Projet Bee-Honey't

version 0.2

BTS SNIR LaSalle Avignon 2020

Table des matières

1 Le projet **2**

1.1 Table des matières . . . 2

1.2 Informations . . . 2

2 Changelog **2**

3 README **2**

3.1 Projet . . . 2

3.1.1 Présentation . . . 2

3.1.2 Recette de l'application PC (desktop) . . . 3

3.1.3 Informations . . . 3

4 A propos **4**

5 Licence GPL **4**

6 Liste des choses à faire **4**

7 Documentation des espaces de nommage **4**

7.1 Référence de l'espace de nommage Ui . . . 4

8 Documentation des classes **4**

8.1 Référence de la classe Communication . . . 4

8.1.1 Description détaillée . . . 6

8.1.2 Documentation des constructeurs et destructeur . . . 7

8.1.3 Documentation des fonctions membres . . . 7

8.1.4 Documentation des données membres . . . 18

8.2 Référence de la classe Configuration . . . 18

8.2.1 Description détaillée . . . 20

8.2.2 Documentation des constructeurs et destructeur . . . 20

8.2.3 Documentation des fonctions membres . . . 21

8.2.4 Documentation des données membres . . . 26

8.3 Référence de la structure ConfigurationTTN . . . 27

8.3.1 Description détaillée . . . 28

8.3.2 Documentation des données membres . . . 28

8.4 Référence de la classe Ihm . . . 29

8.4.1 Description détaillée . . . 32

8.4.2 Documentation des constructeurs et destructeur . . . 32

8.4.3 Documentation des fonctions membres . . . 33

8.4.4 Documentation des données membres . . . 54

8.5 Référence de la classe IHMNouvelleRuche . . . 59

8.5.1 Description détaillée . . . 60

8.5.2 Documentation des constructeurs et destructeur . . . 60

8.5.3 Documentation des fonctions membres . . . 61

8.5.4 Documentation des données membres . . . 63

8.6 Référence de la classe IHMReglageRuche . . . 64

8.6.1 Description détaillée . . . 65

8.6.2 Documentation des constructeurs et destructeur . . . 65

8.6.3 Documentation des fonctions membres . . . 66

8.6.4 Documentation des données membres . . . 66

8.7 Référence de la classe QDialog . . . 67

8.8 Référence de la classe QMainWindow . . . 67

8.9 Référence de la classe QObject . . . 67

8.10 Référence de la structure Ruche . . . 68

8.10.1 Description détaillée . . . 68

8.10.2 Documentation des fonctions membres . . . 68

8.10.3 Documentation des données membres . . . 69

9 Documentation des fichiers **71**

9.1 Référence du fichier Changelog.md . . . 71
9.2 Changelog.md . . . 71
9.3 Référence du fichier communication.cpp . . . 72
9.3.1 Description détaillée . . . 73
9.4 communication.cpp . . . 73
9.5 Référence du fichier communication.h . . . 75
9.5.1 Description détaillée . . . 75
9.6 communication.h . . . 76
9.7 Référence du fichier configuration.cpp . . . 76
9.7.1 Description détaillée . . . 77
9.8 configuration.cpp . . . 77
9.9 Référence du fichier configuration.h . . . 78
9.9.1 Description détaillée . . . 79
9.10 configuration.h . . . 79
9.11 Référence du fichier ihm.cpp . . . 80
9.11.1 Description détaillée . . . 80
9.12 ihm.cpp . . . 80
9.13 Référence du fichier ihm.h . . . 89
9.13.1 Description détaillée . . . 89
9.13.2 Documentation des macros . . . 90
9.13.3 Documentation du type de l'énumération . . . 90
9.14 ihm.h . . . 91
9.15 Référence du fichier main.cpp . . . 92
9.15.1 Description détaillée . . . 93
9.15.2 Documentation des fonctions . . . 93
9.16 main.cpp . . . 94
9.17 Référence du fichier nouvelleruche.cpp . . . 94
9.17.1 Description détaillée . . . 94
9.18 nouvelleruche.cpp . . . 94
9.19 Référence du fichier nouvelleruche.h . . . 95
9.19.1 Description détaillée . . . 96
9.20 nouvelleruche.h . . . 96
9.21 Référence du fichier README.md . . . 96
9.22 README.md . . . 96
9.23 Référence du fichier reglageruche.cpp . . . 98
9.23.1 Description détaillée . . . 98
9.24 reglageruche.cpp . . . 98
9.25 Référence du fichier reglageruche.h . . . 98
9.25.1 Description détaillée . . . 99
9.26 reglageruche.h . . . 99
9.27 Référence du fichier ruche.h . . . 99
9.27.1 Description détaillée . . . 100
9.28 ruche.h . . . 100

Index 101

1 Le projet

Système autonome permettant de connaître à distance certains paramètres d'une ruche afin d'assurer son suivi et d'évaluer la santé des abeilles.

1.1 Table des matières

— README
— Changelog
— A propos
— Licence GPL

1.2 Informations

Auteur

Théo Ackermann theoackrm@gmail.com

Date

2020

Version

0.1

Voir également

https://svn.riouxsvn.com/

2 Changelog

Revision : 35 Author : tackermann Date : vendredi 3 avril 2020 05 :44 :03 Message :

Tag 0.1, Bouml

Revision : 31 Author : tackermann Date : jeudi 2 avril 2020 16 :33 :47 Message : Suppression des

3 README

3.1 Projet

3.1.1 Présentation

Système autonome permettant de connaître à distance certains paramètres d'une ruche afin d'assurer son suivi et d'évaluer la santé des abeilles.

Version : PC (*desktop*)

Fichier de configuration

Le fichier `configuration.ini` contient :

- les paramètres de connexion au serveur TTN
- le nombre de ruches
- les paramètres de chaque ruche

```
[General]
NbRuches=2

[Ruche1]
Adresse=
Latitude=
Longitude=
MiseEnService=
Nom=Ruche 1
TopicTTN=mes_ruches/devices/ruche_1/up

[Ruche2]
Adresse=
Latitude=
Longitude=
MiseEnService=25/03/2020
Nom=Ruche 2
TopicTTN=mes_ruches/devices/ruche_2/up

[TTN]
Hostname=eu.thethings.network
Password=
Port=1883
Username=mes_ruches
```

3.1.2 Recette de l'application PC (desktop)

- Consulter les données d'une ruche
- Recevoir les données actuelles des ruches (MQTT/Json)
- Éditer les ruches

3.1.3 Informations

Auteur

Théo Ackermann theoackrm@gmail.com

Date

2020

Version

0.2

Voir également

https://svn.riouxsvn.com/bee-honey-t

4 A propos

Auteur

Théo Ackermann theoackrm@gmail.com

Date

2020

Version

0.2

Voir également

https://svn.riouxsvn.com/bee-honey-t

5 Licence GPL

6 Liste des choses à faire

Membre Ihm : :initialiserGraphiqueTemperatures ()
Faire un axeX commun pour tout les graphiques

Membre Ihm : :setValeurTemperatureInterieure (QString nomDeLaRuche, double temperatureInterieure, QString horodatage)
Intégrer l'horodatage pour la mesure (cf. toMSecsSinceEpoch() de la classe QDateTime) pour l'instant l'axe X est une valeur 0,1, ...

7 Documentation des espaces de nommage

7.1 Référence de l'espace de nommage Ui

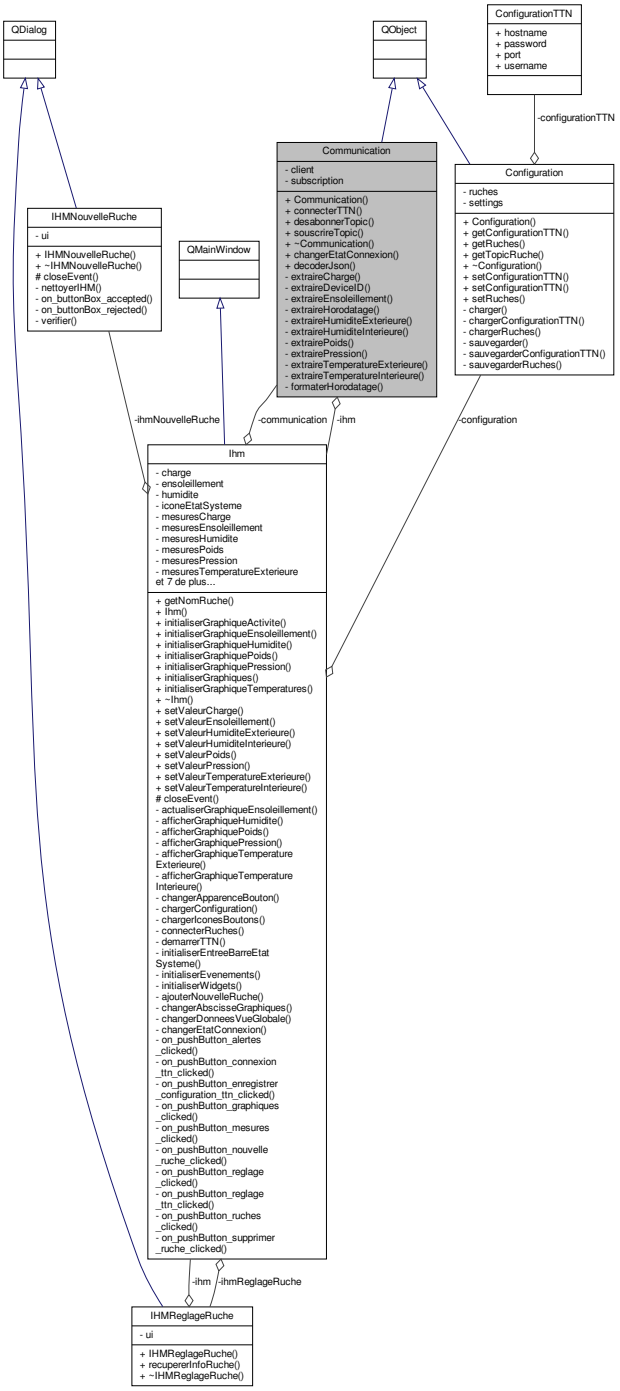
8 Documentation des classes

8.1 Référence de la classe Communication

Permet de recevoir les données.

`#include <communication.h>`

Graphe de collaboration de Communication :



Connecteurs publics

— void changerEtatConnexion ()
Méthode pour notifier un changement d'état de la connexion TTN.
— void decoderJson (const QByteArray &json)
Méthode pour décoder le JSON reçu.

Signaux

— void nouvelEtatConnexion (int etat)

- void nouvelleValeurCharge (QString nomDeLaRuche, int charge, QString horodatage)
- void nouvelleValeurEnsoleillement (QString nomDeLaRuche, int ensoleillement, QString horodatage)
- void nouvelleValeurHumiditeExterieure (QString nomDeLaRuche, double humiditeExterieure, QString horodatage)
- void nouvelleValeurHumiditeInterieure (QString nomDeLaRuche, double humiditeInterieure, QString horodatage)
- void nouvelleValeurPoids (QString nomDeLaRuche, double poids, QString horodatage)
- void nouvelleValeurPression (QString nomDeLaRuche, int pression, QString horodatage)
- void nouvelleValeurTemperatureExterieure (QString nomDeLaRuche, double temperatureExterieure, QString horodatage)
- void nouvelleValeurTemperatureInterieure (QString nomDeLaRuche, double temperatureInterieure, QString horodatage)

Fonctions membres publiques

- Communication (QObject *parent=nullptr)
  Constructeur de la classe Communication.
- void connecterTTN (QString hostname, int port, QString username, QString password)
  Méthode pour se connecter à TTN.
- void desabonnerTopic (QString topic)
  Méthode pour se désabonner d'un topic TTN.
- void souscrireTopic (QString topic)
  Méthode pour s'abonner à un topic TTN.
- ~Communication ()
  Destructeur de la classe Communication.

Fonctions membres privées

- int extraireCharge (QJsonObject objetJSON)
  Méthode pour extraire la charge de la batterie le l'objet JSON.
- QString extraireDeviceID (QJsonObject objetJSON, QStringList listeCles, int position)
  Méthode pour extraire le device ID de l'objet JSON.
- int extraireEnsoleillement (QJsonObject objetJSON)
  Méthode pour extraire l'ensoleillement de l'objet JSON.
- QString extraireHorodatage (QJsonObject objetJSON)
  Méthode pour extraire le temps de l'objet JSON.
- double extraireHumiditeExterieure (QJsonObject objetJSON)
  Méthode pour extraire l'humidité extérieure de l'objet JSON.
- double extraireHumiditeInterieure (QJsonObject objetJSON)
  Méthode pour extraire l'humidité intérieure de l'objet JSON.
- double extrairePoids (QJsonObject objetJSON)
  Méthode pour extraire le poids le l'objet JSON.
- int extrairePression (QJsonObject objetJSON)
  Méthode pour extraire la pression de l'objet JSON.
- double extraireTemperatureExterieure (QJsonObject objetJSON)
  Méthode pour extraire la température extérieure de l'objet JSON.
- double extraireTemperatureInterieure (QJsonObject objetJSON)
  Méthode pour extraire la température intérieure de l'objet JSON.
- QString formaterHorodatage (QString horodatageBrut)
  Méthode pour mettre dans le bon format l'horodatage reçu.

Attributs privés

- QMqttClient * client
- Ihm * ihm
  interface utilisateur
- QMqttSubscription * subscription

8.1.1 Description détaillée

Permet de recevoir les données.

Auteur

ACKERMANN Théo

Version

0.2

Définition à la ligne 24 du fichier communication.h.

8.1.2 Documentation des constructeurs et destructeur

8.1.2.1 Communication()

```
Communication::Communication (
            QObject * parent = nullptr )
```

Constructeur de la classe Communication.

Paramètres

parent	

Définition à la ligne 16 du fichier communication.cpp.

Références changerEtatConnexion(), client, et decoderJson().

```
00016                                           : QObject(parent), client(new QMqttClient(this))
00017 {
00018     qDebug() << Q_FUNC_INFO;
00019     connect(client, SIGNAL(stateChanged(ClientState)), this, SLOT(
      changerEtatConnexion()));
00020     connect(client, SIGNAL(messageReceived(const QByteArray &, const QMqttTopicName &)), this, SLOT(
      decoderJson(const QByteArray &)));
00021 }
```

8.1.2.2 ~Communication()

```
Communication::~Communication ( )
```

Destructeur de la classe Communication.

Définition à la ligne 26 du fichier communication.cpp.

Références client.

```
00027 {
00028     if(client->state() == QMqttClient::Connected)
00029     {
00030         client->disconnectFromHost();
00031     }
00032     qDebug() << Q_FUNC_INFO;
00033 }
```

8.1.3 Documentation des fonctions membres

8.1.3.1 changerEtatConnexion

```
void Communication::changerEtatConnexion ( )  [slot]
```

Méthode pour notifier un changement d'état de la connexion TTN.

Définition à la ligne 282 du fichier communication.cpp.

Références client, et nouvelEtatConnexion().

Référencé par Communication().

```
00283 {
00284     QString message;
00285     switch(client->state())
00286     {
00287         case 0:
00288             message = "Déconnecté";
00289             break;
00290         case 1:
00291             message = "En cours de connexion";
00292             break;
00293         case 2:
00294             message = "Connecté";
00295             break;
00296     }
00297     qDebug()<< Q_FUNC_INFO << client->state() << message;
00298     emit nouvelEtatConnexion(client->state());
00299 }
```

8.1.3.2 connecterTTN()

```
void Communication::connecterTTN (
            QString hostname,
            int port,
            QString username,
            QString password )
```

Méthode pour se connecter à TTN.

Paramètres

hostname	
port	
username	
password	

Définition à la ligne 43 du fichier communication.cpp.

Références client.

