Greenfeed - EMSE server program

1.0

Generated by Doxygen 1.8.6

Wed Sep 30 2015 16:30:32

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

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

Chapter 3

Data Structure Documentation

3.1 base64_decodestate Struct Reference

Data Fields

- base64_decodestep **step**
- char **plainchar**

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

- includes/b64/cdecode.h

3.2 base64_decodestate Struct Reference

Data Fields

- base64_decodestep **step**
- char **plainchar**

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

- includes/b64/decode.h

3.3 base64_encodestate Struct Reference

Data Fields

- base64_encodestep **step**
- char **result**
- int **stepcount**

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

- includes/b64/cencode.h

3.4 base64_encodestate Struct Reference

Data Fields

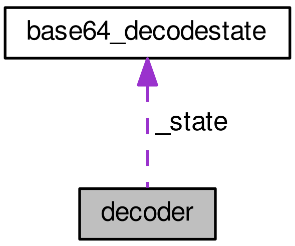
- base64_encodestep **step**
- char **result**
- int **stepcount**

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

- includes/b64/encode.h

3.5 decoder Struct Reference

Collaboration diagram for decoder:



Public Member Functions

- **decoder** (int buffersize_in=BUFFERSIZE)
- int **decode** (char value_in)
- int **decode** (const char ∗code_in, const int length_in, char ∗plaintext_out)
- void **decode** (std::istream &istream_in, std::ostream &ostream_in)

Data Fields

- base64_decodestate **_state**
- int **_buffersize**

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

- includes/b64/decode.h

3.6 downstream_packet Struct Reference

This structure represents a package that must be sent to the IoT station to make it send a LoRa message Some of these fields are optional.

```
#include <includes/downstream_packet.h>
```

Data Fields

- bool send_now
- long timestamp
- char time [35]
- double frequency
- unsigned int rf_channel
- unsigned int power
- char modulation [5]
- char datarate [15]
- unsigned int fsk_datarate
- char code_rate [10]
- unsigned int fsk_deviation
- bool i_polarization
- unsigned int rf_preamble
- unsigned int payload_size
- char ∗ data
- bool disable_CRC

3.6.1 Detailed Description

This structure represents a package that must be sent to the IoT station to make it send a LoRa message Some of these fields are optional.

3.6.2 Field Documentation

3.6.2.1 char code_rate[10]

The code rate used to emit the packet, example 4/5

3.6.2.2 char∗ data

The actual payload in base64

3.6.2.3 char datarate[15]

The datarate used in case the LORA modulation is chosen

3.6.2.4 bool disable_CRC

If this field is set to true, the CRC check will be disabled (should always be set to false!)

3.6.2.5 double frequency

The central frequency used when the packet is emitted

3.6.2.6 unsigned int fsk_datarate

The datarate used in case the FSK modulation is chosen

3.6.2.7 unsigned int fsk_deviation

The FSK frequency deviation

3.6.2.8 bool i_polarization

If this field is set to true, the LoRa modulation polarization will be inversed

3.6.2.9 char modulation[5]

The LoRa modulation used: LORA or FSK. Other values will be discarded

3.6.2.10 unsigned int payload_size

The payload size in bytes

3.6.2.11 unsigned int power

The power used by the antenna to emit the packet

3.6.2.12 unsigned int rf_channel

The channel the IoT station will use to emit the packet

3.6.2.13 unsigned int rf_preamble

The RF preamble size

3.6.2.14 bool send_now

If this field is equal to true, the packet will be sent by the IoT station immediately

3.6.2.15 char time[35]

Same use that the timestamp field. The representation of the date is different

3.6.2.16 long timestamp

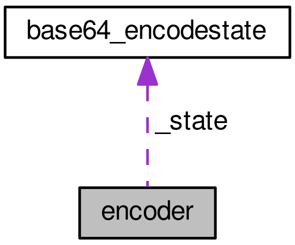
If the send_now field is set to false, the packet will be sent when the current date of the IoT station reaches this value

