POPWin

Generated by Doxygen 1.8.8

Tue Jun 30 2015 13:24:00

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 example_neighbor Struct Reference

Collaboration diagram for example_neighbor:

Public Attributes

- struct example_neighbor * **next**
- rimeaddr_t **addr**
- struct ctimer **ctimer**

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

- gatewayMote/example-multihop.c

4.2 Message Struct Reference

```
#include <queue.h>
```

Public Attributes

- uint8_t **type**
- uint8_t **tag**
- uint8_t **message**
- uint8_t **nodeid**
- uint8_t **value**
- uint8_t **weight**
- char **message_string** [64]

4.2.1 Detailed Description

This sample is about how to implement a queue in c

Type of item is int Add item to tail Get item from head Can get the size Can display all content

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

- gatewayMote/queue.h

4.3 neighbor Struct Reference

Collaboration diagram for neighbor:

Public Attributes

- struct neighbor * **next**
- rimeaddr_t **addr**
- uint8_t **tag**
- uint8_t **weight**

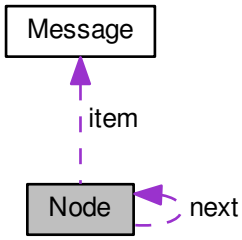
The documentation for this struct was generated from the following file:

- gatewayMote/DRW.h

4.4 Node Struct Reference

`#include <queue.h>`

Collaboration diagram for Node:



Public Attributes

- struct Message **item**
- struct Node * **next**

4.4.1 Detailed Description

The Node struct, contains item and the pointer that point to next node.

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

- gatewayMote/queue.h

4.5 NotifyMessage Struct Reference

A structure representing a notification message.

`#include <popwin_messages.h>`

Public Attributes

- enum MeasurementType measurementType

  Type of measurement.
- enum DataType dataType

  Type of data.
- unsigned short id

  Id of emitter: Id is not mandatory. Only for convenience.
- enum MeasurementUnit unit

  Unit of measurement.
- size_t dataSize

  Size of the data, for storage in buffer.
- char data [BUFFERDATASIZE]

  Buffer containing the data on text format.

4.5.1 Detailed Description

A structure representing a notification message.

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

- gatewayMote/popwin_messages.h

4.6 POPSensor Class Reference

A POPSensor object organizes communication with the remote sensors by creating a set of SensorProxies. This is transparent to the user.

`#include <POPSensor.h>`

Public Member Functions

- POPSensor (const std::string &x_url, const std::string &x_resourceFileName)@

  Constructor (URL of target platteform is specified)
- POPSensor (int x_pow, const std::string &x_resourceFileName)@

  Constructor (power requirement of target platteform is specified)
- ~POPSensor ()

  Destructor.
- POPSensorData Gather ()

  Retrieve data gathered.
- void Broadcast (int x_publicationType, int x_data)

  Broacast data through the network.
- void Broadcast (int x_publicationType, double x_data)

  Broadcast a message to all sensors.
- void Broadcast (int x_publicationType, const std::string &x_data)

  Broadcast a message to all sensors.
- double Reduce (int x_mtype, int x_dataType, int x_fct)

  Apply a reduce operation to the stored data {size, min, max, aver, sum, stdev}.
- void Clear ()

  Clear data gathered.
- void Notify (int x_measurementType, int x_measurementUnit, const std::string &x_message)

  Methods specific to the POPWin project.
- void Subscribe (int x_measurementType, int x_dataType)

  Send a subscription to sensors.
- bool IsConnected ()

  Check if connected to any sensor.
- int GetDataSize ()

  Return the size of the stored data.

4.6.1 Detailed Description

A POPSensor object organizes communication with the remote sensors by creating a set of SensorProxies. This is transparent to the user.

4.6.2 Member Function Documentation

4.6.2.1 void POPSensor::Broadcast (int *x_publicationType*, int *x_data*)

Broacast data through the network.

Broadcast a message to all sensors.

4.6.2.2 void POPSensor::Clear ( )

Clear data gathered.

