

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	Navigator Class Reference	5
3.1.1	Detailed Description	6
3.1.2	Constructor & Destructor Documentation	6
3.1.2.1	Navigator()	6
3.1.3	Member Function Documentation	6
3.1.3.1	addObstacle()	6
3.1.3.2	convertToArray()	7
3.1.3.3	createMap()	7
3.1.3.4	milliToGrid()	7
3.1.3.5	nextDirection()	8
3.1.3.6	nextMove()	8
3.1.3.7	noObstacle()	9
3.1.3.8	printMap()	9
3.1.3.9	printMoveData()	9
3.1.3.10	printReturnMoveData()	9
3.1.3.11	prioritiseMap()	10
3.1.3.12	reprioritiseMap()	10

3.1.3.13	sidesClear()	10
3.1.3.14	testMap()	11
3.1.3.15	testMilliToGrid()	11
3.1.3.16	testMove()	11
3.1.3.17	testObstacleData()	11
3.1.4	Member Data Documentation	11
3.1.4.1	grid_	11
3.2	ObstacleDetection Class Reference	12
3.2.1	Detailed Description	12
3.2.2	Constructor & Destructor Documentation	13
3.2.2.1	ObstacleDetection() [1/2]	13
3.2.2.2	ObstacleDetection() [2/2]	13
3.2.3	Member Function Documentation	13
3.2.3.1	detectAllSensors()	13
3.2.3.2	detectFrontSensor()	13
3.2.3.3	detectLeftSensor()	14
3.2.3.4	detectRightSensor()	14
3.2.3.5	odsToNavTestObstacles()	14
3.2.4	Member Data Documentation	14
3.2.4.1	frontSensorPtr_	14
3.2.4.2	iterations	15
3.2.4.3	leftSensorPtr_	15
3.2.4.4	navPtr_	15
3.2.4.5	rightSensorPtr_	15
3.3	ObstacleSensor Class Reference	15
3.3.1	Detailed Description	17
3.3.2	Constructor & Destructor Documentation	17
3.3.2.1	ObstacleSensor() [1/2]	17
3.3.2.2	ObstacleSensor() [2/2]	17
3.3.3	Member Function Documentation	18

3.3.3.1	activateSensor()	18
3.3.3.2	calculateSoundCm()	18
3.3.3.3	printDistance()	18
3.3.3.4	printSound()	19
3.3.3.5	updateOdsData()	19
3.3.4	Member Data Documentation	19
3.3.4.1	bias_	19
3.3.4.2	distance_	19
3.3.4.3	duration_	20
3.3.4.4	echoPin_	20
3.3.4.5	gridX_	20
3.3.4.6	gridY_	20
3.3.4.7	heading_	20
3.3.4.8	maxDistance_	20
3.3.4.9	objGridRef_	21
3.3.4.10	objX_	21
3.3.4.11	objXDist_	21
3.3.4.12	objY_	21
3.3.4.13	objYDist_	21
3.3.4.14	offsetX_	21
3.3.4.15	offsetY_	21
3.3.4.16	sensorAngle_	22
3.3.4.17	sensorGridAngle_	22
3.3.4.18	sonar_	22
3.3.4.19	soundcm_	22
3.3.4.20	triggerPin_	22
3.3.4.21	unitVect_	22
3.3.4.22	xDistComp_	23
3.3.4.23	xPos_	23
3.3.4.24	yDistComp_	23
3.3.4.25	yPos_	23

4 File Documentation	25
4.1 botMain.ino File Reference	25
4.1.1 Detailed Description	28
4.1.2 Macro Definition Documentation	28
4.1.2.1 BACKWARD_DIR	28
4.1.2.2 DIAG_BACK_LEFT	28
4.1.2.3 DIAG_BACK_RIGHT	28
4.1.2.4 DIAG_FOR_LEFT	28
4.1.2.5 DIAG_FOR_RIGHT	29
4.1.2.6 FORWARD_DIR	29
4.1.2.7 LEFT_DIR	29
4.1.2.8 RIGHT_DIR	29
4.1.3 Function Documentation	29
4.1.3.1 BT()	29
4.1.3.2 btPrintMap()	29
4.1.3.3 btPrintSensorDistances()	30
4.1.3.4 cgk_left()	30
4.1.3.5 cgk_right()	30
4.1.3.6 getCoordinates()	30
4.1.3.7 getCoordinatesV2()	30
4.1.3.8 getHeading()	30
4.1.3.9 headingInit()	31
4.1.3.10 loop()	31
4.1.3.11 moveBackward()	31
4.1.3.12 moveForward()	31
4.1.3.13 sendBTData()	31
4.1.3.14 setAnchorsManual()	31
4.1.3.15 setup()	32
4.1.3.16 turnLeft()	32
4.1.3.17 turnRight()	32

