# Motorcycle Awareness System (MAS) ME553 Group EPE 6

Generated by Doxygen 1.8.6

Sun Apr 26 2015 18:16:07

ii CONTENTS

## **Contents**

1	Moto	orcycle Awareness System Overview	1					
2	Todo	o List	2					
3	Clas	ss Index	2					
	3.1	Class List	2					
4	File	Index	2					
	4.1	File List	2					
5	Clas	Class Documentation 2						
	5.1	BlueToothMessage_t Struct Reference	3					
		5.1.1 Detailed Description	3					
		5.1.2 Member Data Documentation	3					
	5.2	GpsSignal_t Struct Reference	4					
		5.2.1 Detailed Description	4					
		5.2.2 Member Data Documentation	5					
	5.3	MotorcycleAwarenessSystem Class Reference	5					
		5.3.1 Detailed Description	6					
		5.3.2 Constructor & Destructor Documentation	7					
		5.3.3 Member Function Documentation	7					
		5.3.4 Member Data Documentation	12					
	5.4	MotorCycleLocation_t Struct Reference	13					
		5.4.1 Detailed Description	13					
		5.4.2 Member Data Documentation	13					
5.5 RadarSignal_t Struct Reference								
		5.5.1 Detailed Description	14					
		5.5.2 Member Data Documentation	14					
5.6 TurnSignal_t Struct Reference								
		5.6.1 Detailed Description	15					
		5.6.2 Member Data Documentation	16					
6	File Documentation							
	6.1	main.cpp File Reference	16					
		6.1.1 Function Documentation	17					
	6.2	main.cpp	17					
	6.3	mainpage.dox File Reference						

¥			26	
5.9	Motorcy	rcleAwarenessSystemTypes.hpp	25	
	6.8.2	Enumeration Type Documentation	24	
	6.8.1	Typedef Documentation	24	
8.8	rcleAwarenessSystemTypes.hpp File Reference	23		
.7	MotorcycleAwarenessSystem.hpp			
.6	MotorcycleAwarenessSystem.hpp File Reference			
.5	MotorcycleAwarenessSystem.cpp			
.4	Motorcy	cleAwarenessSystem.cpp File Reference	18	
	6.4 6.5 6.6 6.7 6.8	Motorcy Motorcy Motorcy 6.8 Motorcy 6.8.1 6.8.2 Motorcy	MotorcycleAwarenessSystem.hpp File Reference  MotorcycleAwarenessSystem.hpp  MotorcycleAwarenessSystemTypes.hpp File Reference 6.8.1 Typedef Documentation 6.8.2 Enumeration Type Documentation  MotorcycleAwarenessSystemTypes.hpp  MotorcycleAwarenessSystemTypes.hpp	

## 1 Motorcycle Awareness System Overview

This low-level design documentation details a mockup of the Motorcycle Awareness System. It provides part of the functionality of the MAS for both the motorcycle and car. Mocking of input signals such as the radar signals, vehicle-to-vehicle (V2V) communication signals, and GPS signals mimics the interactions that the MAS would have with actual sensor data on the finished product.

The implemented functionality includes continuous tracking of the motorcycle using GPS, determining whether a hazard is present using the motorcycle's radar sensor signals, and relaying a warning to the motorcycle rider of a potential threat. For the car, the logic determines whether the car's blinker is on based on blinker signal data on the CAN bus, determines whether a motorcycle is in range and assess the potential danger, and issues a warning to the driver as necessary.

#### **MAS Concept Sketch**

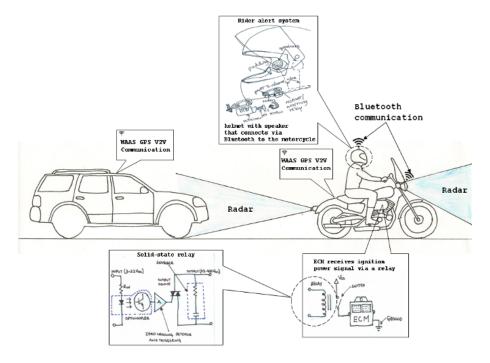


Figure 1: MAS concept sketch

## 2 Todo List

2	1000 LIST	
Ме	ember MotorcycleAwarenessSystem::GetMotorcycleLocation (void)  Acquire the data packet from the motorcycle	
Me	ember MotorcycleAwarenessSystem::IsMotorcycleInRange (void)  Process the data packet and determine threat	
Me	ember MotorcycleAwarenessSystem::RelayWarningToOperator (void)  Transmit the bluetooth message to the operator	
3	Class Index	
3.1	Class List	
He	ere are the classes, structs, unions and interfaces with brief descriptions:	
	BlueToothMessage_t Struct for bluetooth message	3
	GpsSignal_t Structure that emulates a GPS signal	4
	MotorcycleAwarenessSystem Class declaration for the Motorcycle Awareness System (MAS)	5
	MotorCycleLocation_t Struct for the V2V data	13
	RadarSignal_t Structure that emulates a Radar signal	14
	TurnSignal_t	15
4	File Index	
4.1	File List	
He	ere is a list of all files with brief descriptions:	
	main.cpp	16
	MotorcycleAwarenessSystem.cpp	18

## 5 Class Documentation

MotorcycleAwarenessSystem.hpp

 ${\bf Motorcycle Awareness System Types.hpp}$ 

**21** 

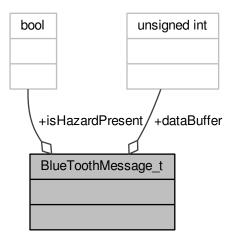
**23** 

#### 5.1 BlueToothMessage\_t Struct Reference

Struct for bluetooth message.