Clear the stored messages.

4.6.2.3 POPSensorData POPSensor::Gather ( )

Retrieve data gathered.

Return a POPSensorData structure containing the messages received from sensors.

4.6.2.4 bool POPSensor::IsConnected ( )

Check if connected to any sensor.

Return true if the POPSensor is connected to at least one sensor.

4.6.2.5 void POPSensor::Notify (int *x_measurementType*, int *x_measurementUnit*, const std::string & *x_message*)

Methods specific to the POPWin project.

Send a notification to all sensors.

Send a publication to sensors Send notification to the connected sensor

4.6.2.6 void POPSensor::Subscribe (int *x_measurementType*, int *x_dataType*)

Send a subscription to sensors.

Subscribe to messages of given type and data type.

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

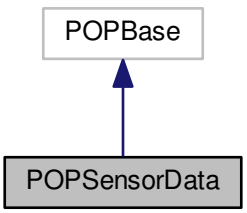
- POPSensor.h
- POPSensor.cc

4.7 POPSensorData Class Reference

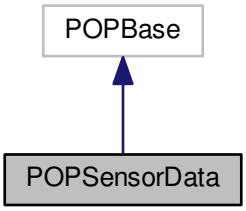
A serializable object that can store the results of data acquisition.

`#include <POPSensorData.h>`

Inheritance diagram for POPSensorData:



Collaboration diagram for POPSensorData:



Public Types

- enum reduceFunctions {
  size, **min**, **max**, **aver**,
  sum, **stdev** }

  The different reduce functions.
- typedef enum
  POPSensorData::reduceFunctions POPReduceF

  The different reduce functions.

Public Member Functions

- void Serialize (POPBuffer &buf, bool pack)

  Serialize and deserialize the class.
- void Print ()

  Print the data to stdout.
- void PrintToFile (std::ostream &xr_ostream)

  Print the data to a .csv file.
- void Clear ()

  Clear the data.
- int GetSize () const

  Return the size of the data.
- void Insert (const POPSensorData &)

  Insert a new record into the data.
- template<typename T >
  void Insert (std::pair< RecordHeader, T > &x_pair)

  Insert a new record into the data.
- template<typename T >
  double Reduce (enum MeasurementType x_mtype, POPReduceF x_fct) const
- template<>
  std::map< RecordHeader, int > & RefData ()

  Return a reference to the data.
- template<>
  std::map< RecordHeader, double > & RefData ()

  Return a reference to the data.

- template<>
  std::map< RecordHeader, string > & RefData ()

  Return a reference to the data.
- template<>
  const std::map< RecordHeader,
  int > & GetData () const

  Return the data.
- template<>
  const std::map< RecordHeader,
  double > & GetData () const

  Return the data.
- template<>
  const std::map< RecordHeader,
  string > & GetData () const

  Return the data.

4.7.1 Detailed Description

A serializable object that can store the results of data acquisition.

4.7.2 Member Function Documentation

4.7.2.1 template<typename T > double POPSensorData::Reduce (enum MeasurementType *x_mtype,* POPReduceF *x_fct*) const [inline]

Apply reduce on all the content note that there are two template types. The second is meant for the accumulator: for int use <int,long long> for double use <double,double>

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

- POPSensorData.h
- POPSensorData.cc

4.8 PublishMessage Struct Reference

Publication message.

`#include <popwin_messages.h>`

Public Attributes

- enum PublicationType publicationType

  Type of the publication.
- enum DataType dataType

  Type of the data.
- unsigned short id

  Id of emitter.
- size_t dataSize

  Size of the data for buffering.
- char data [BUFFERDATASIZE]

  Buffer containing the serialized data.

4.8.1 Detailed Description

Publication message.