4.1.4	Variable Documentation	32
4.1.4.1	AFMS	32
4.1.4.2	algorithm	32
4.1.4.3	anchors	32
4.1.4.4	anchors_x	32
4.1.4.5	anchors_y	33
4.1.4.6	basicAvgAmt	33
4.1.4.7	cgk_back_little	33
4.1.4.8	cgk_back_one	33
4.1.4.9	cgk_brake_time	33
4.1.4.10	cgk_first_angle_divider	33
4.1.4.11	cgk_fwd_end_turn	33
4.1.4.12	cgk_fwd_little	33
4.1.4.13	cgk_fwd_one	34
4.1.4.14	cgk_motor_speed	34
4.1.4.15	cgk_straight	34
4.1.4.16	cgk_turn_delay	34
4.1.4.17	dimension	34
4.1.4.18	frontEchoPin	34
4.1.4.19	frontSensor	34
4.1.4.20	frontsensorAngle	35
4.1.4.21	frontTriggerPin	35
4.1.4.22	frontXOffset	35
4.1.4.23	frontYOffset	35
4.1.4.24	heading	35
4.1.4.25	headingOffset	35
4.1.4.26	height	35
4.1.4.27	heights	36
4.1.4.28	leftEchoPin	36
4.1.4.29	leftSensor	36

4.1.4.30	leftsensorAngle	36
4.1.4.31	leftTriggerPin	36
4.1.4.32	leftXOffset	36
4.1.4.33	leftYOffset	37
4.1.4.34	myMotor	37
4.1.4.35	nav	37
4.1.4.36	num_anchors	37
4.1.4.37	num_to_avg	37
4.1.4.38	result	37
4.1.4.39	returnToStart	37
4.1.4.40	rightEchoPin	38
4.1.4.41	rightSensor	38
4.1.4.42	rightsensorAngle	38
4.1.4.43	rightTriggerPin	38
4.1.4.44	rightXOffset	38
4.1.4.45	rightYOffset	38
4.1.4.46	servo1	39
4.1.4.47	tempPin	39
4.1.4.48	x_loc	39
4.1.4.49	x_loc_prev	39
4.1.4.50	xPosStart	39
4.1.4.51	xPosTarget	39
4.1.4.52	y_loc	39
4.1.4.53	y_loc_prev	40
4.1.4.54	yPosStart	40
4.1.4.55	yPosTarget	40
4.2	common.h File Reference	40
4.2.1	Variable Documentation	40
4.2.1.1	DATA	40
4.2.1.2	ELEMENT_VALUE	40

4.2.1.3	ELEMENT_XPOS	41
4.2.1.4	ELEMENT_YPOS	41
4.2.1.5	HEIGHT	41
4.2.1.6	OBSTACLE_COUNT	41
4.2.1.7	test_mode	41
4.2.1.8	WIDTH	41
4.3	Navigator.cpp File Reference	41
4.3.1	Variable Documentation	42
4.3.1.1	moveHistory	42
4.3.1.2	xPosStartNav	42
4.3.1.3	xPosTargetNav	42
4.3.1.4	yPosStartNav	42
4.3.1.5	yPosTargetNav	42
4.4	Navigator.h File Reference	42
4.4.1	Variable Documentation	43
4.4.1.1	DATA	43
4.4.1.2	ELEMENT_VALUE	43
4.4.1.3	ELEMENT_XPOS	43
4.4.1.4	ELEMENT_YPOS	43
4.4.1.5	HEIGHT	44
4.4.1.6	OBSTACLE_COUNT	44
4.4.1.7	WIDTH	44
4.5	ObstacleDetection.cpp File Reference	44
4.6	ObstacleDetection.h File Reference	44
4.7	ObstacleSensor.cpp File Reference	44
4.8	ObstacleSensor.h File Reference	44
	Index	45

Chapter 1

Class Index

1.1 Class List

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

Navigator	
Navigator object controls all aspects of navigation command and decision making	5
ObstacleDetection	
ObstacleDetection object controls the activation patterns of all ObstacleSensor objects and outputs detected obstacle grid references to the navigation system	12
ObstacleSensor	
ObstacleSensor object contains all angles and distances/locations of a particular sensor allowing for distance calculations from the pozyx locator to be conducted	15

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

botMain.ino	
Top level file for the AVS which integrates all sub-systems on the arduino	25
common.h	40
Navigator.cpp	41
Navigator.h	42
ObstacleDetection.cpp	44
ObstacleDetection.h	44
ObstacleSensor.cpp	44
ObstacleSensor.h	44

Chapter 3

Class Documentation

3.1 Navigator Class Reference

[Navigator](#) object controls all aspects of navigation command and decision making.