#include <MotorcycleAwarenessSystemTypes.hpp>

Collaboration diagram for BlueToothMessage\_t:



#### **Public Attributes**

bool isHazardPresent

Hazard flag.

• unsigned int dataBuffer [255]

Bluetooth data buffer.

#### 5.1.1 Detailed Description

Struct for bluetooth message.

Definition at line 42 of file MotorcycleAwarenessSystemTypes.hpp.

#### 5.1.2 Member Data Documentation

#### 5.1.2.1 unsigned int BlueToothMessage\_t::dataBuffer[255]

Bluetooth data buffer.

Definition at line 45 of file MotorcycleAwarenessSystemTypes.hpp.

#### 5.1.2.2 bool BlueToothMessage\_t::isHazardPresent

Hazard flag.

Definition at line 44 of file MotorcycleAwarenessSystemTypes.hpp.

The documentation for this struct was generated from the following file:

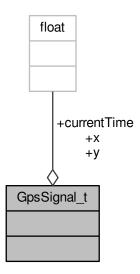
MotorcycleAwarenessSystemTypes.hpp

#### 5.2 GpsSignal\_t Struct Reference

Structure that emulates a GPS signal.

#include <MotorcycleAwarenessSystemTypes.hpp>

Collaboration diagram for GpsSignal\_t:



#### **Public Attributes**

Coordinate\_t x

x-axis coordinate

Coordinate\_t y

y-axis coordinate

currentTime\_t currentTime

Current time at coordinates x,y.

#### 5.2.1 Detailed Description

Structure that emulates a GPS signal.

Definition at line 25 of file MotorcycleAwarenessSystemTypes.hpp.

#### 5.2.2 Member Data Documentation

#### 5.2.2.1 currentTime\_t GpsSignal\_t::currentTime

Current time at coordinates x,y.

Definition at line 29 of file MotorcycleAwarenessSystemTypes.hpp.

#### 5.2.2.2 Coordinate\_t GpsSignal\_t::x

x-axis coordinate

Definition at line 27 of file MotorcycleAwarenessSystemTypes.hpp.

#### 5.2.2.3 Coordinate\_t GpsSignal\_t::y

y-axis coordinate

Definition at line 28 of file MotorcycleAwarenessSystemTypes.hpp.

The documentation for this struct was generated from the following file:

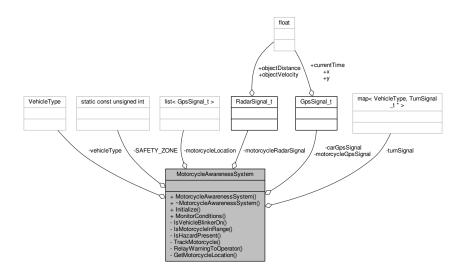
MotorcycleAwarenessSystemTypes.hpp

#### 5.3 MotorcycleAwarenessSystem Class Reference

Class declaration for the Motorcycle Awareness System (MAS)

#include <MotorcycleAwarenessSystem.hpp>

Collaboration diagram for MotorcycleAwarenessSystem:



#### **Public Member Functions**

MotorcycleAwarenessSystem (VehicleType vehicleType)
 Constructor.

~MotorcycleAwarenessSystem (void)

Destructor.

void Initialize (TurnSignal\_t \*motorcycleTurnSignal, TurnSignal\_t \*carTurnSignal, RadarSignal\_t \*motorcycleGpsSignal, GpsSignal t \*carGpsSignal t \*carGpsSignal)

Method to initialize the MAS system.

void MonitorConditions (void)

Method to continuously monitor the conditions during run-time.

#### **Private Member Functions**

bool IsVehicleBlinkerOn (void)

Method to determine whether the car's blinker is ON.

bool IsMotorcycleInRange (void)

Method to determine whether the motorcycle is within the car's range.

- bool IsHazardPresent (void)
- void TrackMotorcycle (void)

Method to track the motorcycle using its GPS signal.

void RelayWarningToOperator (void)

Method to relay a warning to operator via bluetooth connectivity.

MotorCycleLocation\_t GetMotorcycleLocation (void)

#### **Private Attributes**

• std::list< GpsSignal\_t > motorcycleLocation

Container used to track the motorcycle's location.

VehicleType vehicleType

The vehicle type (motorcycle or car)

std::map< VehicleType,</li>
 TurnSignal t \* > turnSignal

Storage for the turn signals.

• RadarSignal\_t \* motorcycleRadarSignal

Pointer to a motorcycle radar signal.

GpsSignal\_t \* motorcycleGpsSignal

Pointer to a motorcycle GPS signal.

GpsSignal\_t \* carGpsSignal

Pointer to a car GPS signal.

#### **Static Private Attributes**

static const unsigned int SAFETY ZONE = 15U

#### 5.3.1 Detailed Description

Class declaration for the Motorcycle Awareness System (MAS)

Definition at line 12 of file MotorcycleAwarenessSystem.hpp.

- 5.3.2 Constructor & Destructor Documentation
- 5.3.2.1 MotorcycleAwarenessSystem::MotorcycleAwarenessSystem ( VehicleType vehicleType )

Constructor.

Class definition for the Motorcycle Awareness System (MAS). This class processes various signals and interactions to realize the MAS

Definition at line 13 of file MotorcycleAwarenessSystem.cpp.

