Libelium Smart Boards

1.3.1

Generated by Doxygen 1.9.1

Chapter 1

Main Page

1.0.0.1 Welcome!

Here you can find our documentation for the four different codes we have developed in order to monitor our IoT ecosystem.

1.0.0.2 Project Description

This project aims to monitor the condition of the iti's Smart Home wirelessly through the LoRaWAN protocol. For that reason, there has been used four different Smart Boards from Libelium to get data from various sensors and track the condition that exists in the Smart House every some time stamps. Such sensors are for hazardous gases (Gas board), for water quality (Water board), for wheather conditions (Agriculture board) and for various events like water presence (Events board). For sending the data (lora_send()) and for joining the network (lora_join()), there have been implemented two libraries and they are imported to each board's code.

For more details for each board open the links below:

For the Agriculture board: Info

For the Events board: Info

For the Gas board: Info

For the Water board: Info

Author

Ioannis Chatzipaschalis

Chapter 2

Smart Agriculture board

2.1 Info

The Smart Agriculture board is responsible for monitoring the outdoor wheather conditions such as the wetness, the height of rain, the direction of the vane, the temprature and the pressure in order to have the best conditions for growing plants. There are functions that get and print the sensors' measurements, another that prepares the data for sending and two last that are responsible for the connection eshtablishment (lora_join) and for the data transmission (lora_send), which you can find them in the separate files imported as libraries. The board's IDs such as Device EUI etc have been declared at the top of the code.

Chapter 3

Smart Events board

3.1 Info

The Smart Events board is responsible for monitoring the water presence to avoid a flood in critical rooms like a power room. There are functions that get and print the sensors' measurements, another that prepares the data for sending and two last that are responsible for the connection eshtablishment (lora_join) and for the data transmission (lora_send), which you can find them in the separate files imported as libraries. There have been voltage definitions at the top of the code for the water presence conditions based on the sensor manufacturer. In that part of the code you can also find the board's IDs such as Device EUI etc that have been declared.

Chapter 4

Smart Gas board

4.1 Info

The Smart Gas board is responsible for monitoring the concentration (in ppm) of hazardous gases in the atmosphere such as CO and NO2. There are functions that get and print the sensors' measurements, another that prepares the data for sending and two last that are responsible for the connection eshtablishment (lora_join) and for the data transmission (lora_send), which you can find them in the separate files imported as libraries. You can also see the calibration values as they are recommended from the manufacturer. The board's IDs such as Device EUI etc have been declared at the top of the code.

Chapter 5

Smart Water board

5.1 Info

The Smart Water board is responsible for monitoring the water's temprature and pH. There are functions that get and print the sensors' measurements, another that prepares the data for sending and two last that are responsible for the connection eshtablishment (lora_join) and for the data transmission (lora_send), which you can find them in the separate files imported as libraries. You can also see the calibration values as they are recommended from the manufacturer. The board's IDs such as Device EUI etc have been declared at the top of the code.

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

Chapter 7

File Documentation

7.1 Codes/lora_join.cpp File Reference

```
#include <WaspLoRaWAN.h>
#include "Waspmote.h"
#include "lora_join.h"
```

Functions

- void lora_join (char device_eui[ ], char app_eui[ ], char app_key[ ], uint8_t socket, uint8_t adr_flag)

  LoRaWAN Connection Function.

7.1.1 Function Documentation

7.1.1.1 lora_join()

```
void lora_join (
            char device_eui[ ],
            char app_eui[ ],
            char app_key[ ],
            uint8_t socket,
            uint8_t adr_flag )
```

LoRaWAN Connection Function.

Returns

Nothing

Parameters

device_eui	The EUI of the board.
app_eui	The EUI of the application.
app_key	The key needed for the encryption.
socket	Specifies the socket of the board we are using.
adr_flag	Indicates if we need Adaptive Data Rate or not.

7.2 Codes/lora_join.h File Reference

Functions

- void lora_join (char device_eui[ ], char app_eui[ ], char app_key[ ], uint8_t socket, uint8_t adr_flag)
  LoRaWAN Connection Function.