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

Public Member Functions

- [Navigator](#) (int xPosBegin, int yPosBegin, int xPosEnd, int yPosEnd)
- void [prioritiseMap](#) (int xPos, int yPos)
- void [reprioritiseMap](#) (int xPos, int yPos)
- void [addObstacle](#) (int xPos, int yPos)
- void [noObstacle](#) (int xPos, int yPos)
- int [nextDirection](#) (int xPos, int yPos, int [xPosTarget](#), int [yPosTarget](#), int [heading](#))
- int [nextMove](#) (int xPos, int yPos, int [xPosTarget](#), int [yPosTarget](#), int [heading](#))
- int [sidesClear](#) (int xPos, int yPos, int [heading](#))
- void [testMap](#) ()
- void [printMap](#) ()
- void [testObstacleData](#) ()
- void [testMilliToGrid](#) (int i)
- void [testMove](#) ()
- void [printMoveData](#) (int id, int direction, int [heading](#), int command)
- void [printReturnMoveData](#) ()
- int [milliToGrid](#) (int milliCoord)

Public Attributes

- int [grid_](#) [[HEIGHT](#)+2][[WIDTH](#)+2][[DATA](#)]

Private Member Functions

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

3.1.1 Detailed Description

[Navigator](#) object controls all aspects of navigation command and decision making.

[Navigator](#) takes information regarding location, heading and obstacle locations to populate a live map utilising a flood fill algorithm to prioritise movements. It then decides on a movement based on these priorities and commands the motor control sub-system to make the appropriate movement. It also outputs the map via bluetooth to the GUI including obstacles which are stored as a large number within this map.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Navigator()

```
Navigator::Navigator (
    int xPosBegin,
    int yPosBegin,
    int xPosEnd,
    int yPosEnd )
```

Default constructor

Parameters

<i>xPosBegin</i>	Cartesian x-coordinate of the start grid square
<i>yPosBegin</i>	Cartesian y-coordinate of the start grid square
<i>xPosTarget</i>	Cartesian x-coordinate of the target grid square
<i>yPosTarget</i>	Cartesian y-coordinate of the target grid square

3.1.3 Member Function Documentation

3.1.3.1 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

3.1.3.2 convertToArray()

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

Converts y grid coordinate to y array coordinate

Parameters

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

Returns

Array coordinate

3.1.3.3 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

3.1.3.4 milliToGrid()

```
int Navigator::milliToGrid (
    int milliCoord )
```

Converts millimetre coordinate to grid coordinate

Parameters

<i>coord</i>	Millimetre coordinate
--------------	-----------------------

Returns

Grid coordinate

3.1.3.5 nextDirection()

```
int Navigator::nextDirection (
    int xPos,
    int yPos,
    int xPosTarget,
    int yPosTarget,
    int heading )
```

Determines direction of movement with respect to the current grid square and neighbouring values

Parameters

<i>xPos</i>	Current horizontal position
<i>yPos</i>	Current vertical position
<i>xPosTarget</i>	Target horizontal position
<i>yPosTarget</i>	Target vertical position
<i>heading</i>	Current heading

Returns

Integer uniquely mapped to cardinal direction required to advance to target location

3.1.3.6 nextMove()

```
int Navigator::nextMove (
    int xPos,
    int yPos,
    int xPosTarget,
    int yPosTarget,
    int heading )
```

Determines move required to move to a grid square determined by [nextDirection\(\)](#)

Parameters

<i>xPos</i>	Current horizontal position
<i>yPos</i>	Current vertical position
<i>xPosTarget</i>	Target horizontal position
<i>yPosTarget</i>	Target vertical position
<i>heading</i>	Current heading

Returns

Integer uniquely mapped to motor controller move function

3.1.3.7 noObstacle()

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

Removes an obstacle from the map and gives it an element value of 0

Parameters

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

3.1.3.8 printMap()

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

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

3.1.3.9 printMoveData()

```
void Navigator::printMoveData (
    int id,
    int direction,
    int heading,
    int command )
```

Prints movement related data for each move including its number, direction, resultant heading and command

Parameters

<i>id</i>	move number
<i>direction</i>	movement direction
<i>heading</i>	resultant heading after movement
<i>command</i>	movement command to be sent to motor controller

3.1.3.10 printReturnMoveData()

```
void Navigator::printReturnMoveData ( )
```

Prints movement commands for a required return pathway

3.1.3.11 prioritiseMap()

