Motorcycle Awareness System (MAS)

ME553 Group EPE 6

Generated by Doxygen 1.8.6

Sun Apr 26 2015 18:16:07

Contents

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

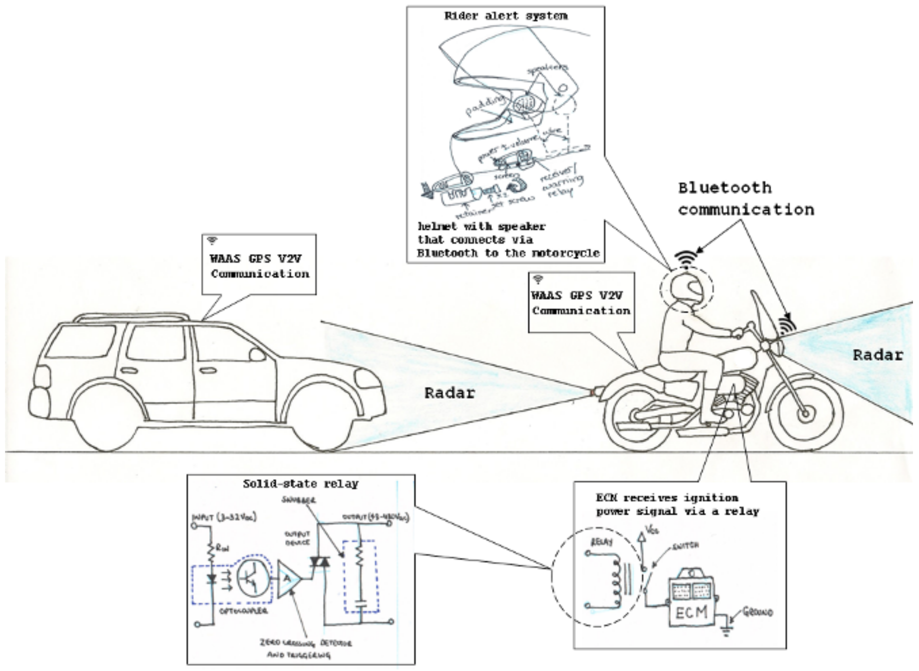

Rider alert system
helmet with speaker that connects via Bluetooth to the motorcycle
Bluetooth communication
WAAS GPS V2V Communication
WAAS GPS V2V Communication
Radar
Radar
Solid-state relay
ECM receives ignition power signal via a relay
RELAY
SWITCH
ECM
GROUND

Figure 1: MAS concept sketch

2 Todo List

Member MotorcycleAwarenessSystem::GetMotorcycleLocation (void)
Acquire the data packet from the motorcycle

Member MotorcycleAwarenessSystem::IsMotorcycleInRange (void)
Process the data packet and determine threat

Member MotorcycleAwarenessSystem::RelayWarningToOperator (void)
Transmit the bluetooth message to the operator

3 Class Index

3.1 Class List

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

4 File Index

4.1 File List

Here is a list of all files with brief descriptions:

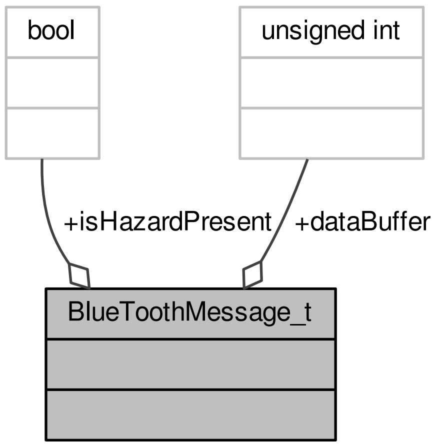
5 Class Documentation

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:

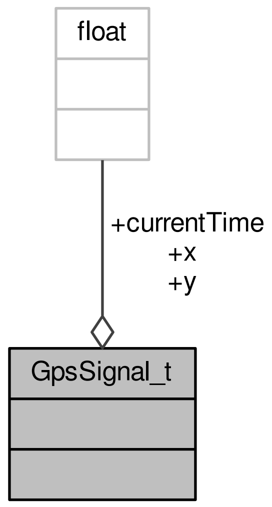
- MotorcycleAwarenessSystemTypes.hpp

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:

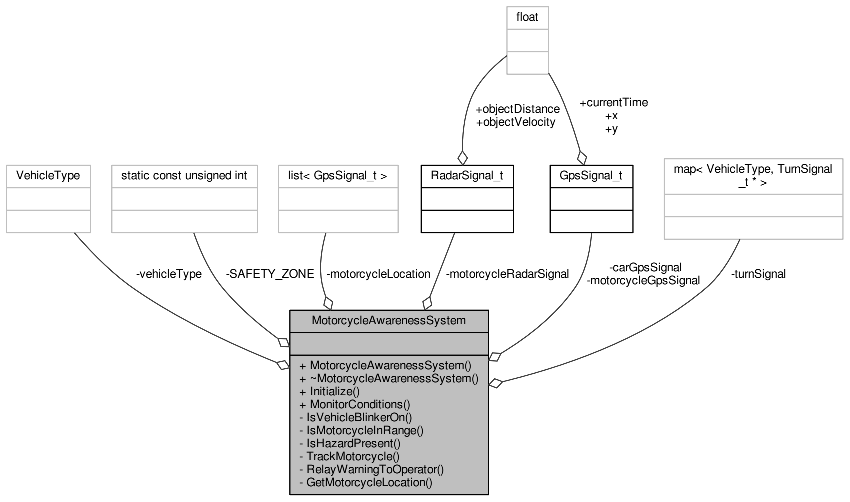
- MotorcycleAwarenessSystemTypes.hpp

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 ∗motorcycle-RadarSignal, GpsSignal_t ∗motorcycleGpsSignal, GpsSignal_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,
  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.

```
    :vehicleType( vehicleType )
{
    // Do nothing
}
```

5.3.2.2 MotorcycleAwarenessSystem::∼MotorcycleAwarenessSystem (void)

Destructor.

Definition at line 20 of file MotorcycleAwarenessSystem.cpp.

```
{
    // Do nothing
}
```

5.3.3 Member Function Documentation

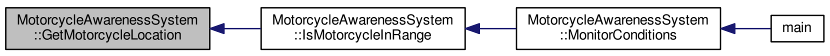
5.3.3.1 MotorCycleLocation_t MotorcycleAwarenessSystem::GetMotorcycleLocation (void) [private]

Todo Acquire the data packet from the motorcycle

Definition at line 137 of file MotorcycleAwarenessSystem.cpp.

```
{
    // Dummy motorcycle location from V2V communication
    MotorCycleLocation_t motorCycleLocation;

    return motorCycleLocation;
}
```

Here is the caller graph for this function:


MotorcycleAwarenessSystem ::GetMotorcycleLocation
MotorcycleAwarenessSystem ::IsMotorcycleInRange
MotorcycleAwarenessSystem ::MonitorConditions
main

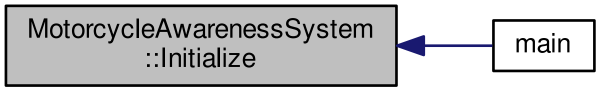
5.3.3.2 void MotorcycleAwarenessSystem::Initialize (TurnSignal_t ∗ *motorcycleTurnSignal,* TurnSignal_t ∗ *carTurnSignal,* RadarSignal_t ∗ *motorcycleRadarSignal,* GpsSignal_t ∗ *motorcycleGpsSignal,* GpsSignal_t ∗ *carGpsSignal*)

Method to initialize the MAS system.