5.3.2.2 MotorcycleAwarenessSystem::~MotorcycleAwarenessSystem (void)

Destructor.

Definition at line 20 of file MotorcycleAwarenessSystem.cpp.

- 5.3.3 Member Function Documentation
- **5.3.3.1** MotorCycleLocation\_t MotorcycleAwarenessSystem::GetMotorcycleLocation(void) [private]

Todo Acquire the data packet from the motorcycle

Definition at line 137 of file MotorcycleAwarenessSystem.cpp.

Here is the caller graph for this function:



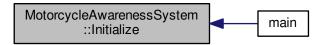
5.3.3.2 void MotorcycleAwarenessSystem::Initialize ( TurnSignal\_t \* motorcycleTurnSignal, TurnSignal\_t \* carTurnSignal, RadarSignal t \* motorcycleRadarSignal, GpsSignal t \* motorcycleGpsSignal, GpsSignal t \* carGpsSignal)

Method to initialize the MAS system.

Definition at line 26 of file MotorcycleAwarenessSystem.cpp.

```
00029 {
00030
          // Initialize the motorcycle radar signal
00031
          this->motorcycleRadarSignal = motorcycleRadarSignal;
00032
00033
          // Initialize the turn signal map
00034
          turnSignal[MOTORCYCLE] = motorcycleTurnSignal;
00035
          turnSignal[CAR] = carTurnSignal;
00036
00037
          // Initialize the GPS signals
00038
          this->motorcycleGpsSignal = motorcycleGpsSignal;
00039
          this->motorcycleGpsSignal = motorcycleGpsSignal;
00040 }
```

Here is the caller graph for this function:



#### **5.3.3.3** bool MotorcycleAwarenessSystem::IsHazardPresent(void) [private]

Method to determine whether a hazard is within the motorcycle's safety zone. An object within the safety zone is of potential danger to the motorcycle rider.

Definition at line 105 of file MotorcycleAwarenessSystem.cpp.

```
00106 {
00107
          bool isHazardPresent = false;
00108
          if ( abs( this->motorcycleRadarSignal->objectDistance ) <=</pre>
00109
      SAFETY_ZONE )
00110
          {
00111
              isHazardPresent = true;
00112
00113
          return isHazardPresent;
00114
00115 }
```

Here is the caller graph for this function:



**5.3.3.4** bool MotorcycleAwarenessSystem::lsMotorcycleInRange(void) [private]

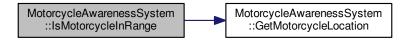
Method to determine whether the motorcycle is within the car's range.

Todo Process the data packet and determine threat

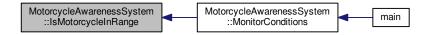
Definition at line 83 of file MotorcycleAwarenessSystem.cpp.

```
00084 {
00085
          bool isInRange = false;
00086
          \ensuremath{//} Determine where the motorcycle is relative to the car using the GPS signals
00087
          if ( abs( (this->carGpsSignal->x) - (this->motorcycleGpsSignal->x) ) <=</pre>
00088
      SAFETY_ZONE &&
               abs( (this->carGpsSignal->y) - (this->motorcycleGpsSignal->y) ) <=</pre>
00089
      SAFETY_ZONE )
00090
          {
00091
               isInRange = true;
00092
00093
          else
00094
          {
               // Analyze the V2V data for a threat
00095
00096
              MotorCycleLocation_t motorCycleLocation =
     GetMotorcycleLocation();
00098
00099
00100
          return isInRange;
00101 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



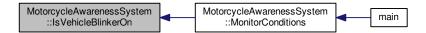
**5.3.3.5** bool MotorcycleAwarenessSystem::lsVehicleBlinkerOn ( void ) [private]

Method to determine whether the car's blinker is ON.

Definition at line 69 of file MotorcycleAwarenessSystem.cpp.

```
00070 {
00071
          bool isBlinkerOn = false;
00072
00073
          if ( this->turnSignal[vehicleType]->isRightBlinkerOn ||
00074
               this->turnSignal[vehicleType]->isLeftBlinkerOn )
00075
00076
              isBlinkerOn = true;
00077
00078
00079
          return isBlinkerOn;
00080 }
```

Here is the caller graph for this function:



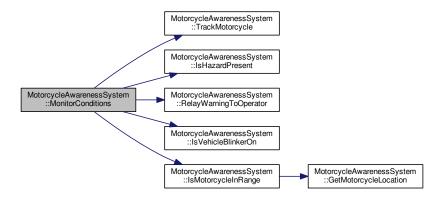
#### 5.3.3.6 void MotorcycleAwarenessSystem::MonitorConditions (void )

Method to continuously monitor the conditions during run-time.

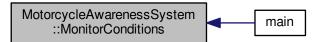
Definition at line 43 of file MotorcycleAwarenessSystem.cpp.

```
00044 {
00045
          if ( MOTORCYCLE == vehicleType )
00046
00047
              // Track the motorcycle
00048
              TrackMotorcycle();
00049
              // Check for hazards
00050
              if ( IsHazardPresent() == true )
00051
                  // Warn the motorcycle operator
00052
00053
                  RelayWarningToOperator();
00054
00055
          // vehicleType == CAR
00056
00057
00058
00059
              // Check for hazards
00060
              if ( (IsVehicleBlinkerOn() == true) && (
     IsMotorcycleInRange() == true) )
00061
00062
                  // Relay message to the car driver
00063
                  RelayWarningToOperator();
00064
00065
00066 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



**5.3.3.7 void MotorcycleAwarenessSystem::RelayWarningToOperator(void)** [private]

Method to relay a warning to operator via bluetooth connectivity.