Référencé par Ihm : :demarrerTTN(), et Ihm : :on_pushButton_connexion_ttn_clicked().

```
00044 {
00045     qDebug() << Q_FUNC_INFO << hostname << port << username << password;
00046     if(client->state() == QMqttClient::Disconnected)
00047     {
00048         client->setHostname(hostname);
00049         client->setPort(port);
00050         client->setUsername(username);
00051         client->setPassword(password);
```

```
00052         client->connectToHost();
00053     }
00054     else if(client->state() == QMqttClient::Connected)
00055     {
00056         client->disconnectFromHost();
00057     }
00058 }
```

8.1.3.3 decoderJson

```
void Communication::decoderJson (
             const QByteArray & json )  [slot]
```

Méthode pour décoder le JSON reçu.

Paramètres

json	

Définition à la ligne 93 du fichier communication.cpp.

Références extraireCharge(), extraireDeviceID(), extraireEnsoleillement(), extraireHorodatage(), extraireHumiditeExterieure(), extraireHumiditeInterieure(), extrairePoids(), extrairePression(), extraireTemperatureExterieure(), extraireTemperatureInterieure(), formaterHorodatage(), nouvelleValeurCharge(), nouvelleValeurEnsoleillement(), nouvelleValeurHumiditeExterieure(), nouvelleValeurHumiditeInterieure(), nouvelleValeurPoids(), nouvelleValeurPression(), nouvelleValeurTemperatureExterieure(), et nouvelleValeurTemperatureInterieure().

Référencé par Communication().

```
00094 {
00095     QJsonDocument documentJSON = QJsonDocument::fromJson(json);
00096     QString nomDeLaRuche;
00097     QString horodatage;
00098
00099     qDebug() << Q_FUNC_INFO << json;
00100     if(!documentJSON.isNull())
00101     {
00102         QJsonObject objetJSON = documentJSON.object();
00103         QStringList listeCles = objetJSON.keys();
00104         // les clés sont triés alphabétiquement
00105         for(int i = 0; i < listeCles.count()-1; i++)
00106         {
00107             if(listeCles.at(i) == "dev_id")
00108             {
00109                 nomDeLaRuche = extraireDeviceID(objetJSON, listeCles, i);
00110             }
00111             if(listeCles.at(i) == "metadata")
00112             {
00113                 horodatage = formaterHorodatage(
      extraireHorodatage(objetJSON[listeCles.at(i)].toObject()));
00114             }
00115             if(listeCles.at(i) == "payload_fields")
00116             {
00117                 QJsonObject objet = objetJSON[listeCles.at(i)].toObject();
00118
00119                 if(objet.contains("temperatureInt"))
00120                 {
00121                     emit nouvelleValeurTemperatureInterieure(
      nomDeLaRuche, extraireTemperatureInterieure(objet), horodatage);
00122                 }
00123                 if(objet.contains("temperatureExt"))
00124                 {
00125                     emit nouvelleValeurTemperatureExterieure(
      nomDeLaRuche, extraireTemperatureExterieure(objet), horodatage);
00126                 }
00127                 if(objet.contains("humiditeInt"))
00128                 {
00129                     emit nouvelleValeurHumiditeInterieure(nomDeLaRuche,
      extraireHumiditeInterieure(objet), horodatage);
00130                 }
00131                 if(objet.contains("humiditeExt"))
```

```
00132                 {
00133                     emit nouvelleValeurHumiditeExterieure(nomDeLaRuche,
      extraireHumiditeExterieure(objet), horodatage);
00134                 }
00135                 if(objet.contains("ensoleillement"))
00136                 {
00137                     emit nouvelleValeurEnsoleillement(nomDeLaRuche,
      extraireEnsoleillement(objet), horodatage);
00138                 }
00139                 if(objet.contains("pression"))
00140                 {
00141                     emit nouvelleValeurPression(nomDeLaRuche,
      extrairePression(objet), horodatage);
00142                 }
00143                 if(objet.contains("poids"))
00144                 {
00145                     emit nouvelleValeurPoids(nomDeLaRuche,
      extrairePoids(objet), horodatage);
00146                 }
00147                 if(objet.contains("charge"))
00148                 {
00149                     emit nouvelleValeurCharge(nomDeLaRuche,
      extraireCharge(objet), horodatage);
00150                 }
00151             }
00152         }
00153     }
00154 }
```

8.1.3.4 desabonnerTopic()

```
void Communication::desabonnerTopic (
            QString topic )
```

Méthode pour se désabonner d'un topic TTN.

Paramètres

topic	

Définition à la ligne 79 du fichier communication.cpp.

Références client.

Référencé par Ihm : :on_pushButton_supprimer_ruche_clicked().

```
00080 {
00081     if(client->state() == QMqttClient::Connected)
00082     {
00083         client->unsubscribe(topic);
00084         qDebug() << Q_FUNC_INFO << topic;
00085     }
00086 }
```

8.1.3.5 extraireCharge()

```
int Communication::extraireCharge (
            QJsonObject objetJSON )  [private]
```

Méthode pour extraire la charge de la batterie le l'objet JSON.

Paramètres

objetJSON	

Renvoie

int

Définition à la ligne 262 du fichier communication.cpp.

Référencé par decoderJson().

```
00263 {
00264     return objetJSON.value(QString("charge")).toInt();
00265 }
```

8.1.3.6 extraireDeviceID()

```
QString Communication::extraireDeviceID (
            QJsonObject objetJSON,
            QStringList listeCles,
            int position )  [private]
```

Méthode pour extraire le device ID de l'objet JSON.

Paramètres

objetJSON	
listeCles	
position	

Renvoie

QString

Définition à la ligne 175 du fichier communication.cpp.

Référencé par decoderJson().

```
00176 {
00177     return objetJSON[listeCles.at(position)].toString();
00178 }
```

8.1.3.7 extraireEnsoleillement()

```
int Communication::extraireEnsoleillement (
            QJsonObject objetJSON )  [private]
```

Méthode pour extraire l'ensoleillement de l'objet JSON.

Paramètres

objetJSON	

Renvoie

int

Définition à la ligne 229 du fichier communication.cpp.

Référencé par decoderJson().

```
00230 {
00231     return objetJSON.value(QString("ensoleillement")).toInt();
00232 }
```

8.1.3.8 extraireHorodatage()

```
QString Communication::extraireHorodatage (
            QJsonObject objetJSON )  [private]
```

Méthode pour extraire le temps de l'objet JSON.

Paramètres

objetJSON	

Renvoie

QString

Définition à la ligne 162 du fichier communication.cpp.

Référencé par decoderJson().

```
00163 {
00164     return objetJSON.value(QString("time")).toString();
00165 }
```

8.1.3.9 extraireHumiditeExterieure()

```
double Communication::extraireHumiditeExterieure (
            QJsonObject objetJSON )  [private]
```

Méthode pour extraire l'humidité extérieure de l'objet JSON.

Paramètres

objetJSON	

Renvoie

double

Définition à la ligne 219 du fichier communication.cpp.

Référencé par decoderJson().

```
00220 {
00221     return objetJSON.value(QString("humiditeExt")).toDouble();
00222 }
```

8.1.3.10 extraireHumiditeInterieure()

```
double Communication::extraireHumiditeInterieure (
            QJsonObject objetJSON )  [private]
```

Méthode pour extraire l'humidité intérieure de l'objet JSON.

Paramètres

objetJSON	

Renvoie

double

Définition à la ligne 208 du fichier communication.cpp.

Référencé par decoderJson().

```
00209 {
00210     return objetJSON.value(QString("humiditeInt")).toDouble();
00211 }
```

8.1.3.11 extrairePoids()

```
double Communication::extrairePoids (
            QJsonObject objetJSON )  [private]
```

Méthode pour extraire le poids le l'objet JSON.

Paramètres

objetJSON	

Renvoie

double

Définition à la ligne 251 du fichier communication.cpp.

Référencé par decoderJson().

```
00252 {
00253     return objetJSON.value(QString("poids")).toDouble();
00254 }
```

8.1.3.12 extrairePression()

```
int Communication::extrairePression (
            QJsonObject objetJSON )  [private]
```

Méthode pour extraire la pression de l'objet JSON.

Paramètres

objetJSON	

Renvoie

int

Définition à la ligne 240 du fichier communication.cpp.

Référencé par decoderJson().

```
00241 {
00242     return objetJSON.value(QString("pression")).toInt();
00243 }
```

8.1.3.13 extraireTemperatureExterieure()

```
double Communication::extraireTemperatureExterieure (
            QJsonObject objetJSON )  [private]
```

Méthode pour extraire la température extérieure de l'objet JSON.

Paramètres

objetJSON	

Renvoie

double

Définition à la ligne 197 du fichier communication.cpp.

Référencé par decoderJson().

```
00198 {
00199     return objetJSON.value(QString("temperatureExt")).toDouble();
00200 }
```

8.1.3.14 extraireTemperatureInterieure()

```
double Communication::extraireTemperatureInterieure (
            QJsonObject objetJSON )  [private]
```

Méthode pour extraire la température intérieure de l'objet JSON.

Paramètres

objetJSON	

Renvoie

double

Définition à la ligne 186 du fichier communication.cpp.

Référencé par decoderJson().

```
00187 {
00188     return objetJSON.value(QString("temperatureInt")).toDouble();
00189 }
```

8.1.3.15 formaterHorodatage()

```
QString Communication::formaterHorodatage (
            QString horodatageBrut )  [private]
```

Méthode pour mettre dans le bon format l'horodatage reçu.

Paramètres

horodatageBrut	

Renvoie

QString

Définition à la ligne 273 du fichier communication.cpp.

Référencé par decoderJson().

```
00274 {
00275     QDateTime horodatage = QDateTime::fromString(horodatageBrut, Qt::ISODate).toLocalTime();
00276     return horodatage.toString("dd/MM/yyyy HH:mm:ss");
00277 }
```

8.1.3.16 nouvelEtatConnexion

```
void Communication::nouvelEtatConnexion (
            int etat )  [signal]
```

Référencé par changerEtatConnexion().

8.1.3.17 nouvelleValeurCharge

```
void Communication::nouvelleValeurCharge (
            QString nomDeLaRuche,
            int charge,
            QString horodatage )  [signal]
```

Référencé par decoderJson().

8.1.3.18 nouvelleValeurEnsoleillement

```
void Communication::nouvelleValeurEnsoleillement (
            QString nomDeLaRuche,
            int ensoleillement,
            QString horodatage )  [signal]
```

Référencé par decoderJson().

8.1.3.19 nouvelleValeurHumiditeExterieure

```
void Communication::nouvelleValeurHumiditeExterieure (
            QString nomDeLaRuche,
            double humiditeExterieure,
            QString horodatage )  [signal]
```

Référencé par decoderJson().

8.1.3.20 nouvelleValeurHumiditeInterieure

```
void Communication::nouvelleValeurHumiditeInterieure (
            QString nomDeLaRuche,
            double humiditeInterieure,
            QString horodatage )  [signal]
```

Référencé par decoderJson().

8.1.3.21 nouvelleValeurPoids

```
void Communication::nouvelleValeurPoids (
            QString nomDeLaRuche,
            double poids,
            QString horodatage )  [signal]
```

Référencé par decoderJson().

8.1.3.22 nouvelleValeurPression

```
void Communication::nouvelleValeurPression (
            QString nomDeLaRuche,
            int pression,
            QString horodatage )  [signal]
```

Référencé par decoderJson().

8.1.3.23 nouvelleValeurTemperatureExterieure

```
void Communication::nouvelleValeurTemperatureExterieure (
            QString nomDeLaRuche,
            double temperatureExterieure,
            QString horodatage )  [signal]
```

Référencé par decoderJson().

8.1.3.24 nouvelleValeurTemperatureInterieure

```
void Communication::nouvelleValeurTemperatureInterieure (
            QString nomDeLaRuche,
            double temperatureInterieure,
            QString horodatage )  [signal]
```

Référencé par decoderJson().

8.1.3.25 souscrireTopic()

```
void Communication::souscrireTopic (
            QString topic )
```

Méthode pour s'abonner à un topic TTN.

Paramètres

topic	

Définition à la ligne 65 du fichier communication.cpp.

Références client, et subscription.

Référencé par Ihm : :ajouterNouvelleRuche(), et Ihm : :connecterRuches().

```
00066 {
00067     if(client->state() == QMqttClient::Connected)
00068     {
00069         subscription = client->subscribe(QMqttTopicFilter(topic));
00070         qDebug() << Q_FUNC_INFO << topic;
00071     }
00072 }
```

8.1.4 Documentation des données membres

8.1.4.1 client

```
QMqttClient* Communication::client  [private]
```

Définition à la ligne 37 du fichier communication.h.

Référencé par changerEtatConnexion(), Communication(), connecterTTN(), desabonnerTopic(), souscrireTopic(), et ~Communication().

8.1.4.2 ihm

```
Ihm* Communication::ihm  [private]
```

interface utilisateur

Définition à la ligne 39 du fichier communication.h.

8.1.4.3 subscription

```
QMqttSubscription* Communication::subscription  [private]
```

Définition à la ligne 38 du fichier communication.h.

Référencé par souscrireTopic().

La documentation de cette classe a été générée à partir des fichiers suivants :

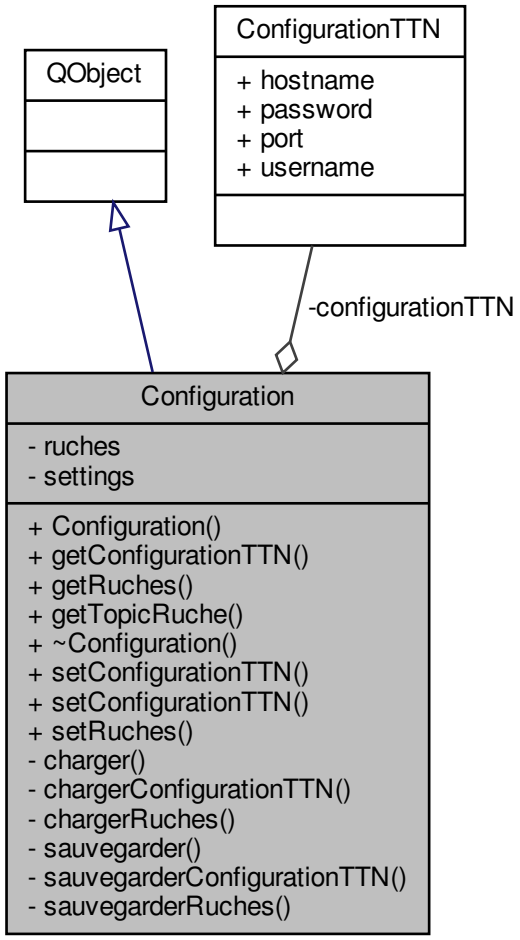
- communication.h
- communication.cpp

8.2 Référence de la classe Configuration

Gère le fichier INI.

```
#include <configuration.h>
```

Graphe de collaboration de Configuration :



Connecteurs publics

- void setConfigurationTTN (ConfigurationTTN configurationTTN)
  Méthode pour définir une configuration TTN.
- void setConfigurationTTN (QString hostname, int port, QString username, QString password)
  Méthode pour définir une configuration TTN.
- void setRuches (QVector< Ruche > ruches)
  Méthode pour définir une ruche.

Fonctions membres publiques

- Configuration (QObject *parent=nullptr)
  Constructeur de la classe Configuration.
- ConfigurationTTN getConfigurationTTN () const
  Méthode pour récupérer la configuration TTN.
- QVector< Ruche > getRuches () const
  Méthode pour récupérer une ruche.
- QString getTopicRuche (QString ruche)
  Méthode qui retourne le topic TTN d'une ruche.
- ~Configuration ()
  Destructeur de la classe Configuration.

Fonctions membres privées

- void charger ()
  Méthode pour charger la configuration TTN et les ruches depuis le fichier INI.
- void chargerConfigurationTTN ()
  Méthode pour charger la configuration TTN.
- void chargerRuches ()
  Méthode pour charger les ruches depuis le fichier INI.
- void sauvegarder ()
  Méthode pour sauvegarder la configuration TTN et les ruches dans le fichier INI.
- void sauvegarderConfigurationTTN ()
  Méthode pour sauvegarder la configuration TTN dans le fichier INI.
- void sauvegarderRuches ()
  Méthode pour sauvegarder les ruches dans le fichier INI.