Definition at line 26 of file MotorcycleAwarenessSystem.cpp.

```
{
    // Initialize the motorcycle radar signal
    this->motorcycleRadarSignal = motorcycleRadarSignal;

    // Initialize the turn signal map
    turnSignal[MOTORCYCLE] = motorcycleTurnSignal;
    turnSignal[CAR] = carTurnSignal;

    // Initialize the GPS signals
    this->motorcycleGpsSignal = motorcycleGpsSignal;
    this->motorcycleGpsSignal = motorcycleGpsSignal;
}
```

Here is the caller graph for this function:



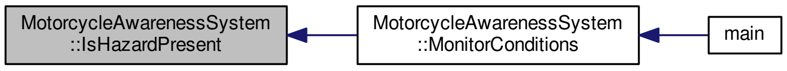
5.3.3.3 bool MotorcycleAwarenessSystem::IsHazardPresent (void) [private]

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.

```
{
    bool isHazardPresent = false;

    if ( abs( this->motorcycleRadarSignal->objectDistance ) <=
      SAFETY_ZONE )
    {
        isHazardPresent = true;
    }

    return isHazardPresent;
}
```

Here is the caller graph for this function:


MotorcycleAwarenessSystem ::IsHazardPresent
MotorcycleAwarenessSystem ::MonitorConditions
main

5.3.3.4 bool MotorcycleAwarenessSystem::IsMotorcycleInRange (void) [private]

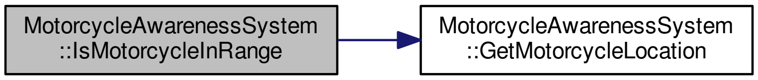
Method to determine whether the motorcycle is within the car's range.

Todo Process the data packet and determine threat

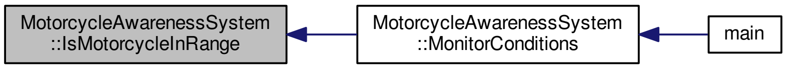
Definition at line 83 of file MotorcycleAwarenessSystem.cpp.

```
{
    bool isInRange = false;

    // Determine where the motorcycle is relative to the car using the GPS signals
    if ( abs( (this->carGpsSignal->x) - (this->motorcycleGpsSignal->x) ) <=
 SAFETY_ZONE &&
         abs( (this->carGpsSignal->y) - (this->motorcycleGpsSignal->y) ) <=
 SAFETY_ZONE )
    {
        isInRange = true;
    }
    else
    {
        // Analyze the V2V data for a threat
        MotorCycleLocation_t motorCycleLocation =
 GetMotorcycleLocation();
    }

    return isInRange;
}
```

Here is the call graph for this function:


MotorcycleAwarenessSystem ::IsMotorcycleInRange
MotorcycleAwarenessSystem ::GetMotorcycleLocation

Here is the caller graph for this function:


MotorcycleAwarenessSystem ::IsMotorcycleInRange
MotorcycleAwarenessSystem ::MonitorConditions
main

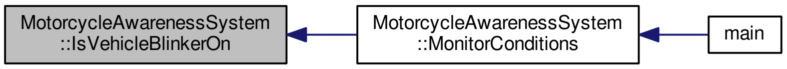
5.3.3.5 bool MotorcycleAwarenessSystem::IsVehicleBlinkerOn (void) [private]

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

Definition at line 69 of file MotorcycleAwarenessSystem.cpp.

```
{
    bool isBlinkerOn = false;

    if ( this->turnSignal[vehicleType]->isRightBlinkerOn ||
         this->turnSignal[vehicleType]->isLeftBlinkerOn )
    {
        isBlinkerOn = true;
    }

    return isBlinkerOn;
}
```

Here is the caller graph for this function:


MotorcycleAwarenessSystem ::IsVehicleBlinkerOn
MotorcycleAwarenessSystem ::MonitorConditions
main

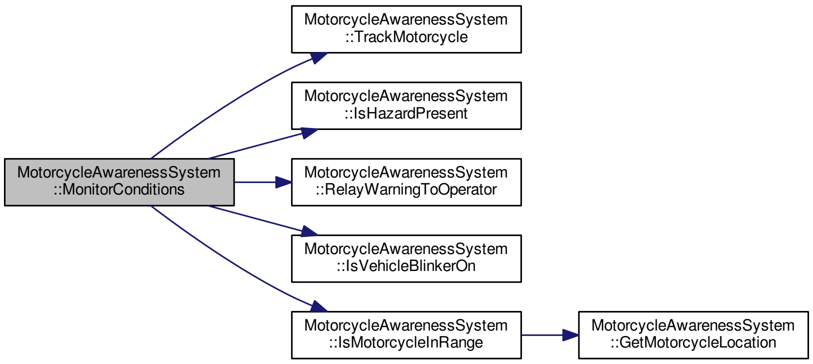
5.3.3.6 void MotorcycleAwarenessSystem::MonitorConditions (void)

Method to continuously monitor the conditions during run-time.