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

Prioritises each grid square using the flood-fill algorithm

Parameters

<i>xPosTarget</i>	Cartesian x-coordinate of the target grid square
<i>yPosTarget</i>	Cartesian y-coordinate of the target grid square

3.1.3.12 reprioritiseMap()

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

Clears current grid and reprioritises each grid square using the flood-fill algorithm

Parameters

<i>xPosTarget</i>	Cartesian x-coordinate of the target grid square
<i>yPosTarget</i>	Cartesian y-coordinate of the target grid square

3.1.3.13 sidesClear()

```
int Navigator::sidesClear (
    int xPos,
    int yPos,
    int heading )
```

Determines if the grid squares of the position

Parameters

<i>xPos</i>	Current horizontal position
<i>yPos</i>	Current vertical position
<i>heading</i>	Current heading

Returns

integer uniquely mapped to a clear side

3.1.3.14 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

3.1.3.15 testMilliToGrid()

```
void Navigator::testMilliToGrid (
    int i )
```

Tests and prints

Parameters

<i>i</i>	Number of iterations to test the conversion of a millimetre location [0, 5000] to grid location [0, 10]
----------	---

3.1.3.16 testMove()

```
void Navigator::testMove ( )
```

Tests moves functions

3.1.3.17 testObstacleData()

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

Adds obstacles from hardcoded obstacle position data

3.1.4 Member Data Documentation**3.1.4.1 grid_**

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

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

- [Navigator.h](#)
- [Navigator.cpp](#)

3.2 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

- int [detectLeftSensor](#) ()
- int [detectRightSensor](#) ()
- int [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 int [iterations](#) = 5
Number of times the sensors are activated to average distance of detection.

3.2.1 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

3.2.2 Constructor & Destructor Documentation

3.2.2.1 ObstacleDetection() [1/2]

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

Default constructor - not used currently.

3.2.2.2 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 Navigator object in order to call addObstacle function and pass obstalce details to navigation system

3.2.3 Member Function Documentation

3.2.3.1 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.

3.2.3.2 detectFrontSensor()

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

Fire front sensor only

Returns

0 or 1 for legal obstacle distance detected

3.2.3.3 detectLeftSensor()

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

Fire left sensor only

Returns

0 or 1 for legal obstacle distance detected

3.2.3.4 detectRightSensor()

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

Fire right sensor only

Returns

0 or 1 for legal obstacle distance detected

3.2.3.5 odsToNavTestObstacles()

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

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

3.2.4 Member Data Documentation**3.2.4.1 frontSensorPtr_**

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

Pointer to front sensor.

3.2.4.2 iterations

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

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

3.2.4.3 leftSensorPtr_

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

Pointer to left sensor.

3.2.4.4 navPtr_

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

Pointer to the navigation system object.

3.2.4.5 rightSensorPtr_

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

Pointer to right sensor.

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

- [ObstacleDetection.h](#)
- [ObstacleDetection.cpp](#)

3.3 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](#) (int triggerPin, int echoPin, float offsetX, float offsetY, float sensorAngle)
- int [activateSensor](#) (int iterations)
- void [printDistance](#) (String sensorName)

Static Public Member Functions

- static void [calculateSoundCm](#) (int dhtPin)
- static void [printSound](#) (float temp, float hum)
- static void [updateOdsData](#) (int x, int y, int [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 [xDistComp_](#)
X and Y Components of distance from sensor.
- float [yDistComp_](#)
- float [sensorGridAngle_](#)
Angle of sensor relative to the grid (i.e. sensorAngle_ + heading_)
- int [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 int [xPos_](#)
AVS pozyx x position.
- static int [yPos_](#)
AVS pozyx y position.
- static int [bias_](#) = 170
Bias for grid reference conversion.

Private Attributes

- int [triggerPin_](#)
Pin definitions.
- int [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.

3.3.1 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.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 ObstacleSensor() [1/2]

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

Default constructor - not used.

3.3.2.2 ObstacleSensor() [2/2]

```
ObstacleSensor::ObstacleSensor (
    int triggerPin,
    int 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

3.3.3 Member Function Documentation

3.3.3.1 activateSensor()

```
int ObstacleSensor::activateSensor (
    int 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

3.3.3.2 calculateSoundCm()

```
void ObstacleSensor::calculateSoundCm (
    int 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
---------------	--

3.3.3.3 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
-------------------	------------------------------

3.3.3.4 printSound()

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

Parameters

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

3.3.3.5 updateOdsData()

```
static void ObstacleSensor::updateOdsData (
    int x,
    int y,
    int 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

3.3.4 Member Data Documentation

3.3.4.1 bias_

```
int ObstacleSensor::bias_ = 170 [static]
```

Bias for grid reference conversion.