7.2.1 Function Documentation

7.2.1.1 lora_join()

```
void lora_join (
            char device_eui[ ],
            char app_eui[ ],
            char app_key[ ],
            uint8_t socket,
            uint8_t adr_flag )
```

LoRaWAN Connection Function.

Returns

Nothing

Parameters

device_eui	The EUI of the board.
app_eui	The EUI of the application.
app_key	The key needed for the encryption.
socket	Specifies the socket of the board we are using.
adr_flag	Indicates if we need Adaptive Data Rate or not.

7.3 Codes/lora_send.cpp File Reference

```
#include <WaspLoRaWAN.h>
#include <WaspUSB.h>
#include <WaspUSB.cpp>
#include "Waspmote.h"
#include "lora_send.h"
```

Functions

- void lora_send (char device_eui[ ], char app_eui[ ], char app_key[ ], uint8_t size_of_buffer, uint8_t port, uint8_t socket, uint8_t bytes[ ])

  Data Transmission Function.

7.3.1 Function Documentation

7.3.1.1 lora_send()

```
void lora_send (
            char device_eui[ ],
            char app_eui[ ],
            char app_key[ ],
            uint8_t size_of_buffer,
            uint8_t port,
            uint8_t socket,
            uint8_t bytes[ ] )
```

Data Transmission Function.

Returns

Nothing

Parameters

device_eui	The EUI of the board.
app_eui	The EUI of the application.
app_key	The key needed for the encryption.
size_of_buffer	The size of our data vector.
port	Specifies the port we are using.
socket	Specifies the socket of the board we are using.
bytes	The data vector to be sent.

7.4 Codes/lora_send.h File Reference

Functions

- void lora_send (char device_eui[ ], char app_eui[ ], char app_key[ ], uint8_t size_of_buffer, uint8_t port, uint8↩_t socket, uint8_t bytes[ ])

  Data Transmission Function.

7.4.1 Function Documentation

7.4.1.1 lora_send()

```
void lora_send (
            char device_eui[ ],
            char app_eui[ ],
            char app_key[ ],
            uint8_t size_of_buffer,
            uint8_t port,
            uint8_t socket,
            uint8_t bytes[ ] )
```

Data Transmission Function.

Returns

Nothing

Parameters

device_eui	The EUI of the board.
app_eui	The EUI of the application.
app_key	The key needed for the encryption.
size_of_buffer	The size of our data vector.
port	Specifies the port we are using.
socket	Specifies the socket of the board we are using.
bytes	The data vector to be sent.

7.5 Codes/Smart_Agriculture.cpp File Reference

```
#include <WaspLoRaWAN.h>
#include <WaspSensorAgr_v30.h>
#include <lora_send.h>
#include <lora_join.h>
```

Macros

- #define SENDRATEMINUTES 30

    Time in minutes.

Functions

- watermarkClass wmSensor1 (SOCKET_2)
- void setup ()
- void loop ()
- void create_bytes ()

    Function for creating the ready to send bytes vector.
- void get_measurements ()

    Function for getting the measurements from the Sensors.
- void measureSensors ()

    Function for Reading the wind sensor values.

Variables

- uint8_t socket = SOCKET0

    Socket Selection for the Sensor Probe.
- char device_eui [ ] = "72c465dc9a6d5390"

    Device parameters for Back-End registration.
- char app_eui [ ] = "72c465dc9a6d5390"
- char app_key [ ] = "1503835a0767d925cd3c8f9f2bcbb6b5"
- uint8_t port = 10

    Define port to use in Back-End: from 1 to 223.
- uint8_t bytes [45] = {0}

    Define data payload to send (maximum is up to data rate)
- char vane_str [10] = {0}
- uint8_t j = 0

    Variables.
- uint8_t i = 0
- int battery_level = 0
- float wetness = 0
- float watermark1
- float temperature = 0
- float anemometer
- float pluviometer1

    mm in current hour
- float pluviometer2

    mm in previous hour
- float pluviometer3

    mm in last 24 hours
- uint8_t cbar = 2
- int vane
- float BME_temperature
- float humidity
- float pressure
- int pending_pulses
- uint8_t counter = 0

- uint8_t vane_flag = 0
- uint8_t size_of_buffer = 0
- uint8_t adr_flag = 1

  Adaptive Data Rate indicator for lora_join()
- leafWetnessClass lwSensor
- weatherStationClass vaneSensor
- weatherStationClass weather
- pt1000Class pt1000Sensor

7.5.1 Macro Definition Documentation

7.5.1.1 SENDRATEMINUTES

```
#define SENDRATEMINUTES 30
```

Time in minutes.