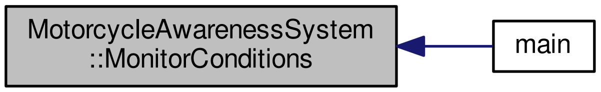
Definition at line 43 of file MotorcycleAwarenessSystem.cpp.

```
{
    if ( MOTORCYCLE == vehicleType )
    {
        // Track the motorcycle
        TrackMotorcycle();
        // Check for hazards
        if ( IsHazardPresent() == true )
        {
            // Warn the motorcycle operator
            RelayWarningToOperator();
        }
    }
    // vehicleType == CAR
    else
    {
        // Check for hazards
        if ( (IsVehicleBlinkerOn() == true) && (
IsMotorcycleInRange() == true) )
        {
            // Relay message to the car driver
            RelayWarningToOperator();
        }
    }
}
```

Here is the call graph for this function:


MotorcycleAwarenessSystem ::MonitorConditions
MotorcycleAwarenessSystem ::TrackMotorcycle
MotorcycleAwarenessSystem ::IsHazardPresent
MotorcycleAwarenessSystem ::RelayWarningToOperator
MotorcycleAwarenessSystem ::IsVehicleBlinkerOn
MotorcycleAwarenessSystem ::IsMotorcycleInRange
MotorcycleAwarenessSystem ::GetMotorcycleLocation

Here is the caller graph for this function:



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

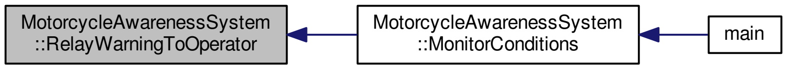
Method to relay a warning to operator via bluetooth connectivity.

Todo Transmit the bluetooth message to the operator

Definition at line 125 of file MotorcycleAwarenessSystem.cpp.

```
{
    // Assemble the message to be sent
    BlueToothMessage_t blueToothMessage;
    blueToothMessage.isHazardPresent = true;
    // Dummy message
    blueToothMessage.dataBuffer[0] = 0x2015;

}
```

Here is the caller graph for this function:


MotorcycleAwarenessSystem ::RelayWarningToOperator
MotorcycleAwarenessSystem ::MonitorConditions
main

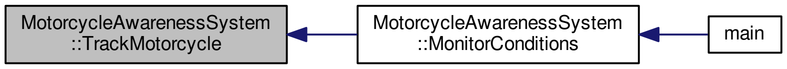
5.3.3.8 void MotorcycleAwarenessSystem::TrackMotorcycle (void) [private]

Method to track the motorcycle using its GPS signal.

Definition at line 118 of file MotorcycleAwarenessSystem.cpp.

```
{
    // Push the motorcycle's GPS location onto the list
    motorcycleLocation.push_front( *motorcycleGpsSignal );
}
```

Here is the caller graph for this function:


MotorcycleAwarenessSystem ::TrackMotorcycle
MotorcycleAwarenessSystem ::MonitorConditions
main

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:

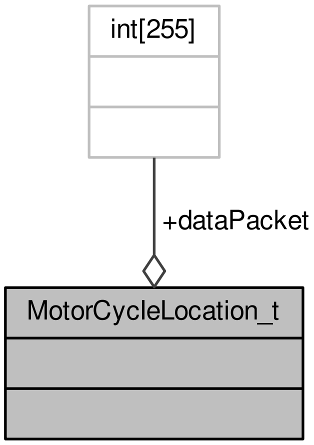
- MotorcycleAwarenessSystem.hpp
- MotorcycleAwarenessSystem.cpp

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:

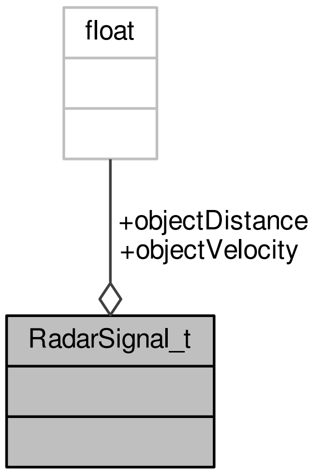
- MotorcycleAwarenessSystemTypes.hpp

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.