**Todo** Transmit the bluetooth message to the operator

Definition at line 125 of file MotorcycleAwarenessSystem.cpp.

Here is the caller graph for this function:



**5.3.3.8 void MotorcycleAwarenessSystem::TrackMotorcycle(void)** [private]

Method to track the motorcycle using its GPS signal.

Definition at line 118 of file MotorcycleAwarenessSystem.cpp.

Here is the caller graph for this function:



#### 5.3.4 Member Data Documentation

**5.3.4.1 GpsSignal\_t**\* MotorcycleAwarenessSystem::carGpsSignal [private]

Pointer to a car GPS signal.

Definition at line 30 of file MotorcycleAwarenessSystem.hpp.

**5.3.4.2 GpsSignal** t\* MotorcycleAwarenessSystem::motorcycleGpsSignal [private]

Pointer to a motorcycle GPS signal.

Definition at line 29 of file MotorcycleAwarenessSystem.hpp.

**5.3.4.3** std::list < GpsSignal\_t > MotorcycleAwarenessSystem::motorcycleLocation [private]

Container used to track the motorcycle's location.

Definition at line 23 of file MotorcycleAwarenessSystem.hpp.

**5.3.4.4 RadarSignal t\* MotorcycleAwarenessSystem::motorcycleRadarSignal** [private]

Pointer to a motorcycle radar signal.

Definition at line 28 of file MotorcycleAwarenessSystem.hpp.

**5.3.4.5 const unsigned int MotorcycleAwarenessSystem::SAFETY\_ZONE = 15U** [static], [private]

Distance from the radar sensor in feet in which a detected object becomes a potential danger

Definition at line 24 of file MotorcycleAwarenessSystem.hpp.

**5.3.4.6** std::map<VehicleType, TurnSignal\_t\*> MotorcycleAwarenessSystem::turnSignal [private]

Storage for the turn signals.

Definition at line 27 of file MotorcycleAwarenessSystem.hpp.

**5.3.4.7 VehicleType** MotorcycleAwarenessSystem::vehicleType [private]

The vehicle type (motorcycle or car)

Definition at line 26 of file MotorcycleAwarenessSystem.hpp.

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

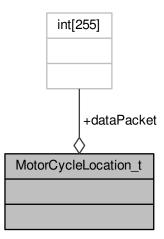
- MotorcycleAwarenessSystem.hpp
- MotorcycleAwarenessSystem.cpp

#### 5.4 MotorCycleLocation\_t Struct Reference

Struct for the V2V data.

#include <MotorcycleAwarenessSystemTypes.hpp>

Collaboration diagram for MotorCycleLocation\_t:



#### **Public Attributes**

DataPacket\_t dataPacket

V2V communication data packet.

#### 5.4.1 Detailed Description

Struct for the V2V data.

Definition at line 50 of file MotorcycleAwarenessSystemTypes.hpp.

#### 5.4.2 Member Data Documentation

#### 5.4.2.1 DataPacket\_t MotorCycleLocation\_t::dataPacket

V2V communication data packet.

Definition at line 52 of file MotorcycleAwarenessSystemTypes.hpp.

The documentation for this struct was generated from the following file:

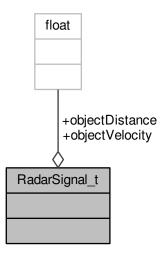
MotorcycleAwarenessSystemTypes.hpp

#### 5.5 RadarSignal\_t Struct Reference

Structure that emulates a Radar signal.

#include <MotorcycleAwarenessSystemTypes.hpp>

Collaboration diagram for RadarSignal\_t:



#### **Public Attributes**

• ObjectDistance objectDistance

Distance from sensed object to radar sensor.

• ObjectVelocity objectVelocity

Velocity of sensed object.

#### 5.5.1 Detailed Description

Structure that emulates a Radar signal.

Definition at line 35 of file MotorcycleAwarenessSystemTypes.hpp.

#### 5.5.2 Member Data Documentation

#### 5.5.2.1 ObjectDistance RadarSignal\_t::objectDistance

Distance from sensed object to radar sensor.

Definition at line 37 of file MotorcycleAwarenessSystemTypes.hpp.

#### 5.5.2.2 ObjectVelocity RadarSignal\_t::objectVelocity

Velocity of sensed object.

Definition at line 38 of file MotorcycleAwarenessSystemTypes.hpp.

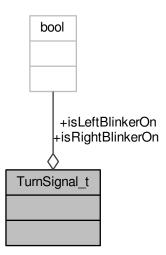
The documentation for this struct was generated from the following file:

MotorcycleAwarenessSystemTypes.hpp

#### 5.6 TurnSignal\_t Struct Reference

#include <MotorcycleAwarenessSystemTypes.hpp>

Collaboration diagram for TurnSignal\_t:



#### **Public Attributes**

• bool isRightBlinkerOn

Right-hand-side blinker signal.

bool isLeftBlinkerOn

Left-hand-side blinker signal.

#### 5.6.1 Detailed Description

Structure used to store the data coming from the turn-signal relay indicating the status of the blinker lights Definition at line 16 of file MotorcycleAwarenessSystemTypes.hpp.

#### 5.6.2 Member Data Documentation

#### 5.6.2.1 bool TurnSignal\_t::isLeftBlinkerOn

Left-hand-side blinker signal.