Attributs privés

- ConfigurationTTN configurationTTN
  configuration MQTT pour se connecter au réseau TheThingsNetwork (TTN)
- QVector< Ruche > ruches
  les ruches
- QSettings settings
  objet pour gérer un fichier .ini

8.2.1 Description détaillée

Gère le fichier INI.

Auteur

ACKERMANN Théo

Version

0.2

Définition à la ligne 37 du fichier configuration.h.

8.2.2 Documentation des constructeurs et destructeur

8.2.2.1 Configuration()

```
Configuration::Configuration (
            QObject * parent = nullptr )
```

Constructeur de la classe Configuration.

Paramètres

parent	
settings	

Définition à la ligne 16 du fichier configuration.cpp.

Références charger().

```
00016                                                : QObject(parent), settings(QDir::currentPath()
      + "/configuration.ini", QSettings::IniFormat)
00017 {
00018     qDebug() << Q_FUNC_INFO;
00019     charger();
00020 }
```

8.2.2.2 ~Configuration()

```
Configuration::~Configuration ( )
```

Destructeur de la classe Configuration.

Définition à la ligne 25 du fichier configuration.cpp.

Références sauvegarder().

```
00026 {
00027     sauvegarder();
00028     qDebug() << Q_FUNC_INFO;
00029 }
```

8.2.3 Documentation des fonctions membres

8.2.3.1 charger()

```
void Configuration::charger ( )  [private]
```

Méthode pour charger la configuration TTN et les ruches depuis le fichier INI.

Définition à la ligne 103 du fichier configuration.cpp.

Références chargerConfigurationTTN(), chargerRuches(), et settings.

Référencé par Configuration().

```
00104 {
00105     qDebug() << Q_FUNC_INFO << settings.allKeys() << settings.childKeys();
00106     chargerConfigurationTTN();
00107     chargerRuches();
00108 }
```

8.2.3.2 chargerConfigurationTTN()

```
void Configuration::chargerConfigurationTTN ( )  [private]
```

Méthode pour charger la configuration TTN.

Définition à la ligne 113 du fichier configuration.cpp.

Références configurationTTN, ConfigurationTTN : :hostname, ConfigurationTTN : :password, ConfigurationTTN : :port, settings, et ConfigurationTTN : :username.

Référencé par charger().

```
00114 {
00115     settings.beginGroup("TTN");
00116     configurationTTN.hostname = settings.value("Hostname").toString();
00117     configurationTTN.port = settings.value("Port").toInt();
00118     configurationTTN.username = settings.value("Username").toString();
00119     configurationTTN.password = settings.value("Password").toString();
00120     settings.endGroup();
00121     settings.sync();
00122     qDebug() << Q_FUNC_INFO << configurationTTN.hostname <<
      configurationTTN.port <<  configurationTTN.
      username << configurationTTN.password;
00123 }
```

8.2.3.3 chargerRuches()

```
void Configuration::chargerRuches ( )  [private]
```

Méthode pour charger les ruches depuis le fichier INI.

Définition à la ligne 128 du fichier configuration.cpp.

Références Ruche : :adresse, Ruche : :latitude, Ruche : :longitude, Ruche : :miseEnService, Ruche : :nom, ruches, settings, et Ruche : :topicTTN.

Référencé par charger().

```
00129 {
00130     int nbRuches = settings.value("NbRuches", 0).toInt();
00131     qDebug() << Q_FUNC_INFO << nbRuches;
00132     for(int i = 0; i < nbRuches; i++)
00133     {
00134         Ruche ruche;
00135         QString nomRuche = "Ruche" + QString::number(i+1);
00136         settings.beginGroup(nomRuche);
00137         ruche.nom = settings.value("Nom").toString();
00138         ruche.topicTTN = settings.value("TopicTTN").toString();
00139         ruche.miseEnService = settings.value("MiseEnService").toString();
00140         ruche.adresse = settings.value("Adresse").toString();
00141         ruche.latitude = settings.value("Latitude").toString();
00142         ruche.longitude = settings.value("Longitude").toString();
00143         settings.endGroup();
00144         ruches.push_back(ruche);
00145     }
00146 }
```

8.2.3.4 getConfigurationTTN()

```
ConfigurationTTN Configuration::getConfigurationTTN ( ) const
```

Méthode pour récupérer la configuration TTN.

Renvoie

ConfigurationTTN

Définition à la ligne 63 du fichier configuration.cpp.

Références configurationTTN.

Référencé par Ihm : :chargerConfiguration(), et Ihm : :demarrerTTN().

```
00064 {
00065     return configurationTTN;
00066 }
```

8.2.3.5 getRuches()

```
QVector< Ruche > Configuration::getRuches ( ) const
```

Méthode pour récupérer une ruche.

Renvoie

QVector<Ruche>

Définition à la ligne 83 du fichier configuration.cpp.

Références ruches.

Référencé par Ihm : :chargerConfiguration().

```
00084 {
00085     return ruches;
00086 }
```

8.2.3.6 getTopicRuche()

```
QString Configuration::getTopicRuche (
            QString ruche )
```

Méthode qui retourne le topic TTN d'une ruche.

Paramètres

ruche	

Renvoie

QString

Définition à la ligne 94 du fichier configuration.cpp.

Références settings.

```
00095 {
00096     ruche = ruche.replace(" ","");
00097     ruche = ruche + "/TopicTTN";
00098     return settings.value(ruche).toString();
00099 }
```

8.2.3.7 sauvegarder()

```
void Configuration::sauvegarder ( )  [private]
```

Méthode pour sauvegarder la configuration TTN et les ruches dans le fichier INI.

Définition à la ligne 151 du fichier configuration.cpp.

Références sauvegarderConfigurationTTN(), et sauvegarderRuches().

Référencé par ~Configuration().

```
00152 {
00153     qDebug() << Q_FUNC_INFO;
00154     sauvegarderConfigurationTTN();
00155     sauvegarderRuches();
00156 }
```

8.2.3.8 sauvegarderConfigurationTTN()

```
void Configuration::sauvegarderConfigurationTTN ( )  [private]
```

Méthode pour sauvegarder la configuration TTN dans le fichier INI.

Définition à la ligne 161 du fichier configuration.cpp.

Références configurationTTN, ConfigurationTTN : :hostname, ConfigurationTTN : :password, ConfigurationTTN : :port, settings, et ConfigurationTTN : :username.

Référencé par sauvegarder().

```
00162 {
00163     settings.beginGroup("TTN");
00164     settings.setValue("Hostname", configurationTTN.
      hostname);
00165     settings.setValue("Port", configurationTTN.port);
00166     settings.setValue("Username", configurationTTN.
      username);
00167     settings.setValue("Password", configurationTTN.
      password);
00168     settings.endGroup();
00169 }
```

8.2.3.9 sauvegarderRuches()

```
void Configuration::sauvegarderRuches ( )  [private]
```

Méthode pour sauvegarder les ruches dans le fichier INI.

Définition à la ligne 174 du fichier configuration.cpp.

Références ruches, et settings.

Référencé par sauvegarder().

```
00175 {
00176     qDebug() << Q_FUNC_INFO << ruches.size();
00177     for(int i = 0; i < ruches.size(); i++)
00178     {
00179         QString nomRuche = "Ruche" + QString::number(i+1);
00180         settings.beginGroup(nomRuche);
00181         settings.setValue("Nom", ruches[i].nom);
00182         settings.setValue("TopicTTN", ruches[i].topicTTN);
00183         settings.setValue("MiseEnService", ruches[i].miseEnService);
00184         settings.setValue("Adresse", ruches[i].adresse);
00185         settings.setValue("Latitude", ruches[i].latitude);
00186         settings.setValue("Longitude", ruches[i].longitude);
00187         settings.endGroup();
00188     }
00189     settings.setValue("NbRuches", ruches.size());
00190 }
```

8.2.3.10 setConfigurationTTN [1/2]

```
void Configuration::setConfigurationTTN (
            ConfigurationTTN configurationTTN )  [slot]
```

Méthode pour définir une configuration TTN.

Paramètres

configurationTTN	

Définition à la ligne 36 du fichier configuration.cpp.

Références configurationTTN.

```
00037 {
00038     this->configurationTTN = configurationTTN;
00039 }
```

8.2.3.11 setConfigurationTTN [2/2]

```
void Configuration::setConfigurationTTN (
            QString hostname,
            int port,
            QString username,
            QString password )  [slot]
```

Méthode pour définir une configuration TTN.

Paramètres

hostname	
port	
username	
password	

Définition à la ligne 49 du fichier configuration.cpp.

Références configurationTTN, ConfigurationTTN : :hostname, ConfigurationTTN : :password, ConfigurationTTN : :port, et ConfigurationTTN : :username.

```
00050 {
00051     configurationTTN.hostname = hostname;
00052     configurationTTN.port = port;
00053     configurationTTN.username = username;
00054     configurationTTN.password = password;
00055     qDebug() << Q_FUNC_INFO << configurationTTN.hostname <<
      configurationTTN.port <<  configurationTTN.
      username << configurationTTN.password;
00056 }
```

8.2.3.12 setRuches

```
void Configuration::setRuches (
            QVector< Ruche > ruches )  [slot]
```

Méthode pour définir une ruche.

Paramètres

ruches	

Définition à la ligne 73 du fichier configuration.cpp.

Références ruches.

Référencé par Ihm : :ajouterNouvelleRuche(), et Ihm : :on_pushButton_supprimer_ruche_clicked().

```
00074 {
00075     this->ruches = ruches;
00076 }
```

8.2.4 Documentation des données membres

8.2.4.1 configurationTTN

```
ConfigurationTTN Configuration::configurationTTN  [private]
```

configuration MQTT pour se connecter au réseau TheThingsNetwork (TTN)

Définition à la ligne 56 du fichier configuration.h.

Référencé par chargerConfigurationTTN(), getConfigurationTTN(), sauvegarderConfigurationTTN(), et setConfigurationTTN().

8.2.4.2 ruches

```
QVector<Ruche> Configuration::ruches  [private]
```

les ruches

Définition à la ligne 57 du fichier configuration.h.

Référencé par chargerRuches(), getRuches(), sauvegarderRuches(), et setRuches().

8.2.4.3 settings

```
QSettings Configuration::settings  [private]
```

objet pour gérer un fichier .ini

Définition à la ligne 55 du fichier configuration.h.

Référencé par charger(), chargerConfigurationTTN(), chargerRuches(), getTopicRuche(), sauvegarderConfigurationTTN(), et sauvegarderRuches().

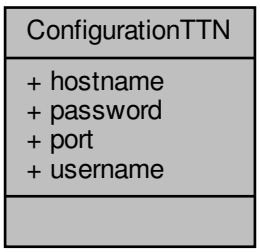
La documentation de cette classe a été générée à partir des fichiers suivants :
- configuration.h
- configuration.cpp

8.3 Référence de la structure ConfigurationTTN

Structure qui définit la configuration MQTT pour se connecter au réseau TheThingsNetwork (TTN)

```
#include <configuration.h>
```

Graphe de collaboration de ConfigurationTTN :



Attributs publics

- QString hostname
  le nom du serveur TTN
- QString password
  le mote de passe du compte TTN
- int port
  le numero de port TTN
- QString username
  le compte d'accès TTN

8.3.1 Description détaillée

Structure qui définit la configuration MQTT pour se connecter au réseau TheThingsNetwork (TTN)

Définition à la ligne 22 du fichier configuration.h.

8.3.2 Documentation des données membres

8.3.2.1 hostname

```
QString ConfigurationTTN::hostname
```

le nom du serveur TTN

Définition à la ligne 24 du fichier configuration.h.

Référencé par Ihm : :chargerConfiguration(), Configuration : :chargerConfigurationTTN(), Ihm : :demarrerTTN(), Configuration : :sauvegarderConfigurationTTN(), et Configuration : :setConfigurationTTN().

8.3.2.2 password

```
QString ConfigurationTTN::password
```

le mote de passe du compte TTN

Définition à la ligne 27 du fichier configuration.h.

Référencé par Ihm : :chargerConfiguration(), Configuration : :chargerConfigurationTTN(), Ihm : :demarrerTTN(), Configuration : :sauvegarderConfigurationTTN(), et Configuration : :setConfigurationTTN().

8.3.2.3 port

```
int ConfigurationTTN::port
```

le numero de port TTN

Définition à la ligne 25 du fichier configuration.h.

Référencé par Ihm : :chargerConfiguration(), Configuration : :chargerConfigurationTTN(), Ihm : :demarrerTTN(), Configuration : :sauvegarderConfigurationTTN(), et Configuration : :setConfigurationTTN().

8.3.2.4 username

```
QString ConfigurationTTN::username
```

le compte d'accès TTN

Définition à la ligne 26 du fichier configuration.h.

Référencé par Ihm : :chargerConfiguration(), Configuration : :chargerConfigurationTTN(), Ihm : :demarrerTTN(), Configuration : :sauvegarderConfigurationTTN(), et Configuration : :setConfigurationTTN().

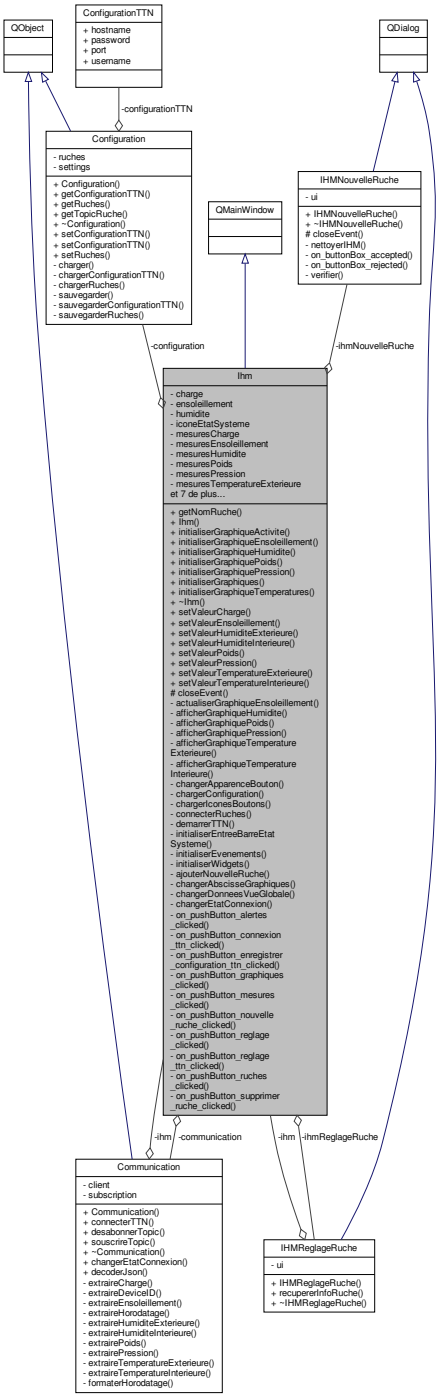
La documentation de cette structure a été générée à partir du fichier suivant :
— configuration.h

8.4 Référence de la classe Ihm

La fenêtre principale de l'application.

`#include <ihm.h>`

Graphe de collaboration de Ihm :



Connecteurs publics

— void setValeurCharge (QString nomDeLaRuche, int charge, QString horodatage)

Méthode pour définir la charge dans l'IHM.
— void setValeurEnsoleillement (QString nomDeLaRuche, int ensoleillement, QString horodatage)
Méthode pour définir l'ensoleillement dans l'IHM.
— void setValeurHumiditeExterieure (QString nomDeLaRuche, double humiditeInterieure, QString horodatage)
Méthode pour définir l'humidite dans l'IHM.
— void setValeurHumiditeInterieure (QString nomDeLaRuche, double humiditeExterieure, QString horodatage)
Méthode pour définir l'humidite dans l'IHM.
— void setValeurPoids (QString nomDeLaRuche, double poids, QString horodatage)
Méthode pour définir le poids dans l'IHM.
— void setValeurPression (QString nomDeLaRuche, int pression, QString horodatage)
Méthode pour définir la pression dans l'IHM.
— void setValeurTemperatureExterieure (QString nomDeLaRuche, double temperatureExterieure, QString horodatage)
Méthode pour définir la température extérieure dans l'IHM.
— void setValeurTemperatureInterieure (QString nomDeLaRuche, double temperatureInterieure, QString horodatage)
Méthode pour définir la température intérieure dans l'IHM.