7.5.2 Function Documentation

7.5.2.1 create_bytes()

```
void create_bytes ( )
```

Function for creating the ready to send bytes vector.

Returns

Nothing

Decoding values to base64 for send preparation

7.5.2.2 get_measurements()

```
void get_measurements ( )
```

Function for getting the measurements from the Sensors.

Returns

Nothing

7.5.2.3 loop()

```
void loop ( )
```

7.5.2.4 measureSensors()

```
void measureSensors ( )
```

Function for Reading the wind sensor values.

Returns

Nothing

7.5.2.5 setup()

```
void setup ( )
```

7.5.2.6 wmSensor1()

```
watermarkClass wmSensor1 (
             SOCKET_2  )
```

7.5.3 Variable Documentation

7.5.3.1 adr_flag

```
uint8_t adr_flag = 1
```

Adaptive Data Rate indicator for lora_join()

7.5.3.2 anemometer

```
float anemometer
```

7.5.3.3 app_eui

```
char app_eui[] = "72c465dc9a6d5390"
```

7.5.3.4 app_key

```
char app_key[] = "1503835a0767d925cd3c8f9f2bcbb6b5"
```

7.5.3.5 battery_level

```
int battery_level = 0
```

7.5.3.6 BME_temperature

```
float BME_temperature
```

7.5.3.7 bytes

```
uint8_t bytes[45] = {0}
```

Define data payload to send (maximum is up to data rate)

7.5.3.8 cbar

```
uint8_t cbar = 2
```

7.5.3.9 counter

```
uint8_t counter = 0
```

7.5.3.10 device_eui

```
char device_eui[] = "72c465dc9a6d5390"
```

Device parameters for Back-End registration.

7.5.3.11 humidity

```
float humidity
```

7.5.3.12 i

```
uint8_t i = 0
```

7.5.3.13 j

```
uint8_t j = 0
```

Variables.

7.5.3.14 lwSensor

```
leafWetnessClass lwSensor
```

7.5.3.15 pending_pulses

```
int pending_pulses
```

7.5.3.16 pluviometer1

```
float pluviometer1
```

mm in current hour

7.5.3.17 pluviometer2

```
float pluviometer2
```

mm in previous hour

7.5.3.18 pluviometer3

```
float pluviometer3
```

mm in last 24 hours

7.5.3.19 port

```
uint8_t port = 10
```

Define port to use in Back-End: from 1 to 223.

7.5.3.20 pressure

```
float pressure
```

7.5.3.21 pt1000Sensor

```
pt1000Class pt1000Sensor
```

7.5.3.22 size_of_buffer

```
uint8_t size_of_buffer = 0
```

7.5.3.23 socket

```
uint8_t socket = SOCKET0
```

Socket Selection for the Sensor Probe.

7.5.3.24 temperature

```
float temperature = 0
```

7.5.3.25 vane

```
int vane
```

7.5.3.26 vane_flag

```
uint8_t vane_flag = 0
```

7.5.3.27 vane_str

```
char vane_str[10] = {0}
```

7.5.3.28 vaneSensor

```
weatherStationClass vaneSensor
```

7.5.3.29 watermark1

```
float watermark1
```

7.5.3.30 weather

```
weatherStationClass weather
```

7.5.3.31 wetness

```
float wetness = 0
```

7.6 Codes/Smart_Events.cpp File Reference

```
#include <WaspSensorEvent_v30.h>
#include <WaspLoRaWAN.h>
#include <lora_send.h>
#include <lora_join.h>
```