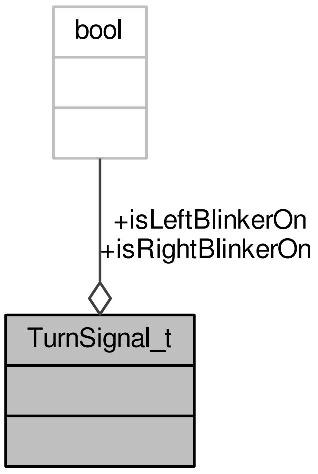
The documentation for this struct was generated from the following file:

- 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:

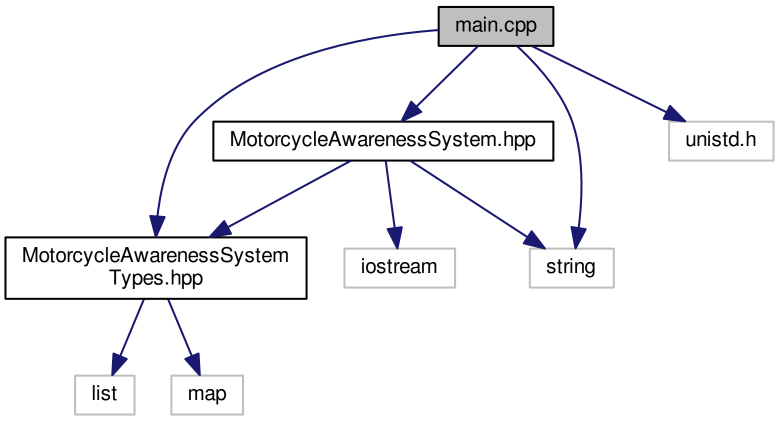
- MotorcycleAwarenessSystemTypes.hpp

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:


main.cpp
MotorcycleAwarenessSystem.hpp
unistd.h
MotorcycleAwarenessSystem Types.hpp
iostream
string
list
map

Functions

- int main (void)