Signaux

— void sauvegarderConfigurationTTN (QString hostname, int port, QString username, QString password)

Fonctions membres publiques

— QString getNomRuche ()
— Ihm (QWidget ∗parent=nullptr)
Constructeur de la classe Ihm.
— void initialiserGraphiqueActivite ()
— void initialiserGraphiqueEnsoleillement ()
Méthode qui initialise le graphique de l'ensoleillement.
— void initialiserGraphiqueHumidite ()
Méthode qui initialise le graphique d'humidité
— void initialiserGraphiquePoids ()
Méthode qui initialise le graphique du poids.
— void initialiserGraphiquePression ()
Méthode qui initialise le graphique de la pression.
— void initialiserGraphiques ()
Méthode qui initialise les graphiques.
— void initialiserGraphiqueTemperatures ()
Méthode qui initialise le graphique des températures.
— ∼Ihm ()
Destructeur de la classe Ihm.

Fonctions membres protégées

— void closeEvent (QCloseEvent ∗event)
Méthode redéfinie qui s'exécute pour chaque évènement reçu par la fenêtre principale.

Connecteurs privés

— void ajouterNouvelleRuche (Ruche ruche)
Méthode pour ajouter une nouvelle ruche.
— void changerAbscisseGraphiques ()
Méthode qui permet de change l'abscisse des graphiques.
— void changerDonneesVueGlobale ()
Méthode permettant de changer la données affiché sur la vue globale.
— void changerEtatConnexion (int etat)
Change l'état de connexion TTN dans l'IHM.
— void on_pushButton_alertes_clicked ()
Bouton/icône affichant l'onglet des alertes.
— void on_pushButton_connexion_ttn_clicked ()
Bouton qui permet de démarrer la connexion avec TTN.
— void on_pushButton_enregistrer_configuration_ttn_clicked ()
Bouton pour enregistrer la configuration TTN dans le fichier INI.
— void on_pushButton_graphiques_clicked ()
Bouton/icône affichant l'onglet des graphiques.

- void on_pushButton_mesures_clicked ()
  Bouton/icône affichant l'onglet des mesures.
- void on_pushButton_nouvelle_ruche_clicked ()
  Bouton qui permet d'ouvrir la fenêtre de création du nouvelle ruche.
- void on_pushButton_reglage_clicked ()
  Bouton qui permet d'ouvrir la fenêtre des réglages de la ruche.
- void on_pushButton_reglage_ttn_clicked ()
  Bouton/icône affichant l'onglet des réglages de connexion TTN.
- void on_pushButton_ruches_clicked ()
  Bouton/icône affichant l'onglet des données de la ruche selectionné
- void on_pushButton_supprimer_ruche_clicked ()
  Bouton qui permet de supprimer la ruche selectionné

Fonctions membres privées

- void actualiserGraphiqueEnsoleillement ()
  Méthode qui met à jour les mesures de l'ensoleillement dans le graphique associée.
- void afficherGraphiqueHumidite ()
  Méthode qui met à jour les mesures de l'humidité dans le graphique associée.
- void afficherGraphiquePoids ()
  Méthode qui met à jour les mesures du poids dans le graphique associée.
- void afficherGraphiquePression ()
  Méthode qui met à jour les mesures de la pression dans le graphique associée.
- void afficherGraphiqueTemperatureExterieure ()
  Méthode qui met à jour les mesures de la température extérieure dans le graphique associée.
- void afficherGraphiqueTemperatureInterieure ()
  Méthode qui met à jour les mesures de la température intérieure dans le graphique associée.
- void changerApparenceBouton (PagesIHM)
  Méthode pour changer l'apparence des bouton selon le bouton cliqué
- void chargerConfiguration ()
  Méthode pour charger la configuration TTN.
- void chargerIconesBoutons ()
  Méthode pour charger les icônes des boutons.
- void connecterRuches ()
  Méthode pour s'abonner à un topic.
- void demarrerTTN ()
  Méthode pour démarrer TTN.
- void initialiserEntreeBarreEtatSysteme ()
  Initialise l'entrée dans la barre d'état du système.
- void initialiserEvenements ()
  Assure la connexion signal/slot.
- void initialiserWidgets ()
  Initialise les différents widgets de l'IHM.

Attributs privés

- QLineSeries * charge
  La courbe de la charge.
- Communication * communication
  association vers la classe Communication
- Configuration * configuration
  association vers la classe Configuration
- QLineSeries * ensoleillement
  La courbe de l'ensoleillement.
- QLineSeries * humidite
  La courbe de l'humidite.
- QSystemTrayIcon * iconeEtatSysteme
  entrée dans la barre d'état du système
- IHMNouvelleRuche * ihmNouvelleRuche
  association vers l'IHM de création d'une nouvelle ruche
- IHMReglageRuche * ihmReglageRuche
  association vers l'IHM de réglage d'une ruche
- QVector< QPointF > mesuresCharge
  Les mesures pour la courbe de la charge.
- QVector< QPointF > mesuresEnsoleillement
  Les mesures pour la courbe de l'ensoleillement.
- QVector< QPointF > mesuresHumidite
  Les mesures pour la courbe de l'humidité.

— QVector< QPointF > mesuresPoids
Les mesures pour la courbe du poids.
— QVector< QPointF > mesuresPression
Les mesures pour la courbe de la pression.
— QVector< QPointF > mesuresTemperatureExterieure
Les mesures pour la courbe de la température extérieure.
— QVector< QPointF > mesuresTemperatureInterieure
Les mesures pour la courbe de la température intérieure.
— QLineSeries * poids
La courbe du poids.
— QLineSeries * pression
La courbe de la pression.
— QVector< Ruche > ruches
les ruches
— QLineSeries * temperatureExterieure
La courbe de la température extérieure.
— QLineSeries * temperatureInterieure
La courbe de la température intérieure.
— Ui : :ihm * ui
interface utilisateur

8.4.1 Description détaillée

La fenêtre principale de l'application.

Auteur

ACKERMANN Théo

Version

0.2

Définition à la ligne 53 du fichier ihm.h.

8.4.2 Documentation des constructeurs et destructeur

8.4.2.1 Ihm()

```
Ihm::Ihm (
            QWidget * parent = nullptr )  [explicit]
```

Constructeur de la classe Ihm.

Paramètres

parent	

Définition à la ligne 20 du fichier ihm.cpp.

Références chargerConfiguration(), chargerIconesBoutons(), demarrerTTN(), initialiserEvenements(), initialiserGraphiques(), initialiserWidgets(), et ui.

```
00020                       :QMainWindow(parent),ui(new Ui::ihm),
      ihmNouvelleRuche(new IHMNouvelleRuche),
```

```
      ihmReglageRuche(new IHMReglageRuche()),
      iconeEtatSysteme(new QSystemTrayIcon(this)), communication(new
      Communication(this)), configuration(new Configuration(this))
00021 {
00022     ui->setupUi(this);
00023     qDebug() << Q_FUNC_INFO;
00024
00025     chargerConfiguration();
00026     chargerIconesBoutons();
00027
00028     initialiserEvenements();
00029     initialiserWidgets();
00030     initialiserGraphiques();
00031
00032     demarrerTTN();
00033
00034     showMaximized();
00035 }
```

8.4.2.2 ~Ihm()

```
Ihm::~Ihm ( )
```

Destructeur de la classe Ihm.

Définition à la ligne 40 du fichier ihm.cpp.

Références ihmNouvelleRuche, ihmReglageRuche, et ui.

```
00041 {
00042     delete ihmNouvelleRuche;
00043     delete ihmReglageRuche;
00044     delete ui;
00045     qDebug() << Q_FUNC_INFO;
00046 }
```

8.4.3 Documentation des fonctions membres

8.4.3.1 actualiserGraphiqueEnsoleillement()

```
void Ihm::actualiserGraphiqueEnsoleillement ( )  [private]
```

Méthode qui met à jour les mesures de l'ensoleillement dans le graphique associée.

Définition à la ligne 886 du fichier ihm.cpp.

Références ensoleillement, et mesuresEnsoleillement.

Référencé par setValeurEnsoleillement().

```
00887 {
00888     //ensoleillement->clear();
00889     ensoleillement->setPointsVisible(true);
00890     ensoleillement->setVisible(true);
00891     ensoleillement->show();
00892     for(int i=0;i<mesuresEnsoleillement.size();i++)
00893     {
00894         ensoleillement->append(mesuresEnsoleillement[i]);
00895         qDebug() << Q_FUNC_INFO << mesuresEnsoleillement[i];
00896     }
00897
00898     qDebug() << Q_FUNC_INFO << "Nouvelle valeur sur le graphique d'ensoleillement";
00899 }
```

8.4.3.2 afficherGraphiqueHumidite()

```
void Ihm::afficherGraphiqueHumidite ( )  [private]
```

Méthode qui met à jour les mesures de l'humidité dans le graphique associée.

Définition à la ligne 876 du fichier ihm.cpp.

Références humidite, et mesuresHumidite.

```
00877 {
00878     humidite->clear();
00879     for(int i=0;i<mesuresHumidite.size();i++)
00880         humidite->append(mesuresHumidite[i]);
00881 }
```

8.4.3.3 afficherGraphiquePoids()

```
void Ihm::afficherGraphiquePoids ( )  [private]
```

Méthode qui met à jour les mesures du poids dans le graphique associée.

Définition à la ligne 914 du fichier ihm.cpp.

Références mesuresPoids, et poids.

```
00915 {
00916     poids->clear();
00917     for(int i=0;i<mesuresPoids.size();i++)
00918         poids->append(mesuresPoids[i]);
00919 }
```

8.4.3.4 afficherGraphiquePression()

```
void Ihm::afficherGraphiquePression ( )  [private]
```

Méthode qui met à jour les mesures de la pression dans le graphique associée.

Définition à la ligne 904 du fichier ihm.cpp.

Références mesuresPression, et pression.

```
00905 {
00906     pression->clear();
00907     for(int i=0;i<mesuresPression.size();i++)
00908         pression->append(mesuresPression[i]);
00909 }
```

8.4.3.5 afficherGraphiqueTemperatureExterieure()

```
void Ihm::afficherGraphiqueTemperatureExterieure ( )  [private]
```

Méthode qui met à jour les mesures de la température extérieure dans le graphique associée.

Définition à la ligne 866 du fichier ihm.cpp.

Références mesuresTemperatureExterieure, et temperatureExterieure.

```
00867 {
00868     temperatureExterieure->clear();
00869     for(int i=0;i<mesuresTemperatureExterieure.size();i++)
00870         temperatureExterieure->append(
      mesuresTemperatureExterieure[i]);
00871 }
```

8.4.3.6 afficherGraphiqueTemperatureInterieure()

```
void Ihm::afficherGraphiqueTemperatureInterieure ( )  [private]
```

Méthode qui met à jour les mesures de la température intérieure dans le graphique associée.

Définition à la ligne 856 du fichier ihm.cpp.

Références mesuresTemperatureInterieure, et temperatureInterieure.

```
00857 {
00858     temperatureInterieure->clear();
00859     for(int i=0;i<mesuresTemperatureInterieure.size();i++)
00860         temperatureInterieure->append(
      mesuresTemperatureInterieure[i]);
00861 }
```

8.4.3.7 ajouterNouvelleRuche

```
void Ihm::ajouterNouvelleRuche (
            Ruche ruche )  [private], [slot]
```

Méthode pour ajouter une nouvelle ruche.

Paramètres

ruche	

Définition à la ligne 803 du fichier ihm.cpp.

Références communication, configuration, Ruche : :nom, ruches, Configuration : :setRuches(), Communication : :souscrireTopic(), Ruche : :topicTTN, et ui.

Référencé par initialiserEvenements().

```
00804 {
```

```
00805     qDebug() << Q_FUNC_INFO << ruche.nom << ruche.topicTTN;
00806     if(ui->comboBox_liste_ruches->currentText() == "Nom de la ruche")
00807         ui->comboBox_liste_ruches->clear();
00808     ui->comboBox_liste_ruches->addItem(ruche.nom);
00809     communication->souscrireTopic(ruche.topicTTN);
00810     ruches.push_back(ruche);
00811     configuration->setRuches(ruches);
00812 }
```

8.4.3.8 changerAbscisseGraphiques

```
void Ihm::changerAbscisseGraphiques ( )  [private], [slot]
```

Méthode qui permet de change l'abscisse des graphiques.

Définition à la ligne 447 du fichier ihm.cpp.

Références ui.

Référencé par initialiserEvenements().

```
00448 {
00449     switch(ui->comboBox_reglages_graphiques->currentIndex())
00450     {
00451         case 0:
00452             qDebug() << Q_FUNC_INFO << "reponse : 1j";
00453             //axisX->setTickCount(10);
00454             //axisX->setFormat("hh:mm");
00455             //axisX->setTitleText("Heure");
00456             break;
00457         case 1:
00458             qDebug() << Q_FUNC_INFO << "reponse : 7j";
00459             //axisX->setTickCount(7);
00460             //axisX->setFormat("dd/MM");
00461             //axisX->setTitleText("Jours");
00462             break;
00463         default:
00464             qDebug() << Q_FUNC_INFO << ui->comboBox_reglages_graphiques->currentIndex();
00465      }
00466 }
```

8.4.3.9 changerApparenceBouton()

```
void Ihm::changerApparenceBouton (
            PagesIHM page )  [private]
```

Méthode pour changer l'apparence des bouton selon le bouton cliqué

Paramètres

nomBouton	

Définition à la ligne 99 du fichier ihm.cpp.

Références PAGE_ACCUEIL, PAGE_ALERTES, PAGE_GRAPHIQUES, PAGE_REGLAGES_TTN, PAGE_VUE_GLOBALE, et ui.

Référencé par on_pushButton_alertes_clicked(), on_pushButton_graphiques_clicked(), on_pushButton_mesures_clicked(), on_↩ pushButton_reglage_ttn_clicked(), et on_pushButton_ruches_clicked().

```
00100 {
00101     if(page == PagesIHM::PAGE_ACCUEIL)
00102     {
00103         ui->pushButton_ruches->setStyleSheet("background:#666666;");
00104         ui->pushButton_mesures->setStyleSheet("");
00105         ui->pushButton_graphiques->setStyleSheet("");
00106         ui->pushButton_alertes->setStyleSheet("");
00107         ui->pushButton_reglage_ttn->setStyleSheet("");
00108     }
00109     if(page == PagesIHM::PAGE_VUE_GLOBALE)
00110     {
00111         ui->pushButton_ruches->setStyleSheet("");
00112         ui->pushButton_mesures->setStyleSheet("background:#666666;");
00113         ui->pushButton_graphiques->setStyleSheet("");
00114         ui->pushButton_alertes->setStyleSheet("");
00115         ui->pushButton_reglage_ttn->setStyleSheet("");
00116     }
00117     if(page == PagesIHM::PAGE_GRAPHIQUES)
00118     {
00119         ui->pushButton_ruches->setStyleSheet("");
00120         ui->pushButton_mesures->setStyleSheet("");
00121         ui->pushButton_graphiques->setStyleSheet("background:#666666;");
00122         ui->pushButton_alertes->setStyleSheet("");
00123         ui->pushButton_reglage_ttn->setStyleSheet("");
00124     }
00125     if(page == PagesIHM::PAGE_ALERTES)
00126     {
00127         ui->pushButton_ruches->setStyleSheet("");
00128         ui->pushButton_mesures->setStyleSheet("");
00129         ui->pushButton_graphiques->setStyleSheet("");
00130         ui->pushButton_alertes->setStyleSheet("background:#666666;");
00131         ui->pushButton_reglage_ttn->setStyleSheet("");
00132     }
00133     if(page == PagesIHM::PAGE_REGLAGES_TTN)
00134     {
00135         ui->pushButton_ruches->setStyleSheet("");
00136         ui->pushButton_mesures->setStyleSheet("");
00137         ui->pushButton_graphiques->setStyleSheet("");
00138         ui->pushButton_alertes->setStyleSheet("");
00139         ui->pushButton_reglage_ttn->setStyleSheet("background:#666666;");
00140     }
00141 }
```

8.4.3.10 changerDonneesVueGlobale

```
void Ihm::changerDonneesVueGlobale ( )  [private], [slot]
```

Méthode permettant de changer la données affiché sur la vue globale.