Macros

- #define DRY 0.0

  Voltage Values.
- #define WET 0.2
- #define VERY_WET 1.0

Functions

- liquidPresenceClass liquidPresence (SOCKET_6)
- pirSensorClass pir (SOCKET_1)
- liquidLevelClass liquidLevel (SOCKET_4)
- void setup ()
- void loop ()

Variables

- uint8_t socket = SOCKET0

  Socket Selection for the Sensor Probe.
- char device_eui [ ] = "72c465dc2f1bcfa7"

  Device parameters for Back-End registration.
- char app_eui [ ] = "72c465dc2f1bcfa7"
- char app_key [ ] = "b5351d67909803c983b66484d25f9362"
- uint8_t port = 10

  Define port to use in Back-End: from 1 to 223.
- uint8_t bytes [12] = {0}

  Define data payload to send (maximum is up to data rate)
- char data [ ] = "0102030405060708090A0B0C0D0E0F"

  Setup the data vector.
- int i

  Variables.
- int battery_level = 0
- uint8_t value
- float voltage
- float resistance
- uint8_t pir_value = 0
- uint8_t liquid_level = 0
- uint8_t liquid_flag = 0
- uint8_t size_of_buffer = sizeof(bytes) / sizeof(bytes[0])
- uint8_t adr_flag = 1

  Adaptive Data Rate indicator for lora_join()

7.6.1 Macro Definition Documentation

7.6.1.1 DRY

```
#define DRY 0.0
```

Voltage Values.

7.6.1.2 VERY_WET

```
#define VERY_WET 1.0
```

7.6.1.3 WET

```
#define WET 0.2
```

7.6.2 Function Documentation

7.6.2.1 liquidLevel()

```
liquidLevelClass liquidLevel (
             SOCKET_4  )
```

7.6.2.2 liquidPresence()

```
liquidPresenceClass liquidPresence (
             SOCKET_6  )
```

7.6.2.3 loop()

```
void loop ( )
```

7.6.2.4 pir()

```
pirSensorClass pir (
             SOCKET_1  )
```

7.6.2.5 setup()

```
void setup ( )
```

7.6.3 Variable Documentation

7.6.3.1 adr_flag

```
uint8_t adr_flag = 1
```

Adaptive Data Rate indicator for lora_join()

7.6.3.2 app_eui

```
char app_eui[ ] = "72c465dc2f1bcfa7"
```

7.6.3.3 app_key

```
char app_key[ ] = "b5351d67909803c983b66484d25f9362"
```

7.6.3.4 battery_level

```
int battery_level = 0
```

7.6.3.5 bytes

```
uint8_t bytes[12] = {0}
```

Define data payload to send (maximum is up to data rate)

7.6.3.6 data

```
char data[] = "0102030405060708090A0B0C0D0E0F"
```

Setup the data vector.

7.6.3.7 device_eui

```
char device_eui[] = "72c465dc2f1bcfa7"
```

Device parameters for Back-End registration.

7.6.3.8 i

```
int i
```

Variables.

7.6.3.9 liquid_flag

```
uint8_t liquid_flag = 0
```

7.6.3.10 liquid_level

```
uint8_t liquid_level = 0
```

7.6.3.11 pir_value

```
uint8_t pir_value = 0
```

7.6.3.12 port

```
uint8_t port = 10
```

Define port to use in Back-End: from 1 to 223.

7.6.3.13 resistance

```
float resistance
```

7.6.3.14 size_of_buffer

```
uint8_t size_of_buffer = sizeof(bytes) / sizeof(bytes[0])
```

7.6.3.15 socket

```
uint8_t socket = SOCKET0
```

Socket Selection for the Sensor Probe.