Definition at line 19 of file MotorcycleAwarenessSystemTypes.hpp.

5.6.2.2 bool TurnSignal\_t::isRightBlinkerOn

Right-hand-side blinker signal.

Definition at line 18 of file MotorcycleAwarenessSystemTypes.hpp.

The documentation for this struct was generated from the following file:

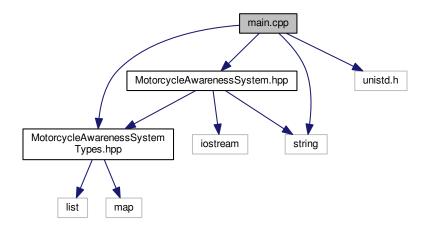
MotorcycleAwarenessSystemTypes.hpp

#### 6 File Documentation

#### 6.1 main.cpp File Reference

```
#include "MotorcycleAwarenessSystem.hpp"
#include "MotorcycleAwarenessSystemTypes.hpp"
#include <string>
#include <unistd.h>
```

Include dependency graph for main.cpp:



#### **Functions**

• int main (void)

6.2 main.cpp 17

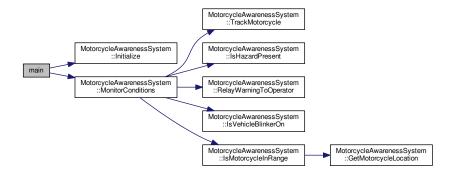
#### 6.1.1 Function Documentation

#### 6.1.1.1 int main ( void )

Definition at line 10 of file main.cpp.

```
00011 {
00012
          // Define mock signals
00013
          TurnSignal_t* motorcycleTurnSignal = new TurnSignal_t;
00014
          TurnSignal_t* carTurnSignal = new TurnSignal_t;
00015
          RadarSignal_t* motorcycleRadarSignal = new RadarSignal_t;
00016
          GpsSignal_t* motorcycleGpsSignal = new GpsSignal_t;
00017
          GpsSignal_t* carGpsSignal = new GpsSignal_t;;
00018
00019
          \ensuremath{//} Instantiate the Motorcycle Awareness System (MAS) for the motorcycle
00020
          MotorcycleAwarenessSystem MAS_motorcycle(
     MOTORCYCLE );
00021
          // Initialize the car MAS
00022
          MAS_motorcycle.Initialize( motorcycleTurnSignal, (TurnSignal_t*)NULL, motorcycleRadarSignal
00023
                                                 motorcycleGpsSignal, carGpsSignal);
00024
00025
          // Instantiate the Motorcycle Awareness System (MAS) for the car
00026
          MotorcycleAwarenessSystem MAS_car( CAR );
00027
          // Initialize the car MAS
00028
          MAS_car.Initialize( (TurnSignal_t*)NULL, carTurnSignal, motorcycleRadarSignal,
00029
                                                 motorcycleGpsSignal, carGpsSignal);
00030
00031
          // Run the MAS systems for the car & motorcycle
00032
00033
00034
              MAS_motorcycle.MonitorConditions();
00035
              MAS_car.MonitorConditions();
00036
              usleep(250);
00037
00038 #ifdef MAS_DEBUG
              std::cout << "The MAS is running" << std::endl;
00039
00040 #endif
00041
00042
00043
          return 0;
00044 }
```

Here is the call graph for this function:



#### 6.2 main.cpp

```
00001 #include "MotorcycleAwarenessSystem.hpp"
00002 #include "MotorcycleAwarenessSystemTypes.hpp"
00003
00004 #ifdef MAS DEBUG
```

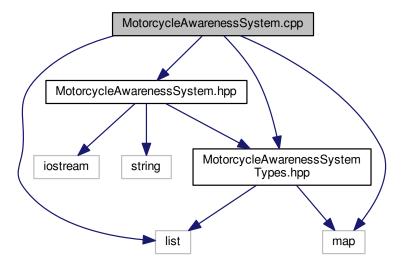
```
00005 #include <iostream>
00006 #endif
00007 #include <string>
00008 #include <unistd.h>
00009
00010 int main( void )
00011 {
00012
          // Define mock signals
00013
          TurnSignal_t* motorcycleTurnSignal = new TurnSignal_t;
          TurnSignal_t* carTurnSignal = new TurnSignal_t;
          RadarSignal_t* motorcycleRadarSignal = new RadarSignal_t;
00016
          GpsSignal_t* motorcycleGpsSignal = new GpsSignal_t;
00017
          GpsSignal_t* carGpsSignal = new GpsSignal_t;;
00019
          // Instantiate the Motorcycle Awareness System (MAS) for the motorcycle
00020
          MotorcycleAwarenessSystem MAS_motorcycle(
     MOTORCYCLE );
          // Initialize the car MAS
00022
          MAS_motorcycle.Initialize( motorcycleTurnSignal, (TurnSignal_t*)NULL,
     motorcycleRadarSignal,
00023
                                                 motorcycleGpsSignal, carGpsSignal );
00024
00025
          \ensuremath{//} Instantiate the Motorcycle Awareness System (MAS) for the car
00026
          MotorcycleAwarenessSystem MAS_car( CAR );
00027
          // Initialize the car MAS
          MAS_car.Initialize( (TurnSignal_t*)NULL, carTurnSignal, motorcycleRadarSignal,
00028
00029
                                                 motorcycleGpsSignal, carGpsSignal );
00030
00031
          // Run the MAS systems for the car & motorcycle
00032
          while (true)
00033
00034
              MAS_motorcycle.MonitorConditions();
00035
              MAS_car.MonitorConditions();
00036
              usleep(250);
00037
00038 #ifdef MAS DEBUG
00039
              std::cout << "The MAS is running" << std::endl;</pre>
00040 #endif
00041
         }
00042
00043
          return 0;
00044 }
```