Définition à la ligne 471 du fichier ihm.cpp.

Références ui.

Référencé par initialiserEvenements().

```
00472 {
00473     switch(ui->comboBox_donnees_affiche->currentIndex())
00474     {
00475         case 0:
00476             qDebug() << Q_FUNC_INFO << "Temperature";
00477             break;
00478         case 1:
00479             qDebug() << Q_FUNC_INFO << "Humidité";
00480             break;
00481         default:
00482             qDebug() << Q_FUNC_INFO << ui->comboBox_reglages_graphiques->currentIndex();
00483      }
00484 }
```

8.4.3.11 changerEtatConnexion

```
void Ihm::changerEtatConnexion (
            int etat )  [private], [slot]
```

Change l'état de connexion TTN dans l'IHM.

Paramètres

etat	

Définition à la ligne 491 du fichier ihm.cpp.

Références connecterRuches(), et ui.

Référencé par initialiserEvenements().

```
00492 {
00493     switch(etat)
00494     {
00495         case 0:
00496             ui->label_etat_connexion->setPixmap(QPixmap(":/off.png"));
00497             ui->pushButton_connexion_ttn->setText("Connecter");
00498             break;
00499         case 1:
00500             ui->label_etat_connexion->setPixmap(QPixmap(":/connexion.png"));
00501             break;
00502         case 2:
00503             ui->label_etat_connexion->setPixmap(QPixmap(":/on.png"));
00504             ui->pushButton_connexion_ttn->setText("Déconnecter");
00505             connecterRuches();
00506             break;
00507     }
00508 }
```

8.4.3.12 chargerConfiguration()

```
void Ihm::chargerConfiguration ( )  [private]
```

Méthode pour charger la configuration TTN.

Définition à la ligne 817 du fichier ihm.cpp.

Références configuration, Configuration : :getConfigurationTTN(), Configuration : :getRuches(), ConfigurationTTN : :hostname, ConfigurationTTN : :password, ConfigurationTTN : :port, ruches, ui, et ConfigurationTTN : :username.

Référencé par Ihm(), et on_pushButton_reglage_ttn_clicked().

```
00818 {
00819     ConfigurationTTN configurationTTN = configuration->
      getConfigurationTTN();
00820     ui->lineEdit_host->setText(configurationTTN.hostname);
00821     ui->spinBox_port->setValue(configurationTTN.port);
00822     ui->lineEdit_username->setText(configurationTTN.username);
00823     ui->lineEdit_password->setText(configurationTTN.password);
00824
00825     ruches = configuration->getRuches();
00826 }
```

8.4.3.13 chargerIconesBoutons()

```
void Ihm::chargerIconesBoutons ( )  [private]
```

Méthode pour charger les icônes des boutons.

Définition à la ligne 528 du fichier ihm.cpp.

Références ui.

Référencé par Ihm().

```
00529 {
00530     ui->pushButton_ruches->setIcon(QIcon(":/ruche.png"));
00531     ui->pushButton_graphiques->setIcon(QIcon(":/graphiques.png"));
00532     ui->pushButton_alertes->setIcon(QIcon(":/alertes.png"));
00533     ui->pushButton_reglage_ttn->setIcon(QIcon(":/reglages.png"));
00534     ui->pushButton_mesures->setIcon(QIcon(":/vue_globale.png"));
00535 }
```

8.4.3.14 closeEvent()

```
void Ihm::closeEvent (
            QCloseEvent * event )  [protected]
```

Méthode redéfinie qui s'exécute pour chaque évènement reçu par la fenêtre principale.

Paramètres

event	

Définition à la ligne 515 du fichier ihm.cpp.

Références iconeEtatSysteme, et NOM_APPLICATION.

```
00516 {
00517     if (iconeEtatSysteme->isVisible())
00518     {
00519         QMessageBox::information(this, NOM_APPLICATION, "Utiliser le menu Quitter pour
      mettre fin à l'application.");
00520         hide();
00521         event->ignore();
00522     }
00523 }
```

8.4.3.15 connecterRuches()

```
void Ihm::connecterRuches ( )  [private]
```

Méthode pour s'abonner à un topic.

Définition à la ligne 840 du fichier ihm.cpp.

Références communication, ruches, Communication : :souscrireTopic(), et ui.

Référencé par changerEtatConnexion().

```
00841 {
00842     qDebug() << Q_FUNC_INFO << ruches.size();
00843     if(ruches.size() > 0)
00844         ui->comboBox_liste_ruches->clear();
00845     for(int i = 0; i < ruches.size(); i++)
00846     {
00847         qDebug() << Q_FUNC_INFO << ruches[i].nom << ruches[i].topicTTN;
00848         communication->souscrireTopic(ruches[i].topicTTN);
00849         ui->comboBox_liste_ruches->addItem(ruches[i].nom);
00850     }
00851 }
```

8.4.3.16 demarrerTTN()

```
void Ihm::demarrerTTN ( )  [private]
```

Méthode pour démarrer TTN.

Définition à la ligne 831 du fichier ihm.cpp.

Références communication, configuration, Communication : :connecterTTN(), Configuration : :getConfigurationTTN(), Configuration↩ TTN : :hostname, ConfigurationTTN : :password, ConfigurationTTN : :port, et ConfigurationTTN : :username.

Référencé par Ihm().

```
00832 {
00833     ConfigurationTTN configurationTTN = configuration->
      getConfigurationTTN();
00834     communication->connecterTTN(configurationTTN.
      hostname, configurationTTN.port, configurationTTN.username, configurationTTN.
      password);
00835 }
```

8.4.3.17 getNomRuche()

```
QString Ihm::getNomRuche ( )
```

Définition à la ligne 921 du fichier ihm.cpp.

Références ui.

```
00922 {
00923     return ui->comboBox_liste_ruches->currentText();
00924 }
```

8.4.3.18 initialiserEntreeBarreEtatSysteme()

```
void Ihm::initialiserEntreeBarreEtatSysteme ( )  [private]
```

Initialise l'entrée dans la barre d'état du système.

Définition à la ligne 587 du fichier ihm.cpp.

Références iconeEtatSysteme, et NOM_APPLICATION.

```
00588 {
00589     // Crée les actions pour l'application
00590     QAction *actionMinimiser = new QAction(QString::fromUtf8("Minimiser"), this);
00591     QAction *actionMaximiser = new QAction(QString::fromUtf8("Maximiser"), this);
00592     QAction *actionRestaurer = new QAction(QString::fromUtf8("Restaurer"), this);
00593     QAction *actionQuitter = new QAction(QString::fromUtf8("&Quitter"), this);
00594
00595     // Connecte les actions
00596     connect(actionMinimiser, SIGNAL(triggered(bool)), this, SLOT(hide()));
00597     connect(actionMaximiser, SIGNAL(triggered(bool)), this, SLOT(showMaximized()));
00598     connect(actionRestaurer, SIGNAL(triggered(bool)), this, SLOT(showNormal()));
00599     connect(actionQuitter, SIGNAL(triggered(bool)), qApp, SLOT(quit()));
00600
00601     // Crée le menu pour l'entrée dans la barre d'état système
00602     QMenu * menuEtatSysteme = new QMenu(this);
00603     menuEtatSysteme->addAction(actionMinimiser);
00604     menuEtatSysteme->addAction(actionMaximiser);
00605     menuEtatSysteme->addAction(actionRestaurer);
00606     menuEtatSysteme->addSeparator();
00607     menuEtatSysteme->addAction(actionQuitter);
00608
00609     // Crée l'entrée pour la barre d'état système
00610     iconeEtatSysteme->setContextMenu(menuEtatSysteme);
00611     iconeEtatSysteme->setToolTip(NOM_APPLICATION);
00612     // Crée l'icône pour la barre d'état système
00613     QIcon iconeRuche(":/ruche.png");
00614     iconeEtatSysteme->setIcon(iconeRuche);
00615     setWindowIcon(iconeRuche);
00616
00617     iconeEtatSysteme->show();
00618 }
```

8.4.3.19 initialiserEvenements()

```
void Ihm::initialiserEvenements ( )  [private]
```

Assure la connexion signal/slot.

Définition à la ligne 557 du fichier ihm.cpp.

Références ajouterNouvelleRuche(), changerAbscisseGraphiques(), changerDonneesVueGlobale(), changerEtatConnexion(), communication, configuration, ihmNouvelleRuche, sauvegarderConfigurationTTN(), setValeurCharge(), setValeurEnsoleillement(), setValeurHumiditeExterieure(), setValeurHumiditeInterieure(), setValeurPoids(), setValeurPression(), setValeurTemperatureExterieure(), setValeurTemperatureInterieure(), et ui.

Référencé par Ihm().

```
00558 {
00559     connect(ui->comboBox_reglages_graphiques, SIGNAL(currentIndexChanged(int)), SLOT(
      changerAbscisseGraphiques()));
00560     connect(ui->comboBox_donnees_affiche, SIGNAL(currentIndexChanged(int)), SLOT(
      changerDonneesVueGlobale()));
00561
00562     // Configuration
00563     connect(this, SIGNAL(sauvegarderConfigurationTTN(QString,int,QString,QString
      )), configuration, SLOT(setConfigurationTTN(QString,int,QString,QString)));
00564
00565     // Ajouter une nouvelle ruche
00566     connect(ihmNouvelleRuche, SIGNAL(nouvelleRuche(Ruche)), this, SLOT(
      ajouterNouvelleRuche(Ruche)));
00567
00568     // Afficher valeur reçue TTN
00569     connect(communication, SIGNAL(nouvelleValeurTemperatureInterieure(QString,double,QString))
      , this, SLOT(setValeurTemperatureInterieure(QString,double,QString)));
00570     connect(communication, SIGNAL(nouvelleValeurTemperatureExterieure(QString,double,QString))
      , this, SLOT(setValeurTemperatureExterieure(QString,double,QString)));
00571
00572     connect(communication, SIGNAL(nouvelleValeurHumiditeInterieure(QString,double,QString)),
      this, SLOT(setValeurHumiditeInterieure(QString,double,QString)));
00573     connect(communication, SIGNAL(nouvelleValeurHumiditeExterieure(QString,double,QString)),
      this, SLOT(setValeurHumiditeExterieure(QString,double,QString)));
00574
00575     connect(communication, SIGNAL(nouvelleValeurEnsoleillement(QString,int,QString)), this,
      SLOT(setValeurEnsoleillement(QString,int,QString)));
00576     connect(communication, SIGNAL(nouvelleValeurPression(QString,int,QString)), this, SLOT(
      setValeurPression(QString,int,QString)));
00577     connect(communication, SIGNAL(nouvelleValeurPoids(QString,double,QString)), this, SLOT(
      setValeurPoids(QString,double,QString)));
00578     connect(communication, SIGNAL(nouvelleValeurCharge(QString,int,QString)), this, SLOT(
      setValeurCharge(QString,int,QString)));
00579
00580     // Communication
00581     connect(communication, SIGNAL(nouvelEtatConnexion(int)), this, SLOT(
      changerEtatConnexion(int)));
00582 }
```

8.4.3.20 initialiserGraphiqueActivite()

```
void Ihm::initialiserGraphiqueActivite ( )
```

8.4.3.21 initialiserGraphiqueEnsoleillement()

```
void Ihm::initialiserGraphiqueEnsoleillement ( )
```

Méthode qui initialise le graphique de l'ensoleillement.

Définition à la ligne 321 du fichier ihm.cpp.

Références ensoleillement, et ui.

Référencé par initialiserGraphiques().

```
00322 {
00323     ensoleillement = new QLineSeries();
00324
00325     QChart *chart = new QChart();
00326     chart->legend()->hide();
00327     chart->addSeries(ensoleillement);
00328     chart->setTitle("Ensoleillement");
00329     ensoleillement->setPointsVisible(true);
00330     ui->chartView_ensoleillement->setChart(chart);
00331     ui->chartView_ensoleillement->setRenderHint(QPainter::Antialiasing);
00332
00333     QDateTimeAxis *axisX = new QDateTimeAxis();
00334     axisX->setTickCount(24);
00335     axisX->setFormat("hh");
00336     axisX->setTitleText("Heure");
00337     QDateTime qdatetime;
00338     qint64 msDepuisEpochMin = qdatetime.currentSecsSinceEpoch() - 43200;
00339     qint64 msDepuisEpochMax = qdatetime.currentSecsSinceEpoch() + 43200;
00340
00341     qDebug() << msDepuisEpochMin << msDepuisEpochMax;
00342
00343     //axisX->setMin(qdatetime.fromSecsSinceEpoch(msDepuisEpochMin));
00344     //axisX->setMax(qdatetime.fromSecsSinceEpoch(msDepuisEpochMax));
00345
00346     QValueAxis *axisY = new QValueAxis();
00347     axisY->setTitleText("lux");
00348     axisY->setMin(0);
00349     axisY->setMax(500);
00350
00351     ensoleillement->attachAxis(axisX);
00352     ensoleillement->attachAxis(axisY);
00353
00354     chart->addAxis(axisX, Qt::AlignBottom);
00355     chart->addAxis(axisY, Qt::AlignLeft);
00356     //chart->setAxisY(axisY);
00357     //chart->setAxisX(axisX);
00358
00359     chart->show();
00360 }
```

8.4.3.22 initialiserGraphiqueHumidite()

```
void Ihm::initialiserGraphiqueHumidite ( )
```

Méthode qui initialise le graphique d'humidité

Définition à la ligne 280 du fichier ihm.cpp.

Références AXE_TEMPERATURE_MAX, AXE_TEMPERATURE_MIN, humidite, et ui.

Référencé par initialiserGraphiques().

```
00281 {
00282     humidite = new QLineSeries();
00283     /* Valeurs de test

00284     humidite->append(0, 27);

00285     humidite->append(1, 26);

00286     humidite->append(2, 28);

00287     humidite->append(3, 31);

00288     humidite->append(4, 24);*/
00289
00290     QChart *chart = new QChart();
00291     chart->legend()->hide();
00292     chart->addSeries(humidite);
00293     chart->setTitle("Humidité");
00294     ui->chartView_humidite->setChart(chart);
00295     ui->chartView_humidite->setRenderHint(QPainter::Antialiasing);
00296
00297     QDateTimeAxis *axisX = new QDateTimeAxis();
00298     axisX->setTickCount(7);
00299     axisX->setFormat("dd/MM");
00300     axisX->setTitleText("Jours");
00301     // ou :
00302     //axisX->setTickCount(10);
00303     //axisX->setFormat("hh:mm");
00304     //axisX->setTitleText("Heure");
00305     axisX->setMin(QDateTime::currentDateTime().addDays(-3));
00306     axisX->setMax(QDateTime::currentDateTime().addDays(3));
00307
00308     QValueAxis *axisY = new QValueAxis();
00309     axisY->setTitleText("%");
00310     axisY->setTickCount((((AXE_TEMPERATURE_MAX - (
      AXE_TEMPERATURE_MIN))*2)/10)+1);
00311     axisY->setMin(AXE_TEMPERATURE_MIN);
00312     axisY->setMax(AXE_TEMPERATURE_MAX);
00313
00314     chart->setAxisY(axisY);
00315     chart->setAxisX(axisX);
00316 }
```

8.4.3.23 initialiserGraphiquePoids()

```
void Ihm::initialiserGraphiquePoids ( )
```

Méthode qui initialise le graphique du poids.

Définition à la ligne 406 du fichier ihm.cpp.

Références poids, et ui.