7.6.3.16 value

```
uint8_t value
```

7.6.3.17 voltage

```
float voltage
```

7.7 Codes/Smart_Gas.cpp File Reference

```
#include <WaspLoRaWAN.h>
#include <WaspSensorGas_v30.h>
#include <lora_send.h>
#include <lora_join.h>
```

Macros

- #define MINUTESTOSEND 28

  Time in minutes.
- #define CALIBRATION_POINTS 3

  3 Calibration points
- #define POINT1_PPM_CO 100.0

  Ro value at this concentration.
- #define POINT2_PPM_CO 300.0
- #define POINT3_PPM_CO 1000.0
- #define POINT1_RES_CO 20.3

  Ro Resistance at 100 ppm.
- #define POINT2_RES_CO 1.00
- #define POINT3_RES_CO 0.25
- #define POINT1_PPM_NO2 10.0

  Normal concentration in the air.
- #define POINT2_PPM_NO2 50.0
- #define POINT3_PPM_NO2 100.0
- #define POINT1_RES_NO2 2.00

  Rs at normal concentration in the air.
- #define POINT2_RES_NO2 0.001
- #define POINT3_RES_NO2 0.0002

Functions

- APSensorClass APPSensor (SOCKET_6)
- APSensorClass APPSensor2 (SOCKET_7)
- void setup ()
- void loop ()

Variables

- uint8_t socket = SOCKET0

  Device parameters for Back-End registration.
- char device_eui [ ] = "72c465dcdb8b2afd"
- char app_eui [ ] = "72c465dcdb8b2afd"
- char app_key [ ] = "c1e90d187fb93834e60ba43c26fb9282"
- uint8_t port = 10

  Define port to use in Back-End: from 1 to 223.
- char bytes [33] = {0}

  Define data payload to send (maximum is up to data rate)
- int battery_level = 0

  General Variables.
- float luxes
- uint8_t error
- uint8_t i = 0
- int luxes_send = 0
- uint8_t size_of_buffer = sizeof(bytes) / sizeof(bytes[0])
- uint8_t adr_flag = 0

  Adaptive Data Rate indicator for lora_join()
- float co_voltage

Sensor Variables.

- float co_resistance
- float co_particles
- int co_send = 0
- float no2_voltage
- float no2_resistance
- float no2_particles
- int no2_send = 0
- float polutants_voltage
- float polutants_resistance
- float polutants_particles
- int polutants_send = 0
- float polutants2_voltage
- float polutants2_resistance
- float polutants2_particles
- int polutants2_send = 0
- COSensorClass COSensor
- NO2SensorClass NO2Sensor
- float concentrations_co [ ] = {POINT1_PPM_CO, POINT2_PPM_CO, POINT3_PPM_CO}
- float resistances_co [ ] = {POINT1_RES_CO, POINT2_RES_CO, POINT3_RES_CO}
- float concentrations_no2 [ ] = {POINT1_PPM_NO2, POINT2_PPM_NO2, POINT3_PPM_NO2}
- float voltages_no2 [ ] = {POINT1_RES_NO2, POINT2_RES_NO2, POINT3_RES_NO2}

7.7.1 Macro Definition Documentation

7.7.1.1 CALIBRATION_POINTS

```
#define CALIBRATION_POINTS 3
```

3 Calibration points

7.7.1.2 MINUTESTOSEND

```
#define MINUTESTOSEND 28
```

Time in minutes.

7.7.1.3 POINT1_PPM_CO

```
#define POINT1_PPM_CO 100.0
```

Ro value at this concentration.

7.7.1.4 POINT1_PPM_NO2

```
#define POINT1_PPM_NO2 10.0
```

Normal concentration in the air.

7.7.1.5 POINT1_RES_CO

```
#define POINT1_RES_CO 20.3
```

Ro Resistance at 100 ppm.

Necessary value.

7.7.1.6 POINT1_RES_NO2

```
#define POINT1_RES_NO2 2.00
```

Rs at normal concentration in the air.