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

- includes/downstream_packet.h

3.7 encoder Struct Reference

Collaboration diagram for encoder:



Public Member Functions

- **encoder** (int buffersize_in=BUFFERSIZE)
- int **encode** (char value_in)
- int **encode** (const char ∗code_in, const int length_in, char ∗plaintext_out)
- int **encode_end** (char ∗plaintext_out)
- void **encode** (std::istream &istream_in, std::ostream &ostream_in)

Data Fields

- base64_encodestate **_state**
- int **_buffersize**

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

- includes/b64/encode.h

3.8 iot_pull_data Union Reference

This union represents a message the IoT station send when it receives a LoRa message.

```
#include "includes/iot_pull_data.h"
```

Public Member Functions

- struct __attribute__ ((__packed__))

Data Fields

- char msg [INCOMMING_MSG_SIZE]

3.8.1 Detailed Description

This union represents a message the IoT station send when it receives a LoRa message.

Here is a part of the official documentation. The structure iot_pull_data is a C representation of the documentation

5.2. PULL_DATA packet

That packet type is used by the gateway to poll data from the server.

This data exchange is initialized by the gateway because it might be impossible for the server to send packets to the gateway if the gateway is behind a NAT.

When the gateway initialize the exchange, the network route towards the server will open and will allow for packets to flow both directions. The gateway must periodically send PULL_DATA packets to be sure the network route stays open for the server to be used at any time.

Bytes	Function
0	protocol version = 1
1-2	random token
3	PULL_DATA identifier 0x02
4-11	Gateway unique identifier (MAC address)

This is an union and not a structure for some "easier to use" reasons. For academic reasons, let's explain it. An union is different from a structure in how it manages memory. In a structure, the memory scheme is as follow :

|_field A_|____________ field B__________| Padding |____Field C____| ...

Each field starts when the field before stops (some padding may be added, by that's an other topic). An union typical scheme is as follow:

|_field A_|

|___________field B_________|

|field C|

All the fields start at the same adress.

So, why an union? As the documentation says, the message received by the server is a 12 bytes, and each byte has its own usage. We have to "cut" the message in part to get each packet of bytes. We could have done it by using the substring function, but that's boring (need to manage memory allocation, calculate indexes ...). By using an union AND an internal structure, we can achieve the same thing by way more easily. We use an union to make the message and the internal structure to start at the same address. Then we use the structure memory scheme to create fields which have the perfect size to "cut" the message where it needs to be. Let's see the result :

As we can see, the protocol_version field of the structure has the same content of the first byte of the message, and so does token ...

3.8.2 Member Function Documentation

3.8.2.1 struct __attribute__ ((__packed__)) `[inline]`

This is the structure which will cut the message. To have some explanations about the packed attribute, go see Mr LALEVEE

3.8.3 Field Documentation

3.8.3.1 char msg[INCOMMING_MSG_SIZE]

This is the field in which the message sent by the IoT station will be

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

- includes/iot_pull_data.h

3.9 iot_push_data Union Reference

To have more information about this structure, look at the iot_pull_data.

```
#include "includes/iot_push_data.h"
```

Public Member Functions

- struct **__attribute__** ((__packed__))

Data Fields

- char **msg** [INCOMMING_MSG_SIZE]

3.9.1 Detailed Description

To have more information about this structure, look at the iot_pull_data.

Here is a part of the documentation

3.2. PUSH_DATA packet

That packet type is used by the gateway mainly to forward the RF packets received, and associated metadata, to the server.