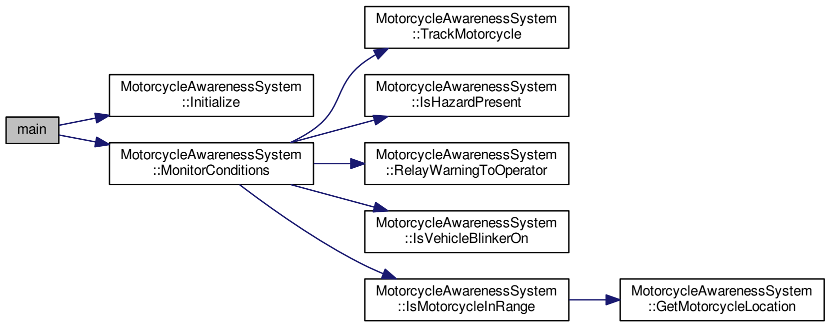
6.1.1 Function Documentation

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     // 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     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;
00040 #endif
00041     }
00042
00043     return 0;
00044 }
```

Here is the call graph for this function:


main
MotorcycleAwarenessSystem ::Initialize
MotorcycleAwarenessSystem ::MonitorConditions
MotorcycleAwarenessSystem ::TrackMotorcycle
MotorcycleAwarenessSystem ::IsHazardPresent
MotorcycleAwarenessSystem ::RelayWarningToOperator
MotorcycleAwarenessSystem ::IsVehicleBlinkerOn
MotorcycleAwarenessSystem ::IsMotorcycleInRange
MotorcycleAwarenessSystem ::GetMotorcycleLocation

6.2 main.cpp

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

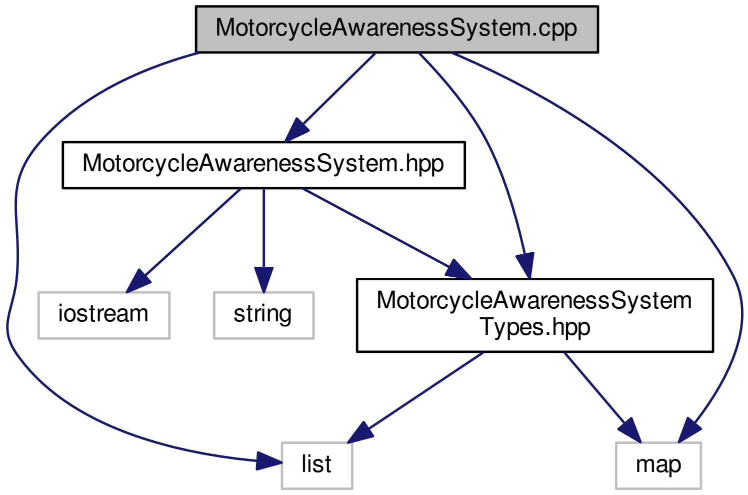
```
#include <iostream>
#endif
#include <string>
#include <unistd.h>

int main( void )
{
    // Define mock signals
    TurnSignal_t* motorcycleTurnSignal = new TurnSignal_t;
    TurnSignal_t* carTurnSignal = new TurnSignal_t;
    RadarSignal_t* motorcycleRadarSignal = new RadarSignal_t;
    GpsSignal_t* motorcycleGpsSignal = new GpsSignal_t;
    GpsSignal_t* carGpsSignal = new GpsSignal_t;;

    // Instantiate the Motorcycle Awareness System (MAS) for the motorcycle
    MotorcycleAwarenessSystem MAS_motorcycle(
MOTORCYCLE );
    // Initialize the car MAS
    MAS_motorcycle.Initialize( motorcycleTurnSignal, (TurnSignal_t*)NULL,
motorcycleRadarSignal,
                                          motorcycleGpsSignal, carGpsSignal );

    // Instantiate the Motorcycle Awareness System (MAS) for the car
    MotorcycleAwarenessSystem MAS_car( CAR );
    // Initialize the car MAS
    MAS_car.Initialize( (TurnSignal_t*)NULL, carTurnSignal, motorcycleRadarSignal,
                                          motorcycleGpsSignal, carGpsSignal );

    // Run the MAS systems for the car & motorcycle
    while ( true )
    {
        MAS_motorcycle.MonitorConditions();
        MAS_car.MonitorConditions();
        usleep(250);

#ifdef MAS_DEBUG
        std::cout << "The MAS is running" << std::endl;
#endif
    }

    return 0;
}
```

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:


MotorcycleAwarenessSystem.cpp
MotorcycleAwarenessSystem.hpp
iostream
string
MotorcycleAwarenessSystem Types.hpp
list
map

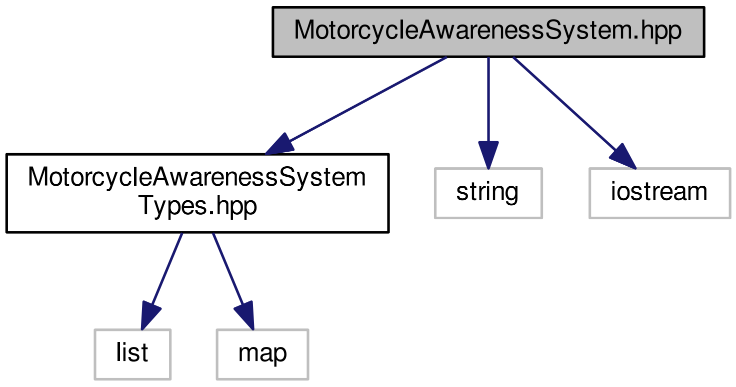
6.5 MotorcycleAwarenessSystem.cpp

```
#include "MotorcycleAwarenessSystem.hpp"
#include "MotorcycleAwarenessSystemTypes.hpp"