3.3.4.2 distance_

```
float ObstacleSensor::distance_
```

Distance from sensor to object detected.

3.3.4.3 duration_

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

Stores First HC-SR04 pulse duration value.

3.3.4.4 echoPin_

```
int ObstacleSensor::echoPin_ [private]
```

3.3.4.5 gridX_

```
int ObstacleSensor::gridX_
```

Grid reference of obstacle detected converted from coordinates above.

3.3.4.6 gridY_

```
int ObstacleSensor::gridY_
```

3.3.4.7 heading_

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

AVS pozyx heading data.

3.3.4.8 maxDistance_

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

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

3.3.4.9 objGridRef_

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

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

3.3.4.10 objX_

```
float ObstacleSensor::objX_
```

Grid coordinates at location object detected.

3.3.4.11 objXDist_

```
float ObstacleSensor::objXDist_
```

Distances from pozyx to object detected.

3.3.4.12 objY_

```
float ObstacleSensor::objY_
```

3.3.4.13 objYDist_

```
float ObstacleSensor::objYDist_
```

3.3.4.14 offsetX_

```
float ObstacleSensor::offsetX_
```

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

3.3.4.15 offsetY_

```
float ObstacleSensor::offsetY_
```

3.3.4.16 sensorAngle_

```
float ObstacleSensor::sensorAngle_
```

3.3.4.17 sensorGridAngle_

```
float ObstacleSensor::sensorGridAngle_
```

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

3.3.4.18 sonar_

```
NewPing ObstacleSensor::sonar_
```

NewPing object causes HC-SR04 sensor pulses and enable.

3.3.4.19 soundcm_

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

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

3.3.4.20 triggerPin_

```
int ObstacleSensor::triggerPin_ [private]
```

Pin definitions.

3.3.4.21 unitVect_

```
int ObstacleSensor::unitVect_[2]
```

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

3.3.4.22 xDistComp_

```
float ObstacleSensor::xDistComp_
```

X and Y Components of distance from sensor.

3.3.4.23 xPos_

```
int ObstacleSensor::xPos_ [static]
```

AVS pozyx x position.

3.3.4.24 yDistComp_

```
float ObstacleSensor::yDistComp_
```

3.3.4.25 yPos_

```
int ObstacleSensor::yPos_ [static]
```

AVS pozyx y position.

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

- [ObstacleSensor.h](#)
- [ObstacleSensor.cpp](#)

Chapter 4

File Documentation

4.1 botMain.ino File Reference

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

```
#include "ObstacleSensor.h"
#include "ObstacleDetection.h"
#include "common.h"
#include <Pozyx.h>
#include <Pozyx_definitions.h>
#include <Servo.h>
#include <Adafruit_MotorShield.h>
#include <SoftwareSerial.h>
```

Macros

- `#define LEFT_DIR 270*(PI/180);`
Left direction relative to front of AVS in radians.
- `#define FORWARD_DIR 0*(PI/180);`
Forward direction relative to front of AVS in radians.
- `#define RIGHT_DIR 90*(PI/180);`
Right direction relative to front of AVS in radians.
- `#define BACKWARD_DIR 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

- SoftwareSerial [BT](#) (19, 18)
- void [setup](#) ()
- void [loop](#) ()
- void [setAnchorsManual](#) ()
 - Function to manually set the anchor coordinates as supplied by pozyx website.*
- void [getCoordinates](#) (int avgNum)
 - Get position information.*
- void [getCoordinatesV2](#) (int avgNum)
 - Get position information more accurately by filtering outliers.*
- void [headingInit](#) ()
 - Initializes the heading by creating an offset so that the heading at init is 0deg NEW 31/05.*
- void [getHeading](#) ()
 - Get heading info updates global heading variable with what appears to be magnetic heading.*
- void [sendBTData](#) ()
 - Get all data from Mazing and Pozyx and concat into string for BT send (HOFFY)*
- void [turnLeft](#) ()
- void [turnRight](#) ()
- void [cgk_left](#) ()
 - cgk Adjusts AVS left - turns hard right, then goes backwards, then hard left, then goes forwards.*
- void [cgk_right](#) ()
 - cgk Adjusts AVS right - turns hard left, then goes backwards, then hard right, then goes forwards.*
- void [moveForward](#) ()
- void [moveBackward](#) ()
- void [btPrintMap](#) ()
- void [btPrintSensorDistances](#) (String sensorName, [ObstacleSensor](#) *sensor)