Référencé par initialiserGraphiques().

```
00407 {
00408     poids = new QLineSeries();
00409     // Valeurs de test
00410     poids->append(0, 321);
00411     poids->append(1, 354);
00412     poids->append(2, 396);
00413     poids->append(3, 348);
00414     poids->append(4, 240);
00415
00416     QChart *chart = new QChart();
00417     chart->legend()->hide();
00418     chart->addSeries(poids);
00419     chart->setTitle("Poids");
00420     ui->chartView_poids->setChart(chart);
00421     ui->chartView_poids->setRenderHint(QPainter::Antialiasing);
00422
00423     QDateTimeAxis *axisX = new QDateTimeAxis();
00424     axisX->setTickCount(7);
00425     axisX->setFormat("dd/MM");
00426     axisX->setTitleText("Jours");
00427     // ou :
00428     //axisX->setTickCount(10);
00429     //axisX->setFormat("hh:mm");
```

```
00430     //axisX->setTitleText("Heure");
00431
00432     axisX->setMin(QDateTime::currentDateTime().addDays(-3));
00433     axisX->setMax(QDateTime::currentDateTime().addDays(3));
00434
00435     QValueAxis *axisY = new QValueAxis();
00436     axisY->setTitleText("Kg");
00437     axisY->setMin(0);
00438     axisY->setMax(500);
00439
00440     chart->setAxisY(axisY);
00441     chart->setAxisX(axisX);
00442 }
```

8.4.3.24 initialiserGraphiquePression()

```
void Ihm::initialiserGraphiquePression ( )
```

Méthode qui initialise le graphique de la pression.

Définition à la ligne 365 du fichier ihm.cpp.

Références pression, et ui.

Référencé par initialiserGraphiques().

```
00366 {
00367     pression = new QLineSeries();
00368     // Valeurs de test
00369     pression->append(0, 321);
00370     pression->append(1, 354);
00371     pression->append(2, 396);
00372     pression->append(3, 348);
00373     pression->append(4, 240);
00374
00375     QChart *chart = new QChart();
00376     chart->legend()->hide();
00377     chart->addSeries(pression);
00378     chart->setTitle("Pression");
00379     ui->chartView_pression->setChart(chart);
00380     ui->chartView_pression->setRenderHint(QPainter::Antialiasing);
00381
00382     QDateTimeAxis *axisX = new QDateTimeAxis();
00383     axisX->setTickCount(7);
00384     axisX->setFormat("dd/MM");
00385     axisX->setTitleText("Jours");
00386     // ou :
00387     //axisX->setTickCount(10);
00388     //axisX->setFormat("hh:mm");
00389     //axisX->setTitleText("Heure");
00390
00391     axisX->setMin(QDateTime::currentDateTime().addDays(-3));
00392     axisX->setMax(QDateTime::currentDateTime().addDays(3));
00393
00394     QValueAxis *axisY = new QValueAxis();
00395     axisY->setTitleText("hPa");
00396     axisY->setMin(0);
00397     axisY->setMax(500);
00398
00399     chart->setAxisY(axisY);
00400     chart->setAxisX(axisX);
00401 }
```

8.4.3.25 initialiserGraphiques()

```
void Ihm::initialiserGraphiques ( )
```

Méthode qui initialise les graphiques.

Définition à la ligne 211 du fichier ihm.cpp.

Références initialiserGraphiqueEnsoleillement(), initialiserGraphiqueHumidite(), initialiserGraphiquePoids(), initialiserGraphique↩Pression(), et initialiserGraphiqueTemperatures().

Référencé par Ihm().

```
00212 {
00213     initialiserGraphiqueTemperatures();
00214     initialiserGraphiqueHumidite();
00215     initialiserGraphiqueEnsoleillement();
00216     initialiserGraphiquePression();
00217     initialiserGraphiquePoids();
00218     //initialiserGraphiqueActivite();
00219
00220     qDebug() << Q_FUNC_INFO;
00221 }
```

8.4.3.26 initialiserGraphiqueTemperatures()

```
void Ihm::initialiserGraphiqueTemperatures ( )
```

Méthode qui initialise le graphique des températures.

A faire Faire un axeX commun pour tout les graphiques

Définition à la ligne 226 du fichier ihm.cpp.

Références AXE_TEMPERATURE_MAX, AXE_TEMPERATURE_MIN, temperatureExterieure, temperatureInterieure, et ui.

Référencé par initialiserGraphiques().

```
00227 {
00228     temperatureInterieure = new QLineSeries();
00229     temperatureInterieure->setName("Intérieure");
00230     temperatureInterieure->setPointsVisible(true);
00231
00232     temperatureExterieure = new QLineSeries();
00233     temperatureExterieure->setName("Extérieure");
00234     temperatureExterieure->setPointsVisible(true);
00235
00236     /* Valeurs de test

00237     temperatureExterieure->append(0, 35);

00238     temperatureExterieure->append(1, 37);

00239     temperatureExterieure->append(2, 35);

00240     temperatureExterieure->append(3, 34);

00241     temperatureExterieure->append(4, 31);*/
00242
00243     QChart *chart = new QChart();
00244     chart->legend()->show();
00245     chart->addSeries(temperatureInterieure);
00246     chart->addSeries(temperatureExterieure);
00247
00248     chart->createDefaultAxes();
00249     chart->setTitle("Températures");
00250     ui->chartView_temperature->setChart(chart);
00251     ui->chartView_temperature->setRenderHint(QPainter::Antialiasing);
00252
```

```
00256     QDateTimeAxis *axisX = new QDateTimeAxis();
00257     axisX->setTickCount(7);
00258     axisX->setFormat("dd/MM");
00259     axisX->setTitleText("Jours");
00260     // ou :
00261     //axisX->setTickCount(10);
00262     //axisX->setFormat("hh:mm");
00263     //axisX->setTitleText("Heure");
00264     axisX->setMin(QDateTime::currentDateTime().addDays(-3));
00265     axisX->setMax(QDateTime::currentDateTime().addDays(3));
00266
00267     QValueAxis *axisY = new QValueAxis();
00268     axisY->setTitleText("°C");
00269     axisY->setTickCount((((AXE_TEMPERATURE_MAX - (
      AXE_TEMPERATURE_MIN))*2)/10)+1);
00270     axisY->setMin(AXE_TEMPERATURE_MIN);
00271     axisY->setMax(AXE_TEMPERATURE_MAX);
00272
00273     chart->setAxisY(axisY);
00274     chart->setAxisX(axisX);
00275 }
```

8.4.3.27 initialiserWidgets()

`void Ihm::initialiserWidgets ( )  [private]`

Initialise les différents widgets de l'IHM.

Définition à la ligne 540 du fichier ihm.cpp.

Références NOM_APPLICATION, ui, et VERSION_APPLICATION.

Référencé par Ihm().

```
00541 {
00542     this->setWindowTitle(NOM_APPLICATION);
00543     ui->label_version->setText(VERSION_APPLICATION);
00544
00545     ui->comboBox_liste_ruches->addItem("Nom de la ruche");
00546
00547     ui->comboBox_reglages_graphiques->addItem("1 jour");
00548     ui->comboBox_reglages_graphiques->addItem("7 jours");
00549
00550     ui->comboBox_donnees_affiche->addItem("Température");
00551     ui->comboBox_donnees_affiche->addItem("Humidité");
00552 }
```

8.4.3.28 on_pushButton_alertes_clicked

`void Ihm::on_pushButton_alertes_clicked ( )  [private], [slot]`

Bouton/icône affichant l'onglet des alertes.

Définition à la ligne 78 du fichier ihm.cpp.

Références changerApparenceBouton(), PAGE_ALERTES, et ui.

```
00079 {
00080     ui->stackedWidget->setCurrentIndex(PagesIHM::PAGE_ALERTES);
00081     changerApparenceBouton(PagesIHM::PAGE_ALERTES);
00082 }
```

8.4.3.29 on_pushButton_connexion_ttn_clicked

```
void Ihm::on_pushButton_connexion_ttn_clicked ( )  [private], [slot]
```

Bouton qui permet de démarrer la connexion avec TTN.

Définition à la ligne 146 du fichier ihm.cpp.

Références communication, Communication : :connecterTTN(), et ui.

```
00147 {
00148     communication->connecterTTN(ui->lineEdit_host->text(),
      ui->spinBox_port->value(), ui->lineEdit_username->text(), ui->lineEdit_password->text());
00149 }
```

8.4.3.30 on_pushButton_enregistrer_configuration_ttn_clicked

```
void Ihm::on_pushButton_enregistrer_configuration_ttn_clicked ( )  [private], [slot]
```

Bouton pour enregistrer la configuration TTN dans le fichier INI.

Définition à la ligne 793 du fichier ihm.cpp.

Références sauvegarderConfigurationTTN(), et ui.

```
00794 {
00795     emit sauvegarderConfigurationTTN(ui->lineEdit_host->text(),
      ui->spinBox_port->value(), ui->lineEdit_username->text(), ui->lineEdit_password->text());
00796 }
```

8.4.3.31 on_pushButton_graphiques_clicked

```
void Ihm::on_pushButton_graphiques_clicked ( )  [private], [slot]
```

Bouton/icône affichant l'onglet des graphiques.

Définition à la ligne 69 du fichier ihm.cpp.

Références changerApparenceBouton(), PAGE_GRAPHIQUES, et ui.

```
00070 {
00071     ui->stackedWidget->setCurrentIndex(PagesIHM::PAGE_GRAPHIQUES);
00072     changerApparenceBouton(PagesIHM::PAGE_GRAPHIQUES);
00073 }
```

8.4.3.32 on_pushButton_mesures_clicked

```
void Ihm::on_pushButton_mesures_clicked ( )  [private], [slot]
```

Bouton/icône affichant l'onglet des mesures.

Définition à la ligne 60 du fichier ihm.cpp.

Références changerApparenceBouton(), PAGE_VUE_GLOBALE, et ui.

```
00061 {
00062     ui->stackedWidget->setCurrentIndex(PagesIHM::PAGE_VUE_GLOBALE);
00063     changerApparenceBouton(PagesIHM::PAGE_VUE_GLOBALE);
00064 }
```

8.4.3.33 on_pushButton_nouvelle_ruche_clicked

```
void Ihm::on_pushButton_nouvelle_ruche_clicked ( )  [private], [slot]
```

Bouton qui permet d'ouvrir la fenêtre de création du nouvelle ruche.

Définition à la ligne 154 du fichier ihm.cpp.

Références ihmNouvelleRuche.

```
00155 {
00156     ihmNouvelleRuche->exec();
00157 }
```

8.4.3.34 on_pushButton_reglage_clicked

```
void Ihm::on_pushButton_reglage_clicked ( )  [private], [slot]
```

Bouton qui permet d'ouvrir la fenêtre des réglages de la ruche.

Définition à la ligne 162 du fichier ihm.cpp.

Références ihmReglageRuche, IHMReglageRuche : :recupererInfoRuche(), et ui.

```
00163 {
00164     if(ui->comboBox_liste_ruches->currentText() == nullptr)
00165     {
00166         QMessageBox::warning(this,"Erreur","Il n'y a pas de ruche.");
00167         qDebug() << Q_FUNC_INFO << "Il n'y a pas de ruche.";
00168     }
00169     else
00170     {
00171         ihmReglageRuche->recupererInfoRuche(ui->comboBox_liste_ruches->
      currentText());
00172         ihmReglageRuche->exec();
00173     }
00174 }
```

8.4.3.35 on_pushButton_reglage_ttn_clicked

```
void Ihm::on_pushButton_reglage_ttn_clicked ( )  [private], [slot]
```

Bouton/icône affichant l'onglet des réglages de connexion TTN.

Définition à la ligne 87 du fichier ihm.cpp.

Références changerApparenceBouton(), chargerConfiguration(), PAGE_REGLAGES_TTN, et ui.

```
00088 {
00089     chargerConfiguration();
00090     ui->stackedWidget->setCurrentIndex(PagesIHM::PAGE_REGLAGES_TTN);
00091     changerApparenceBouton(PagesIHM::PAGE_REGLAGES_TTN);
00092 }
```

8.4.3.36 on_pushButton_ruches_clicked

```
void Ihm::on_pushButton_ruches_clicked ( )  [private], [slot]
```

Bouton/icône affichant l'onglet des données de la ruche selectionné

Définition à la ligne 51 du fichier ihm.cpp.

Références changerApparenceBouton(), PAGE_ACCUEIL, et ui.

```
00052 {
00053     ui->stackedWidget->setCurrentIndex(PagesIHM::PAGE_ACCUEIL);
00054     changerApparenceBouton(PagesIHM::PAGE_ACCUEIL);
00055 }
```

8.4.3.37 on_pushButton_supprimer_ruche_clicked

```
void Ihm::on_pushButton_supprimer_ruche_clicked ( )  [private], [slot]
```

Bouton qui permet de supprimer la ruche selectionné

Définition à la ligne 179 du fichier ihm.cpp.

Références communication, configuration, Communication : :desabonnerTopic(), ruches, Configuration : :setRuches(), et ui.

```
00180 {
00181     if(ui->comboBox_liste_ruches->currentText() == nullptr | ui->comboBox_liste_ruches->currentText() =
      = "Nom de la ruche")
00182     {
00183         QMessageBox::warning(this,"Erreur","Il n'y a pas de ruche.");
00184         qDebug() << Q_FUNC_INFO << "Il n'y a pas de ruche.";
00185     }
00186     else
00187     {
00188         QMessageBox::StandardButton reponse;
00189         QString nom_ruche = ui->comboBox_liste_ruches->currentText();
00190         QString question = "Êtes-vous sûr de vouloir supprimer la ruche '" + nom_ruche + "' ?";
00191         reponse = QMessageBox::question(this,"",question,QMessageBox::Yes|QMessageBox::No);
00192
00193         if(reponse == QMessageBox::Yes)
00194         {
00195             qDebug() << Q_FUNC_INFO << "reponse : Oui";
00196             communication->desabonnerTopic(ruches[
      ui->comboBox_liste_ruches->currentIndex()].topicTTN);
00197             ruches.remove(ui->comboBox_liste_ruches->currentIndex());
00198             configuration->setRuches(ruches);
00199             ui->comboBox_liste_ruches->removeItem(ui->comboBox_liste_ruches->currentIndex());
00200         }
00201         else
00202         {
00203             qDebug() << Q_FUNC_INFO << "reponse : Non";
00204         }
00205     }
00206 }
```

8.4.3.38 sauvegarderConfigurationTTN

```
void Ihm::sauvegarderConfigurationTTN (
            QString hostname,
            int port,
            QString username,
            QString password )  [signal]
```

Référencé par initialiserEvenements(), et on_pushButton_enregistrer_configuration_ttn_clicked().

8.4.3.39 setValeurCharge

```
void Ihm::setValeurCharge (
            QString nomDeLaRuche,
            int charge,
            QString horodatage )  [slot]
```

Méthode pour définir la charge dans l'IHM.

Paramètres

nomDeLaRuche	
charge	
horodatage	

Définition à la ligne 777 du fichier ihm.cpp.

Références charge, mesuresCharge, ruches, et ui.

Référencé par initialiserEvenements().

```
00778 {
00779     if(ruches[ui->comboBox_liste_ruches->currentIndex()].topicTTN.contains(nomDeLaRuche))
00780     {
00781         ui->lcdNumber_charge->display(charge);
00782         QString temps = horodatage;
00783         ui->label_maj_charge->setText(temps);
00784     }
00785     QPointF mesure(mesuresCharge.size(), charge);
00786     mesuresCharge.push_back(mesure);
00787     qDebug() << Q_FUNC_INFO << nomDeLaRuche << "Nouvelle charge :" << charge;
00788 }
```

8.4.3.40 setValeurEnsoleillement

```
void Ihm::setValeurEnsoleillement (
            QString nomDeLaRuche,
            int ensoleillement,
            QString horodatage )  [slot]
```

Méthode pour définir l'ensoleillement dans l'IHM.

Paramètres

nomDeLaRuche	
ensoleillement	
horodatage	

Définition à la ligne 709 du fichier ihm.cpp.

Références actualiserGraphiqueEnsoleillement(), ensoleillement, mesuresEnsoleillement, ruches, et ui.