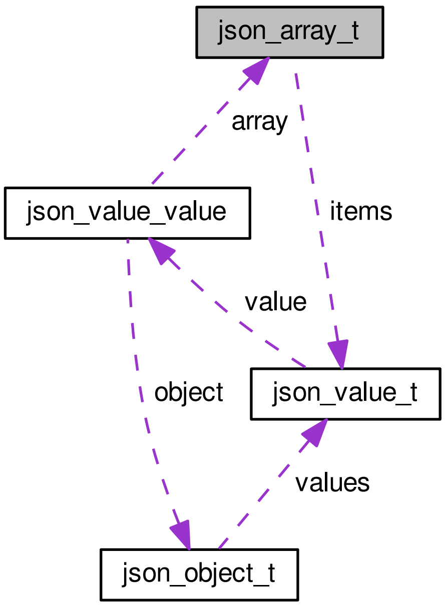
Bytes	Function
0	protocol version = 1
1-2	random token
3	PUSH_DATA identifier 0x00
4-11	Gateway unique identifier (MAC address)
12-end	JSON object, starting with {, ending with }, see section 4

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

- includes/iot_push_data.h

3.10 json_array_t Struct Reference

Collaboration diagram for json_array_t:



Data Fields

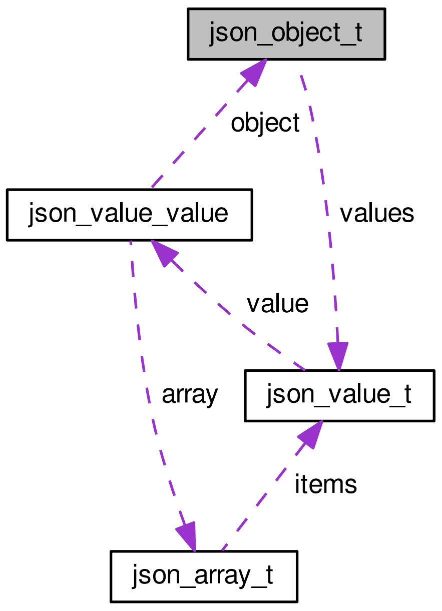
- JSON_Value ∗∗ **items**
- size_t **count**
- size_t **capacity**

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

- parson/parson.c

3.11 json_object_t Struct Reference

Collaboration diagram for json_object_t:



Data Fields

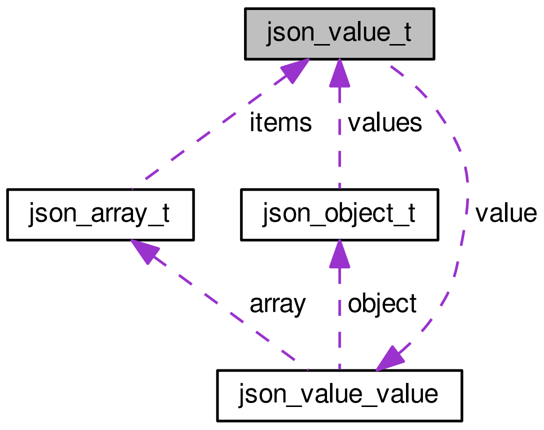
- char ∗∗ **names**
- JSON_Value ∗∗ **values**
- size_t **count**
- size_t **capacity**

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

- parson/parson.c

3.12 json_value_t Struct Reference

Collaboration diagram for json_value_t:



Data Fields

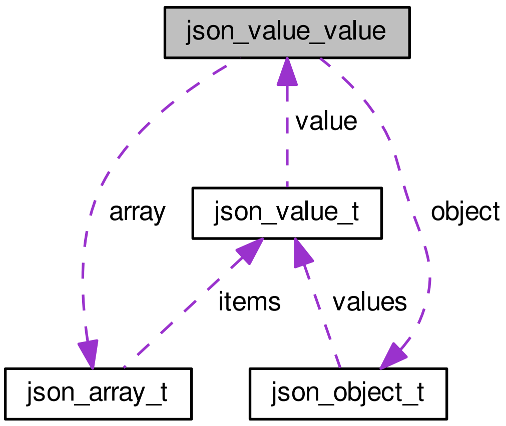
- JSON_Value_Type **type**
- JSON_Value_Value **value**