Variables

- const int [basicAvgAmt](#) = 50
- const int [num_to_avg](#) = 5
- const int [num_anchors](#) = 4
- uint16_t [anchors](#) [[num_anchors](#)] = {0x687c, 0x6821, 0x6827, 0x6851}
- int32_t [anchors_x](#) [[num_anchors](#)] = {0, 0, 4998, 4985}
- int32_t [anchors_y](#) [[num_anchors](#)] = {0, 5005, 4985, 0}
- int32_t [heights](#) [[num_anchors](#)] = {810, 1720, 765, 1715}
- int [algorithm](#) = POZYX_POS_ALG_UWB_ONLY
- int [dimension](#) = POZYX_2D
- int32_t [height](#) = 0
- int [x_loc](#) = 1750
- int [y_loc](#) = 3250
- int [heading](#) = 90
- int [headingOffset](#) = 0
- int [x_loc_prev](#) = 0
- int [y_loc_prev](#) = 0
- String [result](#) = ";911;"
- int [xPosStart](#) = 2
- int [yPosStart](#) = 9
- int [xPosTarget](#) = 7
- int [yPosTarget](#) = 2
- bool [returnToStart](#) = false

- `Navigator nav (xPosStart, yPosStart, xPosTarget, yPosTarget)`
- `const int frontTriggerPin = 35`
Front sensor arduino trigger pin number.
- `const int frontEchoPin = 37`
Front sensor arduino echo pin number.
- `const float frontXOffset = 0`
Front sensor x offset on chassis from pozyx location.
- `const float frontYOffset = 130`
Front sensor y offset on chassis from pozyx location.
- `const float frontsensorAngle = FORWARD_DIR`
Front sensor angle relative to forward of chassis.
- `const int leftTriggerPin = 31`
Left sensor arduino trigger pin number.
- `const int leftEchoPin = 33`
Left sensor arduino echo pin number.
- `const float leftXOffset = 80`
Left sensor x offset on chassis from pozyx location.
- `const float leftYOffset = 130`
Left sensor y offset on chassis from pozyx location.
- `const float leftsensorAngle = LEFT_DIR`
Left sensor angle relative to forward of chassis.
- `const int rightTriggerPin = 39`
Right sensor arduino trigger pin number.
- `const int rightEchoPin = 41`
Right sensor arduino echo pin number.
- `const float rightXOffset = 80`
Right sensor x offset on chassis from pozyx location.
- `const float rightYOffset = 130`
Right sensor y offset on chassis from pozyx location.
- `const float rightsensorAngle = RIGHT_DIR`
Right sensor angle relative to forward of chassis.
- `const int tempPin = A0`
Temperature 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.
- `Adafruit_MotorShield AFMS = Adafruit_MotorShield()`
- `Adafruit_DCMotor * myMotor = AFMS.getMotor(1)`
- `Servo servo1`
- `int cgk_straight = 110`
- `int cgk_back_little = 370`
- `int cgk_fwd_little = 330`
- `int cgk_fwd_end_turn = 350`
- `int cgk_turn_delay = 1000`
- `int cgk_brake_time = 100`
- `int cgk_motor_speed = 254`
- `int cgk_fwd_one = 470`
- `int cgk_back_one = 500`
- `int cgk_first_angle_divider = 2`

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_DIR

```
#define BACKWARD_DIR 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_DIR

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

Forward direction relative to front of AVS in radians.

4.1.2.7 LEFT_DIR

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

Left direction relative to front of AVS in radians.

4.1.2.8 RIGHT_DIR

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

Right direction relative to front of AVS in radians.

4.1.3 Function Documentation

4.1.3.1 BT()

```
SoftwareSerial BT (
    19 ,
    18 )
```

4.1.3.2 btPrintMap()

```
void btPrintMap ( )
```

Print map function equivalent to that used in the navigator class but used in test_mode 2 when serial monitor occurs through the bluetooth port.

4.1.3.3 btPrintSensorDistances()

```
void btPrintSensorDistances (
    String sensorName,
    ObstacleSensor * sensor )
```

Print sensor distances equivalent to that used in the obstacle sensor class but used in test_mode 2 when serial monitor occurs through the bluetooth port.

4.1.3.4 cgk_left()

```
void cgk_left ( )
```

cgk Adjusts AVS left - turns hard right, then goes backwards, then hard left, then goes forwards.

4.1.3.5 cgk_right()

```
void cgk_right ( )
```

cgk Adjusts AVS right - turns hard left, then goes backwards, then hard right, then goes forwards.

4.1.3.6 getCoordinates()

```
void getCoordinates (
    int avgNum )
```

Get position information.

4.1.3.7 getCoordinatesV2()

```
void getCoordinatesV2 (
    int avgNum )
```

Get position information more accurately by filtering outliers.

4.1.3.8 getHeading()

```
void getHeading ( )
```

Get heading info updates global heading variable with what appears to be magnetic heading.

4.1.3.9 headingInit()

