.2.4.7	heading_	14
3.2.4.8	maxDistance_	14
3.2.4.9	objGridRef_	14
3.2.4.10	objX_	14
3.2.4.11	objXDist_	15
3.2.4.12	objY_	15
3.2.4.13	objYDist_	15
3.2.4.14	offsetX_	15
3.2.4.15	offsetY_	15
3.2.4.16	sensorAngle_	15
3.2.4.17	sensorGridAngle_	15
3.2.4.18	sonar_	16
3.2.4.19	soundcm_	16
3.2.4.20	triggerPin_	16
3.2.4.21	unitVect_	16
3.2.4.22	xPos_	16
3.2.4.23	yPos_	16

4 File Documentation	17
4.1 D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAuto↔ Car/botMain/botMain.ino File Reference	17
4.1.1 Detailed Description	18
4.1.2 Macro Definition Documentation	19
4.1.2.1 BACKWARD	19
4.1.2.2 DIAG_BACK_LEFT	19
4.1.2.3 DIAG_BACK_RIGHT	19
4.1.2.4 DIAG_FOR_LEFT	19
4.1.2.5 DIAG_FOR_RIGHT	19
4.1.2.6 FORWARD	20
4.1.2.7 LEFT	20
4.1.2.8 RIGHT	20
4.1.3 Function Documentation	20
4.1.3.1 loop()	20
4.1.3.2 setup()	20
4.1.3.3 testBlueToothGrid()	20
4.1.4 Variable Documentation	21
4.1.4.1 avsHeading_	21
4.1.4.2 avsX_	21
4.1.4.3 avsY_	21
4.1.4.4 dhtPin	21
4.1.4.5 frontEchoPin	21
4.1.4.6 frontSensor	21
4.1.4.7 frontsensorAngle	22
4.1.4.8 frontTriggerPin	22
4.1.4.9 frontXOffset	22
4.1.4.10 frontYOffset	22
4.1.4.11 leftEchoPin	22
4.1.4.12 leftSensor	22
4.1.4.13 leftsensorAngle	23

4.1.4.14	leftTriggerPin	23
4.1.4.15	leftXOffset	23
4.1.4.16	leftYOffset	23
4.1.4.17	nav	23
4.1.4.18	rightEchoPin	23
4.1.4.19	rightSensor	24
4.1.4.20	rightsensorAngle	24
4.1.4.21	rightTriggerPin	24
4.1.4.22	rightXOffset	24
4.1.4.23	rightYOffset	24
4.2	D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAuto↔ Car/botMain/common.h File Reference	24
4.2.1	Variable Documentation	25
4.2.1.1	DATA	25
4.2.1.2	ELEMENT_VALUE	25
4.2.1.3	ELEMENT_XPOS	25
4.2.1.4	ELEMENT_YPOS	25
4.2.1.5	HEIGHT	25
4.2.1.6	WIDTH	25
4.3	D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAuto↔ Car/botMain/ObstacleDetection.cpp File Reference	25
4.4	D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAuto↔ Car/botMain/ObstacleDetection.h File Reference	26
4.5	D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAuto↔ Car/botMain/ObstacleSensor.cpp File Reference	26
4.6	D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAuto↔ Car/botMain/ObstacleSensor.h File Reference	26
Index		27

Chapter 1

Class Index

1.1 Class List

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

ObstacleDetection	
ObstacleDetection object controls the activation patterns of all ObstacleSensor objects and outputs detected obstacle grid references to the navigation system	5
ObstacleSensor	
ObstacleSensor object contains all angles and distances/locations of a particular sensor allowing for distance calculations from the pozyx locator to be conducted	9

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/ botMain.ino Top level file for the AVS which integrates all sub-systems on the arduino	17
D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAutoCar/bot↔ Main/ common.h	24
D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAutoCar/bot↔ Main/ ObstacleDetection.cpp	25
D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAutoCar/bot↔ Main/ ObstacleDetection.h	26
D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAutoCar/bot↔ Main/ ObstacleSensor.cpp	26
D:/OneDrive - UNSW/Alex/ADFA/2018/Semester 1/Engineering Design Practise/git_edpAutoCar/bot↔ Main/ ObstacleSensor.h	26

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

<code>*frontSensorPtr</code>	Pointer to front sensor Obstacle Sensor Object
<code>*leftSensororPtr</code>	Pointer to left sensor Obstacle Sensor Object
<code>*rightSensorPtr</code>	Pointer to right sensor Obstacle Sensor Object
<code>*navPtr</code>	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

<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

3.2.3 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

<i>iterations</i>	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

<i>dhtPin</i>	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

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

3.2.3.4 printSound()

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

Parameters

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

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

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

`_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](#)