#### 6.3 mainpage.dox File Reference

#### 6.4 MotorcycleAwarenessSystem.cpp File Reference

```
#include "MotorcycleAwarenessSystem.hpp"
#include "MotorcycleAwarenessSystemTypes.hpp"
#include <list>
#include <map>
```

Include dependency graph for MotorcycleAwarenessSystem.cpp:



#### 6.5 MotorcycleAwarenessSystem.cpp

```
00001 #include "MotorcycleAwarenessSystem.hpp"
00007 #include "MotorcycleAwarenessSystemTypes.hpp"
80000
00009 #include <list>
00010 #include <map>
00011
00013 MotorcycleAwarenessSystem::MotorcycleAwarenessSystem(
     VehicleType vehicleType )
00014
          :vehicleType( vehicleType )
00015 {
00016
          // Do nothing
00017 }
00018
00020 MotorcycleAwarenessSystem::~MotorcycleAwarenessSystem(
       void )
00021 {
00022
          // Do nothing
00023 }
00024
00026 void MotorcycleAwarenessSystem::Initialize(
      TurnSignal_t* motorcycleTurnSignal, TurnSignal_t* carTurnSignal,
                                                  RadarSignal_t* motorcycleRadarSignal,
     GpsSignal_t* motorcycleGpsSignal,
00028
                                                  GpsSignal_t* carGpsSignal )
00029 {
00030
          // Initialize the motorcycle radar signal
00031
          this->motorcycleRadarSignal = motorcycleRadarSignal;
00032
00033
          // Initialize the turn signal map
          turnSignal[MOTORCYCLE] = motorcycleTurnSignal;
00034
00035
         turnSignal[CAR] = carTurnSignal;
00036
00037
          // Initialize the GPS signals
00038
          this->motorcycleGpsSignal = motorcycleGpsSignal;
          this->motorcycleGpsSignal = motorcycleGpsSignal;
00039
00040 }
00041
00043 void MotorcycleAwarenessSystem::MonitorConditions( void )
00044 {
```

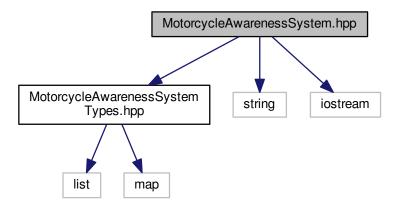
```
00045
          if ( MOTORCYCLE == vehicleType )
00046
          {
00047
              // Track the motorcycle
              TrackMotorcycle();
00048
00049
              // Check for hazards
00050
              if ( IsHazardPresent() == true )
00051
              {
00052
                   // Warn the motorcycle operator
00053
                  RelayWarningToOperator();
00054
00055
00056
          // vehicleType == CAR
00057
          else
00058
00059
              // Check for hazards
00060
              if ((IsVehicleBlinkerOn() == true) && (
      IsMotorcycleInRange() == true) )
00061
00062
                   // Relay message to the car driver
00063
                  RelayWarningToOperator();
00064
00065
00066 }
00067
00069 bool MotorcycleAwarenessSystem::IsVehicleBlinkerOn( void )
00070 {
00071
          bool isBlinkerOn = false:
00072
00073
          if ( this->turnSignal[vehicleType]->isRightBlinkerOn ||
               this->turnSignal[vehicleType]->isLeftBlinkerOn )
00074
00075
00076
              isBlinkerOn = true;
00077
00078
00079
          return isBlinkerOn:
00080 }
00081
00083 bool MotorcycleAwarenessSystem::IsMotorcycleInRange( void )
00084 {
00085
          bool isInRange = false;
00086
          // Determine where the motorcycle is relative to the car using the {\ \ \ } GPS signals
00087
00088
          if ( abs( (this->carGpsSignal->x) - (this->motorcycleGpsSignal->x) ) <=</pre>
      SAFETY ZONE &&
00089
               abs( (this->carGpsSignal->y) - (this->motorcycleGpsSignal->y) > <=
      SAFETY_ZONE )
00090
00091
              isInRange = true;
00092
00093
          else
00094
00095
              // Analyze the V2V data for a threat
00096
              MotorCycleLocation_t motorCycleLocation =
     GetMotorcycleLocation();
00098
00099
00100
          return isInRange;
00101 }
00102
00105 bool MotorcycleAwarenessSystem::IsHazardPresent( void )
00106 {
00107
          bool isHazardPresent = false;
00108
00109
          if ( abs( this->motorcycleRadarSignal->objectDistance ) <=</pre>
      SAFETY_ZONE )
00110
          {
00111
              isHazardPresent = true;
00112
00113
00114
          return isHazardPresent;
00115 }
00116
00118 void MotorcycleAwarenessSystem::TrackMotorcycle( void )
00119 {
00120
          // Push the motorcycle's GPS location onto the list
00121
          motorcycleLocation.push_front( *motorcycleGpsSignal );
00122 }
00123
00125 void MotorcycleAwarenessSystem::RelayWarningToOperator(
      void )
00126 {
```

```
00127
          // Assemble the message to be sent
00128
          BlueToothMessage_t blueToothMessage;
00129
         blueToothMessage.isHazardPresent = true;
00130
          // Dummy message
00131
         blueToothMessage.dataBuffer[0] = 0x2015;
00132
00134 }
00135
00136 // Method to Get motorcycle's location via V2V communication
00137 MotorCycleLocation_t
     MotorcycleAwarenessSystem::GetMotorcycleLocation(void)
00138 {
00139
          // Dummy motorcycle location from V2V communication
00140
         MotorCycleLocation_t motorCycleLocation;
00142
00143
          return motorCycleLocation;
00144 }
```