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

- parson/parson.c

3.13 json_value_value Union Reference

Collaboration diagram for json_value_value:



Data Fields

- char ∗ **string**
- double **number**
- JSON_Object ∗ **object**
- JSON_Array ∗ **array**
- int **boolean**
- int **null**

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

- parson/parson.c

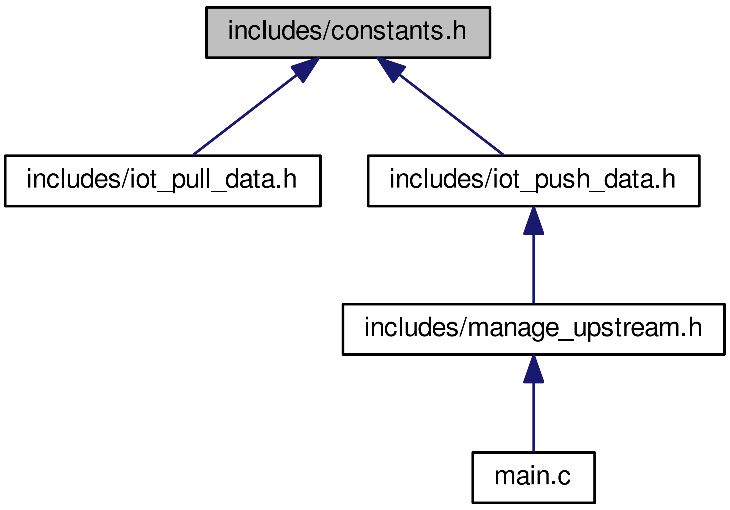
Chapter 4

File Documentation

4.1 includes/constants.h File Reference

This file defines some constants used in the program.

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



Macros

- #define MAX_UPSTREAM_ERROR 5
- #define MAX_DOWNSTREAM_ERROR 5
- #define INCOMMING_MSG_SIZE 2048
- #define INVALID_SOCKET -1
- #define SOCKET_ERROR -2
- #define JSON_BAD_FORMED -3

4.1.1 Detailed Description

This file defines some constants used in the program.

Author

Giuliano Franchetto

Date

30/09/2015

Version

1

4.1.2 Macro Definition Documentation

4.1.2.1 #define INCOMMING_MSG_SIZE 2048

This is the maximum size of a packet sent by the IoT station to the program. This is not robust, but for the first version, this will do it

4.1.2.2 #define INVALID_SOCKET -1

Error code returned when creating a socket fails

4.1.2.3 #define JSON_BAD_FORMED -3

Error code returned when a JSON sent by the IoT station is malformed

4.1.2.4 #define MAX_DOWNSTREAM_ERROR 5

This field defines the number of allowed error before closing the downstream socket with the IoT station

4.1.2.5 #define MAX_UPSTREAM_ERROR 5

This field defines the number of allowed error before closing the upstream socket with the IoT station

4.1.2.6 #define SOCKET_ERROR -2

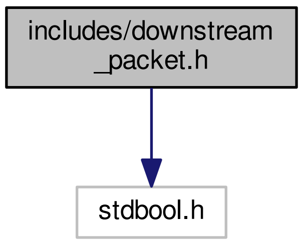
Error code returned when a read/write operation fails

4.2 includes/downstream_packet.h File Reference

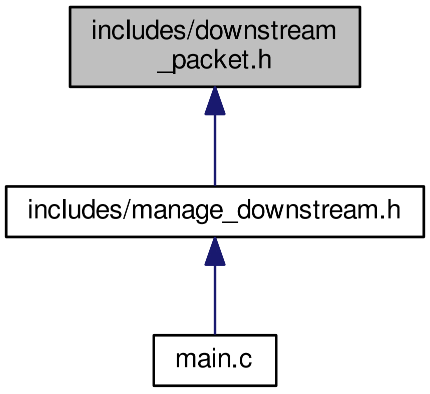
This file defines the structure "downstream_packet".

```
#include <stdbool.h>
```

Include dependency graph for downstream_packet.h:



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



Data Structures

- struct downstream_packet

    This structure represents a package that must be sent to the IoT station to make it send a LoRa message Some of these fields are optional.