```
void headingInit ( )
```

Initializes the heading by creating an offset so that the heading at init is 0deg NEW 31/05.

4.1.3.10 loop()

```
void loop ( )
```

Main loop function which runs while the AVS is working. Utilises all subsystem functions to inform and command the AVS to complete its task.

4.1.3.11 moveBackward()

```
void moveBackward ( )
```

cgk move the car backward 500mm. cgk the plan is to take the current position to determine how far backward to move, and at cgk what perpendicular offset, so that the final heading and position is centred in the cgk backward gird square. It aims to finish on the correct heading too.

4.1.3.12 moveForward()

```
void moveForward ( )
```

cgk move the car forward 500mm. cgk the plan is to take the current position to determine how far forward to move, and at cgk what perpendicular offset, so that the final heading and position is centred in the cgk forward gird square. It aims to finish on the correct heading too.

4.1.3.13 sendBTData()

```
void sendBTData ( )
```

Get all data from Mazing and Pozyx and concat into string for BT send (HOFFY)

4.1.3.14 setAnchorsManual()

```
void setAnchorsManual ( )
```

Function to manually set the anchor coordinates as supplied by pozyx website.

4.1.3.15 setup()

```
void setup ( )
```

Initial setup function begins the various serial monitors, sets up pozyx anchors and heading values, passes location and heading information to obstacle sensor objects and if in operational mode produces a 10 second delay once complete to allow time to reposition the AVS after initial heading has been captured.

4.1.3.16 turnLeft()

```
void turnLeft ( )
```

Turns the AVS in left direction of current heading and moves to left of starting grid square.

4.1.3.17 turnRight()

```
void turnRight ( )
```

Turns the AVS in right direction of current heading and moves to right of starting grid square.

4.1.4 Variable Documentation

4.1.4.1 AFMS

```
Adafruit_MotorShield AFMS = Adafruit_MotorShield()
```

4.1.4.2 algorithm

```
int algorithm = POZYX_POS_ALG_UWB_ONLY
```

4.1.4.3 anchors

```
uint16_t anchors[num_anchors] = {0x687c, 0x6821, 0x6827, 0x6851}
```

4.1.4.4 anchors_x

```
int32_t anchors_x[num_anchors] = {0, 0, 4998, 4985}
```

4.1.4.5 anchors_y

```
int32_t anchors_y[num_anchors] = {0, 5005, 4985, 0}
```

4.1.4.6 basicAvgAmt

```
const int basicAvgAmt = 50
```

4.1.4.7 cgk_back_little

```
int cgk_back_little = 370
```

4.1.4.8 cgk_back_one

```
int cgk_back_one = 500
```

4.1.4.9 cgk_brake_time

```
int cgk_brake_time = 100
```

4.1.4.10 cgk_first_angle_divider

```
int cgk_first_angle_divider = 2
```

4.1.4.11 cgk_fwd_end_turn

```
int cgk_fwd_end_turn = 350
```

4.1.4.12 cgk_fwd_little

```
int cgk_fwd_little = 330
```

4.1.4.13 cgk_fwd_one

```
int cgk_fwd_one = 470
```

4.1.4.14 cgk_motor_speed

```
int cgk_motor_speed = 254
```

4.1.4.15 cgk_straight

```
int cgk_straight = 110
```

4.1.4.16 cgk_turn_delay

```
int cgk_turn_delay = 1000
```

4.1.4.17 dimension

```
int dimension = POZYX_2D
```

4.1.4.18 frontEchoPin

```
const int frontEchoPin = 37
```

Front sensor arduino echo pin number.

4.1.4.19 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

4.1.4.20 frontsensorAngle

```
const float frontsensorAngle = FORWARD_DIR
```

Front sensor angle relative to forward of chassis.

4.1.4.21 frontTriggerPin

```
const int frontTriggerPin = 35
```

Front sensor arduino trigger pin number.

4.1.4.22 frontXOffset

```
const float frontXOffset = 0
```

Front sensor x offset on chassis from pozyx location.

4.1.4.23 frontYOffset

```
const float frontYOffset = 130
```

Front sensor y offset on chassis from pozyx location.

4.1.4.24 heading

```
int heading = 90
```

4.1.4.25 headingOffset

```
int headingOffset = 0
```

4.1.4.26 height

```
int32_t height = 0
```

4.1.4.27 heights

```
int32_t heights[num_anchors] = {810, 1720, 765, 1715}
```

4.1.4.28 leftEchoPin

```
const int leftEchoPin = 33
```

Left sensor arduino echo pin number.

4.1.4.29 leftSensor

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

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

4.1.4.30 leftsensorAngle