#include <list>
#include <map>

MotorcycleAwarenessSystem::MotorcycleAwarenessSystem(
VehicleType vehicleType )
    :vehicleType( vehicleType )
{
    // Do nothing
}

MotorcycleAwarenessSystem::~MotorcycleAwarenessSystem(
 void )
{
    // Do nothing
}

void MotorcycleAwarenessSystem::Initialize(
TurnSignal_t* motorcycleTurnSignal, TurnSignal_t* carTurnSignal,
                                            RadarSignal_t* motorcycleRadarSignal,
GpsSignal_t* motorcycleGpsSignal,
                                            GpsSignal_t* carGpsSignal  )
{
    // Initialize the motorcycle radar signal
    this->motorcycleRadarSignal = motorcycleRadarSignal;

    // Initialize the turn signal map
    turnSignal[MOTORCYCLE] = motorcycleTurnSignal;
    turnSignal[CAR] = carTurnSignal;

    // Initialize the GPS signals
    this->motorcycleGpsSignal = motorcycleGpsSignal;
    this->motorcycleGpsSignal = motorcycleGpsSignal;
}

void MotorcycleAwarenessSystem::MonitorConditions( void )
{
    if ( MOTORCYCLE == vehicleType )
    {
        // Track the motorcycle
        TrackMotorcycle();
        // Check for hazards
        if ( IsHazardPresent() == true )
        {
            // Warn the motorcycle operator
            RelayWarningToOperator();
        }
    }
    // vehicleType == CAR
    else
    {
        // Check for hazards
        if ( (IsVehicleBlinkerOn() == true) && ( IsMotorcycleInRange() == true) )
        {
            // Relay message to the car driver
            RelayWarningToOperator();
        }
    }
}

bool MotorcycleAwarenessSystem::IsVehicleBlinkerOn( void )
{
    bool isBlinkerOn = false;

    if ( this->turnSignal[vehicleType]->isRightBlinkerOn ||
         this->turnSignal[vehicleType]->isLeftBlinkerOn )
    {
        isBlinkerOn = true;
    }

    return isBlinkerOn;
}

bool MotorcycleAwarenessSystem::IsMotorcycleInRange( void )
{
    bool isInRange = false;

    // Determine where the motorcycle is relative to the car using the GPS signals
    if ( abs( (this->carGpsSignal->x) - (this->motorcycleGpsSignal->x) ) <= SAFETY_ZONE &&
         abs( (this->carGpsSignal->y) - (this->motorcycleGpsSignal->y) ) <= SAFETY_ZONE )
    {
        isInRange = true;
    }
    else
    {
        // Analyze the V2V data for a threat
        MotorCycleLocation_t motorCycleLocation = GetMotorcycleLocation();
    }

    return isInRange;
}

bool MotorcycleAwarenessSystem::IsHazardPresent( void )
{
    bool isHazardPresent = false;

    if ( abs( this->motorcycleRadarSignal->objectDistance ) <= SAFETY_ZONE )
    {
        isHazardPresent = true;
    }

    return isHazardPresent;
}

void MotorcycleAwarenessSystem::TrackMotorcycle( void )
{
    // Push the motorcycle's GPS location onto the list
    motorcycleLocation.push_front( *motorcycleGpsSignal );
}

void MotorcycleAwarenessSystem::RelayWarningToOperator( void )
{
    // Assemble the message to be sent
    BlueToothMessage_t blueToothMessage;
    blueToothMessage.isHazardPresent = true;
    // Dummy message
    blueToothMessage.dataBuffer[0] = 0x2015;