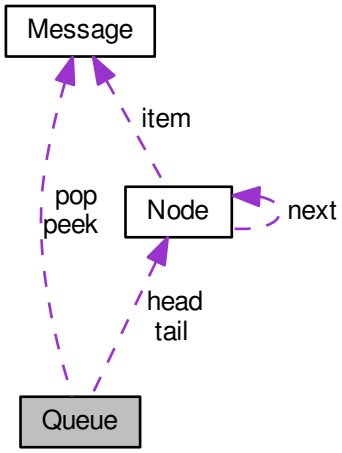
The documentation for this struct was generated from the following file:

- gatewayMote/popwin_messages.h

4.9 Queue Struct Reference

`#include <queue.h>`

Collaboration diagram for Queue:



Public Attributes

- Node * **head**
- Node * **tail**
- void(* **push**)(struct Queue *, struct Message)
- struct Message(* **pop**)(struct Queue *)
- struct Message(* **peek**)(struct Queue *)
- void(* **display**)(struct Queue *)
- int **size**

4.9.1 Detailed Description

The Queue struct, contains the pointers that point to first node and last node, the size of the Queue, and the function pointers.

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

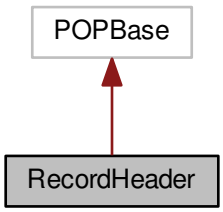
- gatewayMote/queue.h

4.10 RecordHeader Class Reference

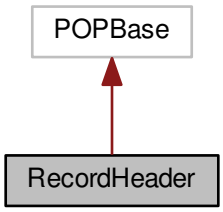
POPSensorData class for the POPWIN project. This object represents the data gathered from sensor.

`#include <POPSensorData.h>`

Inheritance diagram for RecordHeader:



Collaboration diagram for RecordHeader:



Public Member Functions

- RecordHeader ()

  Constructor.
- RecordHeader (unsigned int x_timeStamp, const NotifyMessage &x_msg)

  Constructor.
- void Serialize (POPBuffer &buf, bool pack)

  Serialize the object.
- bool operator< (RecordHeader const &n2) const

  We need to define this operator to use this structure as key for maps.

Public Attributes

- unsigned int timeStamp

  A time stamp that indicates when the record was created.

- enum MeasurementType measurementType

  The type of measurement.
- int id

  Id of the emitter.
- enum MeasurementUnit unit

  Unit of measurement.

4.10.1 Detailed Description

POPSensorData class for the POPWIN project. This object represents the data gathered from sensor.

Author

Laurent Winkler based on work by Valentin Bourqui

Date

Dec 2014Data structure to store a record that comes from a notification message

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

- POPSensorData.h
- POPSensorData.cc

4.11 SensorProxy Class Reference

Proxy class that handles communication with one sensor but resides on the same machine as the gateway. Each proxy is connected to one sensor on the network.

```
#include <SensorProxy.h>
```

Public Member Functions

- SensorProxy (int x_id, const std::string &x_url, const std::string &x_device)@
- ~SensorProxy ()

  Destructur.
- void Notify (int x_measurementType, int x_measurementUnit, const std::string &x_message)

  Send notification to the connected sensor.
- void Publish (int x_publicationType, int x_data)

  Send a publication to the connected sensor.
- void Publish (int x_publicationType, double x_data)

  Send a publication to the gateway.
- void Publish (int x_publicationType, const std::string &x_data)

  Send a publication to the gateway.
- double Reduce (int x_mtype, int x_dataType, int x_fct)

  Apply a reduce operation to the stored data {size, min, max, aver, sum, stdev}.
- void Subscribe (int x_measurementType, int x_dataType)

  Send a subscription to the connected sensor.
- void StartListening ()

  Send data to the remote sensors.
- void StopListening ()

Stop listening to messages coming from sensors.

- POPSensorData Gather ()

  Retrieve data gathered.

- void Clear ()

  Clear data gathered.

- int GetDataSize ()

  Return the size of the stored data.

4.11.1 Detailed Description

Proxy class that handles communication with one sensor but resides on the same machine as the gateway. Each proxy is connected to one sensor on the network.