```
const float leftsensorAngle = LEFT_DIR
```

Left sensor angle relative to forward of chassis.

4.1.4.31 leftTriggerPin

```
const int leftTriggerPin = 31
```

Left sensor arduino trigger pin number.

4.1.4.32 leftXOffset

```
const float leftXOffset = 80
```

Left sensor x offset on chassis from pozyx location.

4.1.4.33 leftYOffset

```
const float leftYOffset = 130
```

Left sensor y offset on chassis from pozyx location.

4.1.4.34 myMotor

```
Adafruit_DCMotor* myMotor = AFMS.getMotor(1)
```

4.1.4.35 nav

```
Navigator nav(xPosStart, yPosStart, xPosTarget, yPosTarget)
```

4.1.4.36 num_anchors

```
const int num_anchors = 4
```

4.1.4.37 num_to_avg

```
const int num_to_avg = 5
```

4.1.4.38 result

```
String result = ";911;"
```

4.1.4.39 returnToStart

```
bool returnToStart = false
```

4.1.4.40 rightEchoPin

```
const int rightEchoPin = 41
```

Right sensor arduino echo pin number.

4.1.4.41 rightSensor

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

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

4.1.4.42 rightsensorAngle

```
const float rightsensorAngle = RIGHT_DIR
```

Right sensor angle relative to forward of chassis.

4.1.4.43 rightTriggerPin

```
const int rightTriggerPin = 39
```

Right sensor arduino trigger pin number.

4.1.4.44 rightXOffset

```
const float rightXOffset = 80
```

Right sensor x offset on chassis from pozyx location.

4.1.4.45 rightYOffset

```
const float rightYOffset = 130
```

Right sensor y offset on chassis from pozyx location.

4.1.4.46 servo1

```
Servo servo1
```

4.1.4.47 tempPin

```
const int tempPin = A0
```

Temperature Sensor arduino pin number.

4.1.4.48 x_loc

```
int x_loc = 1750
```

4.1.4.49 x_loc_prev

```
int x_loc_prev = 0
```

4.1.4.50 xPosStart

```
int xPosStart = 2
```

4.1.4.51 xPosTarget

```
int xPosTarget = 7
```

4.1.4.52 y_loc

```
int y_loc = 3250
```

4.1.4.53 y_loc_prev

```
int y_loc_prev = 0
```

4.1.4.54 yPosStart

```
int yPosStart = 9
```

4.1.4.55 yPosTarget

```
int yPosTarget = 2
```

4.2 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](#) = 4
- const int [ELEMENT_XPOS](#) = 0
- const int [ELEMENT_YPOS](#) = 1
- const int [ELEMENT_VALUE](#) = 2
- const int [OBSTACLE_COUNT](#) = 3
- const int [test_mode](#) = 2

4.2.1 Variable Documentation

4.2.1.1 DATA

```
const int DATA = 4 [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 OBSTACLE_COUNT

```
const int OBSTACLE_COUNT = 3
```

4.2.1.7 test_mode

```
const int test_mode = 2
```

test_mode changes the type of test output provided to user. USB serial monitor, BT serial monitor and operational mode (i.e. no test data and 10 sec delay enforced after initial setup complete).

4.2.1.8 WIDTH

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

4.3 Navigator.cpp File Reference

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

Variables

- StackArray< int > [moveHistory](#)
- int [xPosStartNav](#)
- int [yPosStartNav](#)
- int [xPosTargetNav](#)
- int [yPosTargetNav](#)

4.3.1 Variable Documentation

4.3.1.1 moveHistory

```
StackArray<int> moveHistory
```

4.3.1.2 xPosStartNav

```
int xPosStartNav
```

4.3.1.3 xPosTargetNav

```
int xPosTargetNav
```

4.3.1.4 yPosStartNav

```
int yPosStartNav
```

4.3.1.5 yPosTargetNav

```
int yPosTargetNav
```

4.4 Navigator.h File Reference

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

Classes

- class [Navigator](#)
[Navigator](#) object controls all aspects of navigation command and decision making.

Variables

- const int [HEIGHT](#)
- const int [WIDTH](#)
- const int [DATA](#)
- const int [ELEMENT_XPOS](#)
- const int [ELEMENT_YPOS](#)
- const int [ELEMENT_VALUE](#)
- const int [OBSTACLE_COUNT](#)

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 OBSTACLE_COUNT

```
const int OBSTACLE_COUNT
```

4.4.1.7 WIDTH

```
const int WIDTH
```

4.5 ObstacleDetection.cpp File Reference

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

4.6 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.

4.7 ObstacleSensor.cpp File Reference

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

4.8 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](#)