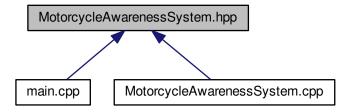
#### 6.6 MotorcycleAwarenessSystem.hpp File Reference

```
#include "MotorcycleAwarenessSystemTypes.hpp"
#include <string>
#include <iostream>
```

Include dependency graph for MotorcycleAwarenessSystem.hpp:



This graph shows which files directly or indirectly include this file:



#### Classes

class MotorcycleAwarenessSystem

Class declaration for the Motorcycle Awareness System (MAS)

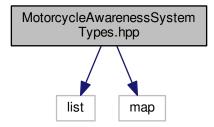
#### 6.7 MotorcycleAwarenessSystem.hpp

```
00001 #ifndef MOTORCYCLEAWARENESSSYSTEM_HPP
00005 #define MOTORCYCLEAWARENESSSYSTEM_HPP
00007 #include "MotorcycleAwarenessSystemTypes.hpp"
00009 #include <string>
00010 #include <iostream>
00011
00012 class MotorcycleAwarenessSystem
00013 {
00014
          public:
00015
              MotorcycleAwarenessSystem( VehicleType
      vehicleType );
00016
              ~MotorcycleAwarenessSystem( void );
00017
              void Initialize( TurnSignal_t* motorcycleTurnSignal,
      TurnSignal_t* carTurnSignal,
00018
                               RadarSignal_t* motorcycleRadarSignal,
      GpsSignal_t* motorcycleGpsSignal,
00019
                               GpsSignal_t* carGpsSignal );
00020
              void MonitorConditions( void );
00021
00022
          private:
              std::list<GpsSignal_t> motorcycleLocation;
00023
00024
              static const unsigned int SAFETY_ZONE = 15U;
              VehicleType vehicleType;
00025
00027
              std::map<VehicleType, TurnSignal_t*> turnSignal;
00028
              RadarSignal_t* motorcycleRadarSignal;
              GpsSignal_t* motorcycleGpsSignal;
00029
00030
              GpsSignal_t* carGpsSignal;
00031
              bool IsVehicleBlinkerOn( void );
00032
00033
              bool IsMotorcycleInRange( void );
00034
              bool IsHazardPresent( void );
00035
              void TrackMotorcycle( void );
00036
              void RelayWarningToOperator( void );
00037
              MotorCycleLocation_t GetMotorcycleLocation( void );
00038 };
00039 #endif
```

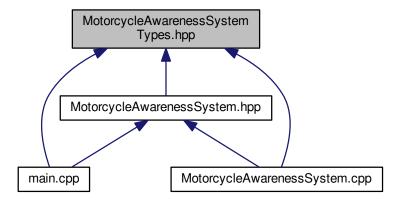
#### 6.8 MotorcycleAwarenessSystemTypes.hpp File Reference

```
#include <list>
#include <map>
```

Include dependency graph for MotorcycleAwarenessSystemTypes.hpp:



This graph shows which files directly or indirectly include this file:



#### Classes

- struct TurnSignal\_t
- struct GpsSignal\_t

Structure that emulates a GPS signal.

struct RadarSignal\_t

Structure that emulates a Radar signal.

struct BlueToothMessage\_t

Struct for bluetooth message.

struct MotorCycleLocation\_t

Struct for the V2V data.

#### **Typedefs**

• typedef float Coordinate\_t

GPS coordinates.

- typedef float currentTime t
- · typedef float ObjectDistance

Distance of an object from a detecting radar sensor.

- typedef float ObjectVelocity
- typedef int DataPacket\_t [255]

#### **Enumerations**

enum VehicleType { MOTORCYCLE, CAR }
 Enum for vehicle type.

#### 6.8.1 Typedef Documentation

#### 6.8.1.1 typedef float Coordinate\_t

GPS coordinates.

Definition at line 22 of file MotorcycleAwarenessSystemTypes.hpp.

6.8.1.2 typedef float currentTime\_t

Current time for a pair of GPS coordinates

Definition at line 23 of file MotorcycleAwarenessSystemTypes.hpp.

6.8.1.3 typedef int DataPacket\_t[255]

Data packet for the V2V communication

Definition at line 48 of file MotorcycleAwarenessSystemTypes.hpp.

6.8.1.4 typedef float ObjectDistance

Distance of an object from a detecting radar sensor.

Definition at line 32 of file MotorcycleAwarenessSystemTypes.hpp.

6.8.1.5 typedef float ObjectVelocity

Velocity of an object detected by a radar sensor

Definition at line 33 of file MotorcycleAwarenessSystemTypes.hpp.

#### 6.8.2 Enumeration Type Documentation

#### 6.8.2.1 enum VehicleType

Enum for vehicle type.