Référencé par initialiserEvenements().

```
00710 {
00711     if(ruches[ui->comboBox_liste_ruches->currentIndex()].topicTTN.contains(nomDeLaRuche))
00712     {
00713         ui->lcdNumber_ensoleillement->display(ensoleillement);
00714         QString temps = horodatage;
00715         ui->label_maj_ensoleillement->setText(temps);
00716     }
00717     QDateTime qdatetime;
00718     qint64 msDepuisEpoch;
00719     msDepuisEpoch = qdatetime.currentSecsSinceEpoch();
00720     QPointF mesure(msDepuisEpoch, ensoleillement);
00721     mesuresEnsoleillement.push_back(mesure);
00722     qDebug() << Q_FUNC_INFO << nomDeLaRuche << "Nouvelle ensoleillement :" <<
      ensoleillement;
00723     qDebug() << mesuresEnsoleillement.size();
00724     for(int i=0;i<mesuresEnsoleillement.size();i++)
00725         qDebug() << mesuresEnsoleillement[i];
00726     actualiserGraphiqueEnsoleillement();
00727 }
```

8.4.3.41 setValeurHumiditeExterieure

```
void Ihm::setValeurHumiditeExterieure (
             QString nomDeLaRuche,
             double humidite,
             QString horodatage )  [slot]
```

Méthode pour définir l'humidite dans l'IHM.

Paramètres

nomDeLaRuche	
humidite	
horodatage	

Définition à la ligne 689 du fichier ihm.cpp.

Références humidite, mesuresHumidite, ruches, et ui.

Référencé par initialiserEvenements().

```
00690 {
00691     if(ruches[ui->comboBox_liste_ruches->currentIndex()].topicTTN.contains(nomDeLaRuche))
00692     {
00693         ui->lcdNumber_humidite_exterieure->display(humidite);
00694         QString temps = horodatage;
00695         ui->label_maj_humidite_exterieure->setText(temps);
00696     }
00697     QPointF mesure(mesuresHumidite.size(), humidite);
00698     mesuresHumidite.push_back(mesure);
00699     qDebug() << Q_FUNC_INFO << nomDeLaRuche << "Nouvelle humidité extérieure:" <<
      humidite;
00700 }
```

8.4.3.42 setValeurHumiditeInterieure

```
void Ihm::setValeurHumiditeInterieure (
            QString nomDeLaRuche,
            double humidite,
            QString horodatage )  [slot]
```

Méthode pour définir l'humidite dans l'IHM.

Paramètres

nomDeLaRuche	
humidite	
horodatage	

Définition à la ligne 669 du fichier ihm.cpp.

Références humidite, mesuresHumidite, ruches, et ui.

Référencé par initialiserEvenements().

```
00670 {
00671     if(ruches[ui->comboBox_liste_ruches->currentIndex()].topicTTN.contains(nomDeLaRuche))
00672     {
00673         ui->lcdNumber_humidite_interieure->display(humidite);
00674         QString temps = horodatage;
00675         ui->label_maj_humidite_interieure->setText(temps);
00676     }
00677     QPointF mesure(mesuresHumidite.size(), humidite);
00678     mesuresHumidite.push_back(mesure);
00679     qDebug() << Q_FUNC_INFO << nomDeLaRuche << "Nouvelle humidité intérieure:" <<
      humidite;
00680 }
```

8.4.3.43 setValeurPoids

```
void Ihm::setValeurPoids (
            QString nomDeLaRuche,
            double poids,
            QString horodatage )  [slot]
```

Méthode pour définir le poids dans l'IHM.

Paramètres

nomDeLaRuche	
poids	
horodatage	

Définition à la ligne 756 du fichier ihm.cpp.

Références mesuresPoids, poids, ruches, et ui.

Référencé par initialiserEvenements().

```
00757 {
00758     //poids = poids*0.001; // valeur à un dixième
00759     if(ruches[ui->comboBox_liste_ruches->currentIndex()].topicTTN.contains(nomDeLaRuche))
00760     {
00761         ui->lcdNumber_poids->display(poids);
00762         QString temps = horodatage;
00763         ui->label_maj_poids->setText(temps);
00764     }
00765     QPointF mesure(mesuresPoids.size(), poids);
00766     mesuresPoids.push_back(mesure);
00767     qDebug() << Q_FUNC_INFO << nomDeLaRuche << "Nouveau poids :" << poids;
00768 }
```

8.4.3.44 setValeurPression

```
void Ihm::setValeurPression (
            QString nomDeLaRuche,
            int pression,
            QString horodatage )  [slot]
```

Méthode pour définir la pression dans l'IHM.

Paramètres

nomDeLaRuche	
pression	
horodatage	

Définition à la ligne 736 du fichier ihm.cpp.

Références mesuresPression, pression, ruches, et ui.

Référencé par initialiserEvenements().

```
00737 {
00738     if(ruches[ui->comboBox_liste_ruches->currentIndex()].topicTTN.contains(nomDeLaRuche))
00739     {
00740         ui->lcdNumber_pression->display(pression);
00741         QString temps = horodatage;
00742         ui->label_maj_pression->setText(temps);
00743     }
00744     QPointF mesure(mesuresPression.size(), pression);
00745     mesuresPression.push_back(mesure);
00746     qDebug() << Q_FUNC_INFO << nomDeLaRuche << "Nouvelle pression :" << pression;
00747 }
```

8.4.3.45 setValeurTemperatureExterieure

```
void Ihm::setValeurTemperatureExterieure (
            QString nomDeLaRuche,
            double temperatureExterieure,
            QString horodatage )  [slot]
```

Méthode pour définir la température extérieure dans l'IHM.

Paramètres

nomDeLaRuche	
temperatureInterieure	
horodatage	

Définition à la ligne 648 du fichier ihm.cpp.

Références mesuresTemperatureExterieure, ruches, temperatureExterieure, et ui.

Référencé par initialiserEvenements().

```
00649 {
00650     if(ruches[ui->comboBox_liste_ruches->currentIndex()].topicTTN.contains(nomDeLaRuche))
00651     {
00652         ui->lcdNumber_temperature_exterieure->display(temperatureExterieure);
00653         QString temps = horodatage;
00654         ui->label_maj_temp_ext->setText(temps);
00655     }
00656     QPointF mesure(mesuresTemperatureExterieure.size(),
      temperatureExterieure);
00657     mesuresTemperatureExterieure.push_back(mesure);
00658     qDebug() << Q_FUNC_INFO << nomDeLaRuche << "Nouvelle température extérieure :" <<
      temperatureExterieure;
00659 }
```

8.4.3.46 setValeurTemperatureInterieure

```
void Ihm::setValeurTemperatureInterieure (
            QString nomDeLaRuche,
            double temperatureInterieure,
            QString horodatage )  [slot]
```

Méthode pour définir la température intérieure dans l'IHM.

Paramètres

temperatureInterieure	

A faire Intégrer l'horodatage pour la mesure (cf. toMSecsSinceEpoch() de la classe QDateTime) pour l'instant l'axe X est une valeur 0,1, ...

Définition à la ligne 625 du fichier ihm.cpp.

Références mesuresTemperatureInterieure, ruches, temperatureInterieure, et ui.

Référencé par initialiserEvenements().

```
00626 {
00627     if(ruches[ui->comboBox_liste_ruches->currentIndex()].topicTTN.contains(nomDeLaRuche))
00628     {
00629         ui->lcdNumber_temperature_interieure->display(temperatureInterieure);
00630         QString temps = horodatage;
00631         ui->label_maj_temp_int->setText(temps);
00632     }
00636     QPointF mesure(mesuresTemperatureInterieure.size(),
      temperatureInterieure);
00637     mesuresTemperatureInterieure.push_back(mesure);
00638     qDebug() << Q_FUNC_INFO << nomDeLaRuche << "Nouvelle température intérieure :" <<
      temperatureInterieure;
00639 }
```

8.4.4 Documentation des données membres

8.4.4.1 charge

```
QLineSeries* Ihm::charge  [private]
```

La courbe de la charge.

Définition à la ligne 132 du fichier ihm.h.

Référencé par setValeurCharge().

8.4.4.2 communication

```
Communication* Ihm::communication  [private]
```

association vers la classe Communication

Définition à la ligne 110 du fichier ihm.h.

Référencé par ajouterNouvelleRuche(), connecterRuches(), demarrerTTN(), initialiserEvenements(), on_pushButton_connexion_ttn_clicked(), et on_pushButton_supprimer_ruche_clicked().

8.4.4.3 configuration

```
Configuration* Ihm::configuration  [private]
```

association vers la classe Configuration

Définition à la ligne 111 du fichier ihm.h.

Référencé par ajouterNouvelleRuche(), chargerConfiguration(), demarrerTTN(), initialiserEvenements(), et on_pushButton_supprimer_ruche_clicked().

8.4.4.4 ensoleillement

```
QLineSeries* Ihm::ensoleillement  [private]
```

La courbe de l'ensoleillement.

Définition à la ligne 123 du fichier ihm.h.

Référencé par actualiserGraphiqueEnsoleillement(), initialiserGraphiqueEnsoleillement(), et setValeurEnsoleillement().

8.4.4.5 humidite

```
QLineSeries* Ihm::humidite  [private]
```

La courbe de l'humidite.

Définition à la ligne 120 du fichier ihm.h.

Référencé par afficherGraphiqueHumidite(), initialiserGraphiqueHumidite(), setValeurHumiditeExterieure(), et setValeurHumiditeInterieure().

8.4.4.6 iconeEtatSysteme

```
QSystemTrayIcon* Ihm::iconeEtatSysteme  [private]
```

entrée dans la barre d'état du système

Définition à la ligne 109 du fichier ihm.h.

Référencé par closeEvent(), et initialiserEntreeBarreEtatSysteme().

8.4.4.7 ihmNouvelleRuche

```
IHMNouvelleRuche* Ihm::ihmNouvelleRuche  [private]
```

association vers l'IHM de création d'une nouvelle ruche

Définition à la ligne 107 du fichier ihm.h.

Référencé par initialiserEvenements(), on_pushButton_nouvelle_ruche_clicked(), et ~Ihm().

8.4.4.8 ihmReglageRuche

```
IHMReglageRuche* Ihm::ihmReglageRuche  [private]
```

association vers l'IHM de réglage d'une ruche

Définition à la ligne 108 du fichier ihm.h.

Référencé par on_pushButton_reglage_clicked(), et ~Ihm().

8.4.4.9 mesuresCharge

```
QVector<QPointF> Ihm::mesuresCharge  [private]
```

Les mesures pour la courbe de la charge.

Définition à la ligne 133 du fichier ihm.h.

Référencé par setValeurCharge().

8.4.4.10 mesuresEnsoleillement

```
QVector<QPointF> Ihm::mesuresEnsoleillement  [private]
```

Les mesures pour la courbe de l'ensoleillement.

Définition à la ligne 124 du fichier ihm.h.

Référencé par actualiserGraphiqueEnsoleillement(), et setValeurEnsoleillement().

8.4.4.11 mesuresHumidite

```
QVector<QPointF> Ihm::mesuresHumidite  [private]
```

Les mesures pour la courbe de l'humidité.

Définition à la ligne 121 du fichier ihm.h.

Référencé par afficherGraphiqueHumidite(), setValeurHumiditeExterieure(), et setValeurHumiditeInterieure().

8.4.4.12 mesuresPoids

```
QVector<QPointF> Ihm::mesuresPoids  [private]
```

Les mesures pour la courbe du poids.

Définition à la ligne 130 du fichier ihm.h.

Référencé par afficherGraphiquePoids(), et setValeurPoids().

8.4.4.13 mesuresPression

```
QVector<QPointF> Ihm::mesuresPression  [private]
```

Les mesures pour la courbe de la pression.

Définition à la ligne 127 du fichier ihm.h.

Référencé par afficherGraphiquePression(), et setValeurPression().

8.4.4.14 mesuresTemperatureExterieure

```
QVector<QPointF> Ihm::mesuresTemperatureExterieure  [private]
```

Les mesures pour la courbe de la température extérieure.

Définition à la ligne 118 du fichier ihm.h.

Référencé par afficherGraphiqueTemperatureExterieure(), et setValeurTemperatureExterieure().

8.4.4.15 mesuresTemperatureInterieure

```
QVector<QPointF> Ihm::mesuresTemperatureInterieure  [private]
```

Les mesures pour la courbe de la température intérieure.

Définition à la ligne 115 du fichier ihm.h.

Référencé par afficherGraphiqueTemperatureInterieure(), et setValeurTemperatureInterieure().

8.4.4.16 poids

```
QLineSeries* Ihm::poids  [private]
```

La courbe du poids.

Définition à la ligne 129 du fichier ihm.h.

Référencé par afficherGraphiquePoids(), initialiserGraphiquePoids(), et setValeurPoids().

8.4.4.17 pression

```
QLineSeries* Ihm::pression  [private]
```

La courbe de la pression.

Définition à la ligne 126 du fichier ihm.h.

Référencé par afficherGraphiquePression(), initialiserGraphiquePression(), et setValeurPression().

8.4.4.18 ruches

```
QVector<Ruche> Ihm::ruches  [private]
```

les ruches

Définition à la ligne 112 du fichier ihm.h.

Référencé par ajouterNouvelleRuche(), chargerConfiguration(), connecterRuches(), on_pushButton_supprimer_ruche_clicked(), setValeurCharge(), setValeurEnsoleillement(), setValeurHumiditeExterieure(), setValeurHumiditeInterieure(), setValeurPoids(), setValeurPression(), setValeurTemperatureExterieure(), et setValeurTemperatureInterieure().

8.4.4.19 temperatureExterieure

```
QLineSeries* Ihm::temperatureExterieure  [private]
```

La courbe de la température extérieure.

Définition à la ligne 117 du fichier ihm.h.

Référencé par afficherGraphiqueTemperatureExterieure(), initialiserGraphiqueTemperatures(), et setValeurTemperatureExterieure().

8.4.4.20 temperatureInterieure

```
QLineSeries* Ihm::temperatureInterieure  [private]
```

La courbe de la température intérieure.

Définition à la ligne 114 du fichier ihm.h.

Référencé par afficherGraphiqueTemperatureInterieure(), initialiserGraphiqueTemperatures(), et setValeurTemperatureInterieure().

8.4.4.21 ui

```
Ui::ihm* Ihm::ui  [private]
```

interface utilisateur

Définition à la ligne 106 du fichier ihm.h.

Référencé par ajouterNouvelleRuche(), changerAbscisseGraphiques(), changerApparenceBouton(), changerDonneesVueGlobale(), changerEtatConnexion(), chargerConfiguration(), chargerIconesBoutons(), connecterRuches(), getNomRuche(), Ihm(), initialiserEvenements(), initialiserGraphiqueEnsoleillement(), initialiserGraphiqueHumidite(), initialiserGraphiquePoids(), initialiserGraphiquePression(), initialiserGraphiqueTemperatures(), initialiserWidgets(), on_pushButton_alertes_clicked(), on_pushButton_connexion_ttn_clicked(), on_pushButton_enregistrer_configuration_ttn_clicked(), on_pushButton_graphiques_clicked(), on_pushButton_mesures_clicked(), on_pushButton_reglage_clicked(), on_pushButton_reglage_ttn_clicked(), on_pushButton_ruches_clicked(), on_pushButton_supprimer_ruche_clicked(), setValeurCharge(), setValeurEnsoleillement(), setValeurHumiditeExterieure(), setValeurHumiditeInterieure(), setValeurPoids(), setValeurPression(), setValeurTemperatureExterieure(), setValeurTemperatureInterieure(), et ~Ihm().