7.7.1.7 POINT2_PPM_CO

```
#define POINT2_PPM_CO 300.0
```

7.7.1.8 POINT2_PPM_NO2

```
#define POINT2_PPM_NO2 50.0
```

7.7.1.9 POINT2_RES_CO

```
#define POINT2_RES_CO 1.00
```

7.7.1.10 POINT2_RES_NO2

```
#define POINT2_RES_NO2 0.001
```

7.7.1.11 POINT3_PPM_CO

```
#define POINT3_PPM_CO 1000.0
```

7.7.1.12 POINT3_PPM_NO2

```
#define POINT3_PPM_NO2 100.0
```

7.7.1.13 POINT3_RES_CO

```
#define POINT3_RES_CO 0.25
```

7.7.1.14 POINT3_RES_NO2

```
#define POINT3_RES_NO2 0.0002
```

7.7.2 Function Documentation

7.7.2.1 APPSensor()

```
APSensorClass APPSensor (
             SOCKET_6  )
```

7.7.2.2 APPSensor2()

```
APSensorClass APPSensor2 (
             SOCKET_7  )
```

7.7.2.3 loop()

```
void loop ( )
```

7.7.2.4 setup()

```
void setup ( )
```

< Ro value at this concentration

< Ro Resistance at 100 ppm. Necessary value.

< Ro value at this concentration

< Ro Resistance at 100 ppm. Necessary value.

7.7.3 Variable Documentation

7.7.3.1 adr_flag

```
uint8_t adr_flag = 0
```

Adaptive Data Rate indicator for lora_join()

7.7.3.2 app_eui

```
char app_eui[ ] = "72c465dcdb8b2afd"
```

7.7.3.3 app_key

```
char app_key[ ] = "c1e90d187fb93834e60ba43c26fb9282"
```

7.7.3.4 battery_level

```
int battery_level = 0
```

General Variables.

7.7.3.5 bytes

```
char bytes[33] = {0}
```

Define data payload to send (maximum is up to data rate)

7.7.3.6 co_particles

```
float co_particles
```

7.7.3.7 co_resistance

```
float co_resistance
```

7.7.3.8 co_send

```
int co_send = 0
```

7.7.3.9 co_voltage

```
float co_voltage
```

Sensor Variables.

7.7.3.10 concentrations_co

```
float concentrations_co[ ] = {POINT1_PPM_CO, POINT2_PPM_CO, POINT3_PPM_CO}
```

7.7.3.11 concentrations_no2

```
float concentrations_no2[ ] = {POINT1_PPM_NO2, POINT2_PPM_NO2, POINT3_PPM_NO2}
```

7.7.3.12 COSensor

```
COSensorClass COSensor
```

7.7.3.13 device_eui

```
char device_eui[] = "72c465dcdb8b2afd"
```

7.7.3.14 error

```
uint8_t error
```

7.7.3.15 i

```
uint8_t i = 0
```

7.7.3.16 luxes

```
float luxes
```

7.7.3.17 luxes_send

```
int luxes_send = 0
```

7.7.3.18 no2_particles

```
float no2_particles
```

7.7.3.19 no2_resistance

```
float no2_resistance
```

7.7.3.20 no2_send

```
int no2_send = 0
```

7.7.3.21 no2_voltage

```
float no2_voltage
```

7.7.3.22 NO2Sensor

```
NO2SensorClass NO2Sensor
```

7.7.3.23 polutants2_particles

```
float polutants2_particles
```

7.7.3.24 polutants2_resistance

```
float polutants2_resistance
```

7.7.3.25 polutants2_send

```
int polutants2_send = 0
```

7.7.3.26 polutants2_voltage

```
float polutants2_voltage
```

7.7.3.27 polutants_particles

```
float polutants_particles
```

7.7.3.28 polutants_resistance

```
float polutants_resistance
```

7.7.3.29 polutants_send

```
int polutants_send = 0
```

7.7.3.30 polutants_voltage

```
float polutants_voltage
```

7.7.3.31 port

```
uint8_t port = 10
```

Define port to use in Back-End: from 1 to 223.

7.7.3.32 resistances_co

```
float resistances_co[ ] = {POINT1_RES_CO, POINT2_RES_CO, POINT3_RES_CO}
```

7.7.3.33 size_of_buffer

```
uint8_t size_of_buffer = sizeof(bytes) / sizeof(bytes[0])
```

7.7.3.34 socket

```
uint8_t socket = SOCKET0
```

Device parameters for Back-End registration.