Enumerator

## MOTORCYCLE

CAR

Definition at line 8 of file MotorcycleAwarenessSystemTypes.hpp.

```
00009 {
00010 MOTORCYCLE,
00011 CAR
00012 };
```

#### 6.9 MotorcycleAwarenessSystemTypes.hpp

```
00001 #ifndef MOTORCYCLEAWARENESSSYSTEMTYPES_HPP
00002 #define MOTORCYCLEAWARENESSSYSTEMTYPES_HPP
00003
00004 #include <list>
00005 #include <map>
00006
00008 enum VehicleType
00009 {
00010
          MOTORCYCLE,
00011
          CAR
00012 };
00013
00016 struct TurnSignal_t
00017 {
00018
          bool isRightBlinkerOn;
          bool isLeftBlinkerOn;
00019
00020 };
00021
00022 typedef float Coordinate_t;
00023 typedef float currentTime_t;
00024 struct GpsSignal_t
00026 {
          Coordinate_t x;
00027
          Coordinate_t y;
currentTime_t currentTime;
00028
00029
00030 };
00031
00032 typedef float ObjectDistance;
00033 typedef float ObjectVelocity;
00034 struct RadarSignal_t
00036 {
00037
          ObjectDistance objectDistance;
00038
          ObjectVelocity objectVelocity;
00039 };
00040
00042 struct BlueToothMessage_t
00043 {
00044
          bool isHazardPresent;
00045
          unsigned int dataBuffer[255];
00046 };
00047
00048 typedef int DataPacket_t[255];
00049 struct MotorCycleLocation_t
00051 {
00052
          DataPacket_t dataPacket;
00053 };
00054 #endif
```

## Index

$\sim$ MotorcycleAwarenessSystem	main, 17
MotorcycleAwarenessSystem, 7	mainpage.dox, 18
	MonitorConditions
BlueToothMessage_t, 3	MotorcycleAwarenessSystem, 10
dataBuffer, 3	MotorCycleLocation_t, 13
isHazardPresent, 3	dataPacket, 13
	MotorcycleAwarenessSystemTypes.hpp
CAR	CAR, 25
MotorcycleAwarenessSystemTypes.hpp, 25	MOTORCYCLE, 25
carGpsSignal	MotorcycleAwarenessSystem, 5
MotorcycleAwarenessSystem, 12	~MotorcycleAwarenessSystem, 7
Coordinate_t	carGpsSignal, 12
MotorcycleAwarenessSystemTypes.hpp, 24	GetMotorcycleLocation, 7
currentTime	Initialize, 7
GpsSignal_t, 5	IsHazardPresent, 8
currentTime_t	IsMotorcycleInRange, 8
MotorcycleAwarenessSystemTypes.hpp, 24	IsVehicleBlinkerOn, 9
	MonitorConditions, 10
dataBuffer	MotorcycleAwarenessSystem, 7
BlueToothMessage_t, 3	motorcycleGpsSignal, 12
dataPacket	motorcycleCpsSignal, 12
MotorCycleLocation_t, 13	motorcycleRadarSignal, 12
DataPacket_t	
MotorcycleAwarenessSystemTypes.hpp, 24	MotorcycleAwarenessSystem, 7
	RelayWarningToOperator, 11
GetMotorcycleLocation	SAFETY_ZONE, 12
MotorcycleAwarenessSystem, 7	TrackMotorcycle, 11
GpsSignal_t, 4	turnSignal, 12
currentTime, 5	vehicleType, 12
x, 5	MotorcycleAwarenessSystem.cpp, 18
y, 5	MotorcycleAwarenessSystem.hpp, 21
	MotorcycleAwarenessSystemTypes.hpp, 23
Initialize	Coordinate_t, 24
MotorcycleAwarenessSystem, 7	currentTime_t, 24
IsHazardPresent	DataPacket_t, 24
MotorcycleAwarenessSystem, 8	ObjectDistance, 24
isHazardPresent	ObjectVelocity, 24
BlueToothMessage_t, 3	VehicleType, 24
isLeftBlinkerOn	motorcycleGpsSignal
TurnSignal_t, 16	MotorcycleAwarenessSystem, 12
IsMotorcycleInRange	motorcycleLocation
MotorcycleAwarenessSystem, 8	MotorcycleAwarenessSystem, 12
isRightBlinkerOn	motorcycleRadarSignal
TurnSignal_t, 16	MotorcycleAwarenessSystem, 12
IsVehicleBlinkerOn	
MotorcycleAwarenessSystem, 9	ObjectDistance
•	MotorcycleAwarenessSystemTypes.hpp, 24
MOTORCYCLE	objectDistance
MotorcycleAwarenessSystemTypes.hpp, 25	RadarSignal_t, 14
main	ObjectVelocity
main.cpp, 17	MotorcycleAwarenessSystemTypes.hpp, 24
main.cpp, 16	objectVelocity

INDEX 27

```
RadarSignal_t, 14
RadarSignal_t, 14
    objectDistance, 14
    objectVelocity, 14
RelayWarningToOperator
    MotorcycleAwarenessSystem, 11
SAFETY_ZONE
    MotorcycleAwarenessSystem, 12
TrackMotorcycle
    MotorcycleAwarenessSystem, 11
turnSignal
    MotorcycleAwarenessSystem, 12
TurnSignal_t, 15
    isLeftBlinkerOn, 16
    isRightBlinkerOn, 16
VehicleType
    MotorcycleAwarenessSystemTypes.hpp, 24
vehicleType
    MotorcycleAwarenessSystem, 12
Х
    GpsSignal_t, 5
У
    GpsSignal_t, 5
```