4.2.1 Detailed Description

This file defines the structure "downstream_packet".

Author

Giuliano Franchetto

Date

30/09/2015

Version

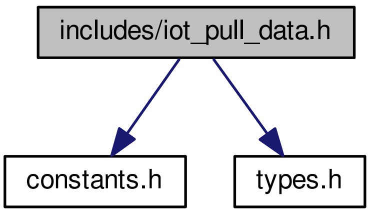
1

The reader can find the official documentation on this url: https://github.com/Lora-net/packet_-forwarder/blob/master/PROTOCOL.TXT

4.3 includes/iot_pull_data.h File Reference

This file defines a structure used to represent the pull_data message sent by the IoT station.

```
#include <constants.h>
#include <types.h>
```
Include dependency graph for iot_pull_data.h:



Data Structures

- union iot_pull_data

  This union represents a message the IoT station send when it receives a LoRa message.

4.3.1 Detailed Description

This file defines a structure used to represent the pull_data message sent by the IoT station.

Author

Giuliano Franchetto

Date

30/09/2015

Version

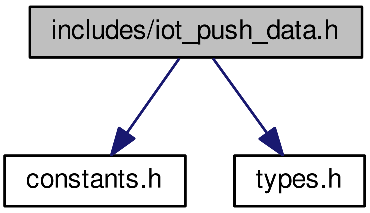
1

4.4 includes/iot_push_data.h File Reference

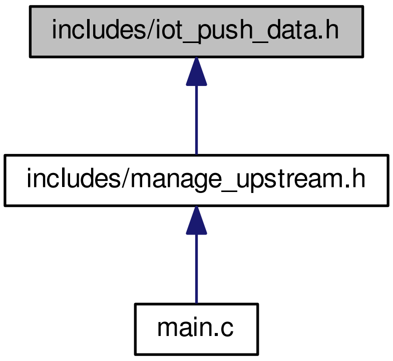
This file defines a structure used to represent the push_data message sent by the IoT station.

```
#include "constants.h"
#include "types.h"
```

Include dependency graph for iot_push_data.h:



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



Data Structures

- union iot_push_data

    To have more information about this structure, look at the iot_pull_data.

4.4.1 Detailed Description

This file defines a structure used to represent the push_data message sent by the IoT station.

Author

Giuliano Franchetto

Date

30/09/2015

Version

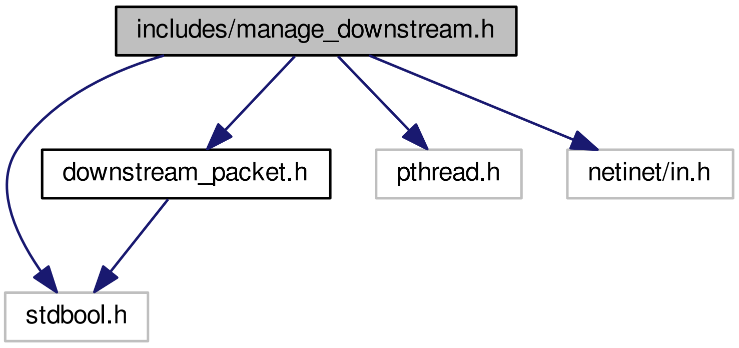
1

4.5 includes/manage_downstream.h File Reference

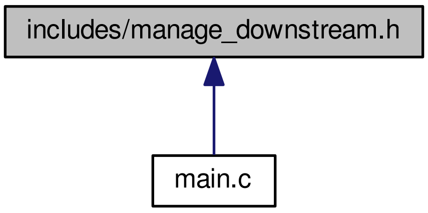
This file defines some function prototypes and global variables.

```
#include <stdbool.h>
#include <downstream_packet.h>
#include <pthread.h>
#include <netinet/in.h>
```

Include dependency graph for manage_downstream.h:



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



Functions

- int send_downstream_message (SOCKET socket, struct sockaddr_in from, socklen_t size, downstream_-packet packet, bool use_minimum_data)

This function send a message to the IoT station. This message will next be sent in LoRa.