Author

Laurent Winkler based on work by Valentin Bourqui

Date

Dec 2014

4.11.2 Constructor & Destructor Documentation

4.11.2.1 SensorProxy::SensorProxy (int *x_id*, const std::string & *x_url*, const std::string & *x_device*) [inline]

Constructor

Parameters

x_id	Id to set to this object (for low-level communication)
x_url	URL on which this parallel object is allocated
x_device	Device name e.g. /dev/ttyUSB0

4.11.3 Member Function Documentation

4.11.3.1 void SensorProxy::Clear ( )

Clear data gathered.

Clear the stored messages.

4.11.3.2 POPSensorData SensorProxy::Gather ( )

Retrieve data gathered.

Return a POPSensorData structure containing the messages received from sensors.

4.11.3.3 void SensorProxy::Notify (int *x_measurementType*, int *x_measurementUnit*, const std::string & *x_message*)

Send notification to the connected sensor.

Send a notification to the gateway.

4.11.3.4 void SensorProxy::Publish (int *x_publicationType,* int *x_data*)

Send a publication to the connected sensor.

Send a publication to the gateway.

4.11.3.5 void SensorProxy::StartListening ( )

Send data to the remote sensors.

Start listening to messages coming from sensors.

4.11.3.6 void SensorProxy::Subscribe (int *x_measurementType,* int *x_dataType*)

Send a subscription to the connected sensor.

Send a subscription to the gateway.

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

- SensorProxy.h
- SensorProxy.cc

4.12 SubscribeMessage Struct Reference

Subscription message.

`#include <popwin_messages.h>`

Public Attributes

- unsigned short id

  Id of emitter: Id is not mandatory. Only for convenience.
- enum MeasurementType measurementType

  Type of measurement.
- enum DataType dataType

  Type of the data.

4.12.1 Detailed Description

Subscription message.

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

- gatewayMote/popwin_messages.h

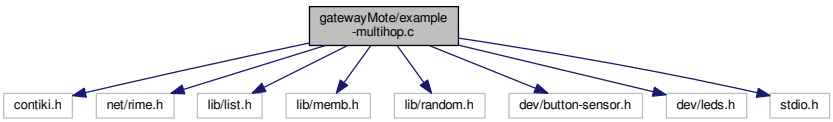
Chapter 5

File Documentation

5.1 gatewayMote/example-multihop.c File Reference

```
#include "contiki.h"
#include "net/rime.h"
#include "lib/list.h"
#include "lib/memb.h"
#include "lib/random.h"
#include "dev/button-sensor.h"
#include "dev/leds.h"
#include <stdio.h>
```

Include dependency graph for example-multihop.c:



Classes

- struct example_neighbor

Macros

- #define **CHANNEL** 135
- #define **NEIGHBOR_TIMEOUT** 60 * CLOCK_SECOND
- #define **MAX_NEIGHBORS** 16

Functions

- **LIST** (neighbor_table)
- **MEMB** (neighbor_mem, struct example_neighbor, MAX_NEIGHBORS)

5.1.1 Detailed Description

Testing the multihop forwarding layer (multihop) in Rime

Author

Adam Dunkels adam@sics.se

```
This example shows how to use the multihop Rime module, how
to use the announcement mechanism, how to manage a list
with the list module, and how to allocate memory with the
memb module.

The multihop module provides hooks for forwarding packets
in a multi-hop fashion, but does not implement any routing
protocol. A routing mechanism must be provided by the
application or protocol running on top of the multihop
module. In this case, this example program provides the
routing mechanism.

The routing mechanism implemented by this example program
is very simple: it forwards every incoming packet to a
random neighbor. The program maintains a list of neighbors,
which it populated through the use of the announcement
mechanism.

The neighbor list is populated by incoming announcements
from neighbors. The program maintains a list of neighbors,
where each entry is allocated from a MEMB() (memory block
pool). Each neighbor has a timeout so that they do not
occupy their list entry for too long.

When a packet arrives to the node, the function forward()
is called by the multihop layer. This function picks a
random neighbor to send the packet to. The packet is
forwarded by every node in the network until it reaches its
final destination (or is discarded in transit due to a
transmission error or a collision).
```

Index