7.7.3.35 voltages_no2

```
float voltages_no2[] = {POINT1_RES_NO2, POINT2_RES_NO2, POINT3_RES_NO2}
```

7.8 Codes/Smart_Water.cpp File Reference

```
#include <WaspLoRaWAN.h>
#include <WaspSensorSW.h>
#include <lora_send.h>
#include <lora_join.h>
```

Macros

- #define CALIBRATION_POINT_10 1.985

  pH - 10, 1.985 Volts
- #define CALIBRATION_POINT_7 2.070

  pH - 7, 2.070 Volts
- #define CALIBRATION_POINT_4 2.227

  pH - 4, 2.227 Volts
- #define CALIBRATION_TEMPERATURE 23.7

Functions

- void setup ()
- void loop ()

Variables

- uint8_t socket = SOCKET0

  Socket Selection for the Sensor Probe.
- char device_eui [ ] = "72c465dcf469a174"

  Device parameters for Back-End registration.
- char app_eui [ ] = "72c465dcf469a174"
- char app_key [ ] = "66c203b5c8149813ef44cb88c5bd88be"
- uint8_t port = 10

  Define port to use in Back-End: from 1 to 223.
- uint8_t bytes [11] = {0}

  Define data payload to send (maximum is up to data rate)
- int battery_level = 0

  Variables.
- float pH_voltage
- float temperature
- float pH_value
- int16_t temperature_send
- int pH_value_send
- uint8_t size_of_buffer = sizeof(bytes) / sizeof(bytes[0])
- uint8_t adr_flag = 0

  Adaptive Data Rate indicator for lora_join()
- pHClass pHSensor
- pt1000Class temperatureSensor

7.8.1 Macro Definition Documentation

7.8.1.1 CALIBRATION_POINT_10

```
#define CALIBRATION_POINT_10 1.985
```

pH - 10, 1.985 Volts

7.8.1.2 CALIBRATION_POINT_4

```
#define CALIBRATION_POINT_4 2.227
```

pH - 4, 2.227 Volts

7.8.1.3 CALIBRATION_POINT_7

```
#define CALIBRATION_POINT_7 2.070
```

pH - 7, 2.070 Volts

7.8.1.4 CALIBRATION_TEMPERATURE

```
#define CALIBRATION_TEMPERATURE 23.7
```

7.8.2 Function Documentation

7.8.2.1 loop()

```
void loop ( )
```

7.8.2.2 setup()

```
void setup ( )
```

7.8.3 Variable Documentation

7.8.3.1 adr_flag

```
uint8_t adr_flag = 0
```

Adaptive Data Rate indicator for lora_join()

7.8.3.2 app_eui

```
char app_eui[ ] = "72c465dcf469a174"
```

7.8.3.3 app_key

```
char app_key[ ] = "66c203b5c8149813ef44cb88c5bd88be"
```

7.8.3.4 battery_level

```
int battery_level = 0
```

Variables.

7.8.3.5 bytes

```
uint8_t bytes[11] = {0}
```

Define data payload to send (maximum is up to data rate)

7.8.3.6 device_eui

```
char device_eui[ ] = "72c465dcf469a174"
```

Device parameters for Back-End registration.

7.8.3.7 pH_value

```
float pH_value
```

7.8.3.8 pH_value_send

```
int pH_value_send
```

7.8.3.9 pH_voltage

```
float pH_voltage
```

7.8.3.10 pHSensor

```
pHClass pHSensor
```

7.8.3.11 port

```
uint8_t port = 10
```

Define port to use in Back-End: from 1 to 223.

7.8.3.12 size_of_buffer

```
uint8_t size_of_buffer = sizeof(bytes) / sizeof(bytes[0])
```

7.8.3.13 socket

```
uint8_t socket = SOCKET0
```

Socket Selection for the Sensor Probe.

7.8.3.14 temperature

```
float temperature
```

7.8.3.15 temperature_send

```
int16_t temperature_send
```

7.8.3.16 temperatureSensor

```
pt1000Class temperatureSensor
```

7.9 doc_pages/Agriculture.md File Reference

7.10 doc_pages/Events.md File Reference

7.11 doc_pages/Gas.md File Reference

7.12 doc_pages/Main.md File Reference

7.13 doc_pages/Water.md File Reference

Index