- void ∗ manage_downstream (void ∗r)

    The thread routine called by main.

Variables

- pthread_t thread_manage_downstream
- bool stop_thread_downstream
- int downstream_error

4.5.1 Detailed Description

This file defines some function prototypes and global variables.

Author

Giuliano Franchetto

Date

30/09/2015

Version

1

4.5.2 Function Documentation

4.5.2.1 void ∗ manage_downstream (void ∗ *r*)

The thread routine called by main.

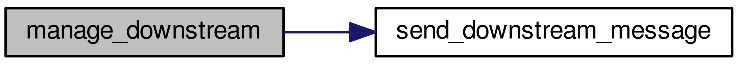
Parameters

r	unused param

Returns

NULL

Here is the call graph for this function:



4.5.2.2 int send_downstream_message (SOCKET *socket,* struct sockaddr_in *from,* socklen_t *size,* downstream_packet *packet,* bool *use_minimum_data*)

This function send a message to the IoT station. This message will next be sent in LoRa.

Parameters

socket	the udp socket with the IoT station
from	a sockaddr_in structure which defines the communication with the IoT station
size	
packet	the packet to send to the IoT station, see downstream_packet
use_minimum_-data	packet will be filled with default data in some fields. Needs to be set to true for the moment

Returns

0 on success, or the errno code

4.5.3 Variable Documentation

4.5.3.1 int downstream_error

The number of error this thread made

4.5.3.2 bool stop_thread_downstream

A boolean which must set to true to stop the thread

4.5.3.3 pthread_t thread_manage_downstream

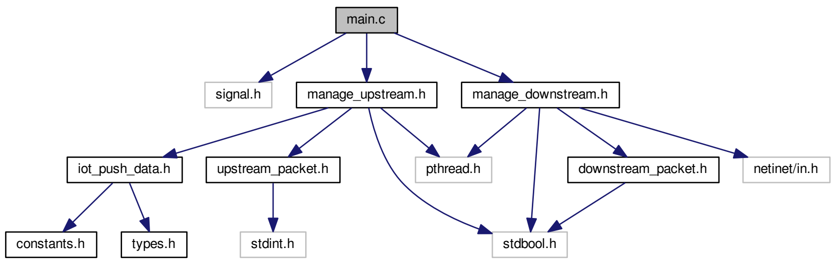
The pthread_t used for the manage_downstream thread

4.6 main.c File Reference

This file is the entry of the main program to communicate with the Kerlink IoT station.

```
#include <signal.h>
#include <manage_upstream.h>
#include <manage_downstream.h>
```

Include dependency graph for main.c:



Functions

- void **signal_int** (int sig)
- int **main** ()

Variables

- pthread_t **thread_manage_upstream**
- pthread_t thread_manage_downstream
- bool **stop_thread_upstream** = false
- bool stop_thread_downstream = false
- int **upstream_error** = 0
- int downstream_error = 0

4.6.1 Detailed Description

This file is the entry of the main program to communicate with the Kerlink IoT station.

Author

Giuliano Franchetto

Version

1.0

Date

09/30/2015

4.6.2 Variable Documentation

4.6.2.1 int downstream_error = 0

The number of error this thread made

4.6.2.2 bool stop_thread_downstream = false

A boolean which must set to true to stop the thread

4.6.2.3 pthread_t thread_manage_downstream

The pthread_t used for the manage_downstream thread