}

// Method to Get motorcycle's location via V2V communication
MotorCycleLocation_t
MotorcycleAwarenessSystem::GetMotorcycleLocation( void )
{
    // Dummy motorcycle location from V2V communication
    MotorCycleLocation_t motorCycleLocation;

    return motorCycleLocation;
}
```

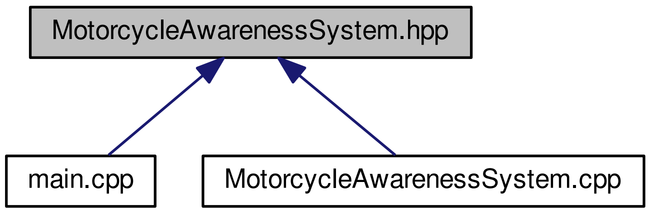
6.6 MotorcycleAwarenessSystem.hpp File Reference

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

Include dependency graph for MotorcycleAwarenessSystem.hpp:


MotorcycleAwarenessSystem.hpp
MotorcycleAwarenessSystem Types.hpp
string
iostream
list
map

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



Classes

- class MotorcycleAwarenessSystem

  Class declaration for the Motorcycle Awareness System (MAS)

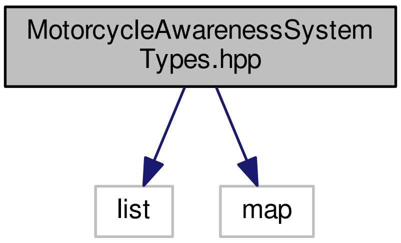
6.7 MotorcycleAwarenessSystem.hpp

```
#ifndef MOTORCYCLEAWARENESSSYSTEM_HPP
#define MOTORCYCLEAWARENESSSYSTEM_HPP

#include "MotorcycleAwarenessSystemTypes.hpp"

#include <string>
#include <iostream>

class MotorcycleAwarenessSystem
{
    public:
        MotorcycleAwarenessSystem( VehicleType
vehicleType );
        ~MotorcycleAwarenessSystem( void );
        void Initialize( TurnSignal_t* motorcycleTurnSignal,
TurnSignal_t* carTurnSignal,
                         RadarSignal_t* motorcycleRadarSignal,
GpsSignal_t* motorcycleGpsSignal,
                         GpsSignal_t* carGpsSignal );
        void MonitorConditions( void );

    private:
        std::list<GpsSignal_t> motorcycleLocation;
        static const unsigned int SAFETY_ZONE = 15U;
        VehicleType vehicleType;
        std::map<VehicleType, TurnSignal_t*> turnSignal;
        RadarSignal_t* motorcycleRadarSignal;
        GpsSignal_t* motorcycleGpsSignal;
        GpsSignal_t* carGpsSignal;

        bool IsVehicleBlinkerOn( void );
        bool IsMotorcycleInRange( void );
        bool IsHazardPresent( void );
        void TrackMotorcycle( void );
        void RelayWarningToOperator( void );
        MotorCycleLocation_t GetMotorcycleLocation( void );
};
#endif
```

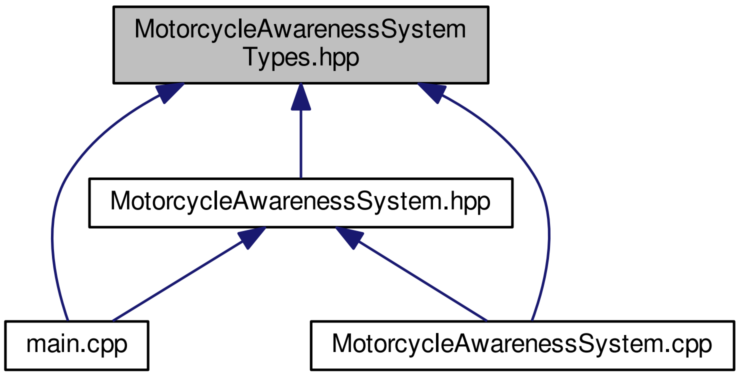
6.8 MotorcycleAwarenessSystemTypes.hpp File Reference

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

Include dependency graph for MotorcycleAwarenessSystemTypes.hpp:



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


MotorcycleAwarenessSystem
Types.hpp
MotorcycleAwarenessSystem.hpp
main.cpp
MotorcycleAwarenessSystem.cpp

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;
00019     bool isLeftBlinkerOn;
00020 };
00021
00022 typedef float Coordinate_t;
00023 typedef float currentTime_t;
00024 struct GpsSignal_t
00026 {
00027     Coordinate_t x;
00028     Coordinate_t y;
00029     currentTime_t currentTime;
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