La documentation de cette classe a été générée à partir des fichiers suivants :

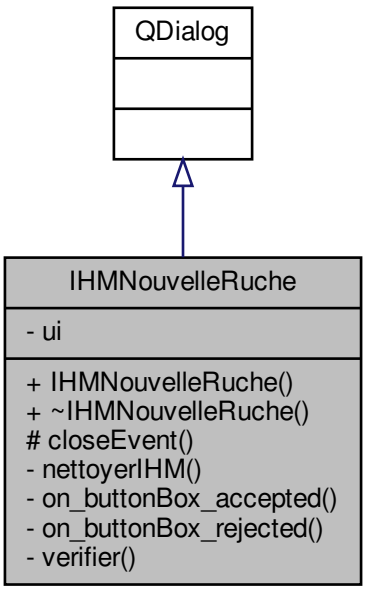
- ihm.h
- ihm.cpp

8.5 Référence de la classe IHMNouvelleRuche

La fenêtre pour créer une nouvelle ruche.

```
#include <nouvelleruche.h>
```

Graphe de collaboration de IHMNouvelleRuche :



Signaux

— void nouvelleRuche (Ruche)

Fonctions membres publiques

— IHMNouvelleRuche (QWidget *parent=nullptr)
Constructeur de la classe IHMNouvelleRuche.
— ~IHMNouvelleRuche ()
Destructeur de la classe IHMNouvelleRuche.

Fonctions membres protégées

— void closeEvent (QCloseEvent *event)
Méthode pour empêcher la fermeture de la fenetre si la ligne de TTN est vide.

Connecteurs privés

— void on_buttonBox_accepted ()
Méthode qui est effectuée si le bouton de confirmation de la fenêtre est pressé.
— void on_buttonBox_rejected ()
Méthode qui est effectuée si le bouton d'annulation de la fenêtre est pressé.
— void verifier ()
Méthode pour activer/désactiver le bouton de confirmation selon si une valeur de topic TTN a été entrée.

Fonctions membres privées

— void nettoyerIHM ()
Méthode pour supprimer les données entré dans l'IHM.

Attributs privés

— Ui : :nouvelleRuche * ui
interface

8.5.1 Description détaillée

La fenêtre pour créer une nouvelle ruche.

Auteur

ACKERMANN Théo

Version

0.2

Définition à la ligne 30 du fichier nouvelleruche.h.

8.5.2 Documentation des constructeurs et destructeur

8.5.2.1 IHMNouvelleRuche()

```
IHMNouvelleRuche::IHMNouvelleRuche (
            QWidget * parent = nullptr )  [explicit]
```

Constructeur de la classe IHMNouvelleRuche.

Paramètres

parent	

Définition à la ligne 16 du fichier nouvelleruche.cpp.

Références ui, et verifier().

```
00016                                                :
00017     QDialog(parent),
00018     ui(new Ui::nouvelleRuche)
00019 {
00020     ui->setupUi(this);
00021     qDebug() << Q_FUNC_INFO;
00022     ui->dateEdit_mise_en_service->setDate(QDate::currentDate());
00023     QPushButton *ok = ui->buttonBox->button(QDialogButtonBox::Ok);
00024     ok->setEnabled(false);
00025     connect(ui->lineEdit_ttn, SIGNAL(textChanged(QString)), this, SLOT(
      verifier()));
00026 }
```

8.5.2.2 ~IHMNouvelleRuche()

```
IHMNouvelleRuche::~IHMNouvelleRuche ( )
```

Destructeur de la classe IHMNouvelleRuche.

Définition à la ligne 31 du fichier nouvelleruche.cpp.

Références ui.

```
00032 {
00033     delete ui;
00034     qDebug() << Q_FUNC_INFO;
00035 }
```

8.5.3 Documentation des fonctions membres

8.5.3.1 closeEvent()

```
void IHMNouvelleRuche::closeEvent (
            QCloseEvent * event )  [protected]
```

Méthode pour empêcher la fermeture de la fenetre si la ligne de TTN est vide.

Paramètres

event	

Définition à la ligne 42 du fichier nouvelleruche.cpp.

Références ui.

```
00043 {
00044     qDebug() << Q_FUNC_INFO;
00045     if(ui->lineEdit_ttn->text().isEmpty())
00046     {
00047         event->ignore();
00048         ui->label_affichage_erreur->setText("Veuillez renseigner un topic TTN.");
00049     }
00050 }
```

8.5.3.2 nettoyerIHM()

```
void IHMNouvelleRuche::nettoyerIHM ( )  [private]
```

Méthode pour supprimer les données entré dans l'IHM.

Définition à la ligne 95 du fichier nouvelleruche.cpp.

Références ui.

Référencé par on_buttonBox_accepted(), et on_buttonBox_rejected().

```
00096 {
00097     ui->lineEdit_nom->clear();
00098     ui->lineEdit_ttn->clear();
00099     ui->lineEdit_adresse->clear();
00100     ui->lineEdit_longitude->clear();
00101     ui->lineEdit_latitude->clear();
00102
00103     ui->label_affichage_erreur->clear();
00104 }
```

8.5.3.3 nouvelleRuche

```
void IHMNouvelleRuche::nouvelleRuche (
            Ruche  )  [signal]
```

Référencé par on_buttonBox_accepted().

8.5.3.4 on_buttonBox_accepted

```
void IHMNouvelleRuche::on_buttonBox_accepted ( )  [private], [slot]
```

Méthode qui est effectuée si le bouton de confirmation de la fenêtre est pressé.

Définition à la ligne 55 du fichier nouvelleruche.cpp.

Références Ruche : :adresse, Ruche : :latitude, Ruche : :longitude, Ruche : :miseEnService, nettoyerIHM(), Ruche : :nom, nouvelle↩Ruche(), Ruche : :topicTTN, et ui.

```
00056 {
00057     Ruche ruche;
00058     ruche.nom = ui->lineEdit_nom->text();
00059     ruche.topicTTN = "mes-ruches/devices/" + ui->lineEdit_ttn->text() + "/up";
00060     ruche.adresse = ui->lineEdit_adresse->text();
00061     ruche.miseEnService = ui->dateEdit_mise_en_service->date().toString("dd/MM/yyyy");
00062     ruche.latitude = ui->lineEdit_latitude->text();
00063     ruche.longitude = ui->lineEdit_longitude->text();
00064     emit nouvelleRuche(ruche);
00065     nettoyerIHM();
00066 }
```

8.5.3.5 on_buttonBox_rejected

```
void IHMNouvelleRuche::on_buttonBox_rejected ( )  [private], [slot]
```

Méthode qui est effectuée si le bouton d'annulation de la fenêtre est pressé.

Définition à la ligne 71 du fichier nouvelleruche.cpp.

Références nettoyerIHM().

```
00072 {
00073     nettoyerIHM();
00074 }
```

8.5.3.6 verifier

```
void IHMNouvelleRuche::verifier ( )  [private], [slot]
```

Méthode pour activer/désactiver le bouton de confirmation selon si une valeur de topic TTN a été entrée.

Définition à la ligne 79 du fichier nouvelleruche.cpp.

Références ui.

Référencé par IHMNouvelleRuche().

```
00080 {
00081     QPushButton *ok = ui->buttonBox->button(QDialogButtonBox::Ok);
00082     if(!ui->lineEdit_ttn->text().isEmpty())
00083     {
00084         ok->setEnabled(true);
00085     }
00086     else
00087     {
00088         ok->setEnabled(false);
00089     }
00090 }
```

8.5.4 Documentation des données membres

8.5.4.1 ui

```
Ui::nouvelleRuche* IHMNouvelleRuche::ui  [private]
```

interface

Définition à la ligne 47 du fichier nouvelleruche.h.

Référencé par closeEvent(), IHMNouvelleRuche(), nettoyerIHM(), on_buttonBox_accepted(), verifier(), et ~IHMNouvelleRuche().

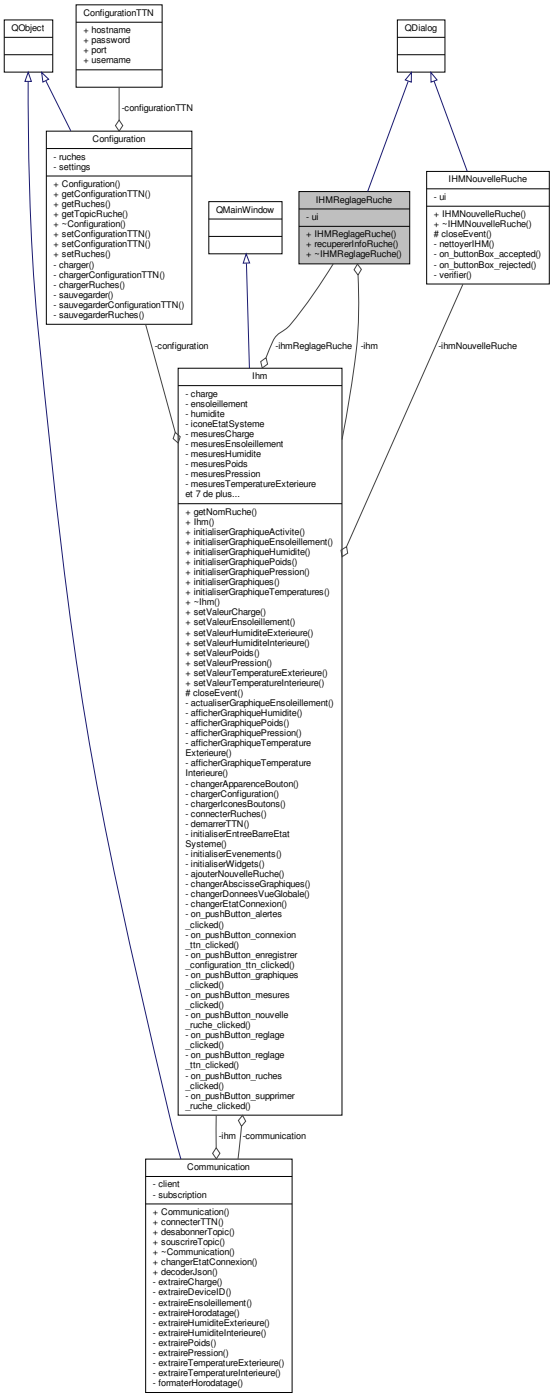
La documentation de cette classe a été générée à partir des fichiers suivants :
- nouvelleruche.h
- nouvelleruche.cpp

8.6 Référence de la classe IHMReglageRuche

La fenêtre pour changer les réglages de la ruche.

`#include <reglageruche.h>`

Graphe de collaboration de IHMReglageRuche :



Fonctions membres publiques

- IHMReglageRuche (QWidget *parent=nullptr)
  Constructeur de la classe IHMReglageRuche.
- void recupererInfoRuche (QString)
- ~IHMReglageRuche ()
  Destructeur de la classe IHMReglageRuche.

Attributs privés

— Ihm * ihm
— Ui : :reglageRuche * ui
interface

8.6.1 Description détaillée

La fenêtre pour changer les réglages de la ruche.

Auteur

ACKERMANN Théo

Version

0.2

Définition à la ligne 27 du fichier reglageruche.h.

8.6.2 Documentation des constructeurs et destructeur

8.6.2.1 IHMReglageRuche()

```
IHMReglageRuche::IHMReglageRuche (
            QWidget * parent = nullptr )  [explicit]
```

Constructeur de la classe IHMReglageRuche.

Paramètres

parent	

Définition à la ligne 17 du fichier reglageruche.cpp.

Références ui.

```
00017                                                :
00018     QDialog(parent),
00019     ui(new Ui::reglageRuche)
00020 {
00021     ui->setupUi(this);
00022     qDebug() << Q_FUNC_INFO;
00023 }
```

8.6.2.2 ~IHMReglageRuche()

```
IHMReglageRuche::~IHMReglageRuche ( )
```

Destructeur de la classe IHMReglageRuche.

Définition à la ligne 28 du fichier reglageruche.cpp.

Références ui.

```
00029 {
00030     delete ui;
00031     qDebug() << Q_FUNC_INFO;
00032 }
```

8.6.3 Documentation des fonctions membres

8.6.3.1 recupererInfoRuche()

```
void IHMReglageRuche::recupererInfoRuche (
            QString nomRuche )
```

Définition à la ligne 34 du fichier reglageruche.cpp.

Références ui.

Référencé par Ihm : :on_pushButton_reglage_clicked().

```
00035 {
00036     qDebug() << Q_FUNC_INFO;
00037     ui->lineEdit_nomRuche->setText(nomRuche);
00038
00039 }
```

8.6.4 Documentation des données membres

8.6.4.1 ihm

```
Ihm* IHMReglageRuche::ihm  [private]
```

Définition à la ligne 39 du fichier reglageruche.h.

8.6.4.2 ui

```
Ui::reglageRuche* IHMReglageRuche::ui  [private]
```

interface

Définition à la ligne 38 du fichier reglageruche.h.

Référencé par IHMReglageRuche(), recupererInfoRuche(), et ∼IHMReglageRuche().

La documentation de cette classe a été générée à partir des fichiers suivants :

- reglageruche.h
- reglageruche.cpp

8.7 Référence de la classe QDialog

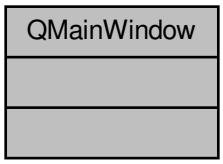
Graphe de collaboration de QDialog :

La documentation de cette classe a été générée à partir du fichier suivant :

— reglageruche.h

8.8 Référence de la classe QMainWindow

Graphe de collaboration de QMainWindow :

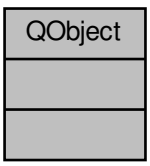


La documentation de cette classe a été générée à partir du fichier suivant :

— ihm.h

8.9 Référence de la classe QObject

Graphe de collaboration de QObject :



La documentation de cette classe a été générée à partir du fichier suivant :

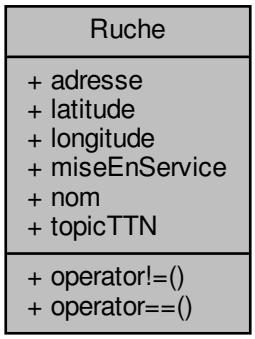
— communication.h

8.10 Référence de la structure Ruche

Structure qui définit une ruche.

`#include <ruche.h>`

Graphe de collaboration de Ruche :



Fonctions membres publiques

- bool operator != (const Ruche &r) const
- bool operator== (const Ruche &r) const

Attributs publics

- QString adresse
  l'adresse
- QString latitude
  la latitude
- QString longitude
  la longitude
- QString miseEnService
  la date de mise en service
- QString nom
  le nom de la ruche
- QString topicTTN
  le topic TTN associé à la ruche

8.10.1 Description détaillée

Structure qui définit une ruche.

Définition à la ligne 17 du fichier ruche.h.

8.10.2 Documentation des fonctions membres

8.10.2.1 operator"!=()

```
bool Ruche::operator!= (
            const Ruche & r ) const  [inline]
```

Définition à la ligne 32 du fichier ruche.h.

```
00033     {
00034         return !(*this == r);
00035     }
```

8.10.2.2 operator==()

```
bool Ruche::operator== (
            const Ruche & r ) const  [inline]
```

Définition à la ligne 25 du fichier ruche.h.

Références topicTTN.

```
00026     {
00027         // le nom aussi ?
00028         if(this->topicTTN != r.topicTTN)
00029                 return false;
00030         return true;
00031     }
```

8.10.3 Documentation des données membres

8.10.3.1 adresse

```
QString Ruche::adresse
```

l'adresse

Définition à la ligne 22 du fichier ruche.h.

Référencé par Configuration : :chargerRuches(), et IHMNouvelleRuche : :on_buttonBox_accepted().

8.10.3.2 latitude

```
QString Ruche::latitude
```

la latitude

Définition à la ligne 23 du fichier ruche.h.

Référencé par Configuration : :chargerRuches(), et IHMNouvelleRuche : :on_buttonBox_accepted().

8.10.3.3 longitude

`QString Ruche::longitude`

la longitude

Définition à la ligne 24 du fichier ruche.h.

Référencé par Configuration : :chargerRuches(), et IHMNouvelleRuche : :on_buttonBox_accepted().

8.10.3.4 miseEnService

`QString Ruche::miseEnService`

la date de mise en service

Définition à la ligne 21 du fichier ruche.h.

Référencé par Configuration : :chargerRuches(), et IHMNouvelleRuche : :on_buttonBox_accepted().

8.10.3.5 nom

`QString Ruche::nom`

le nom de la ruche

Définition à la ligne 19 du fichier ruche.h.

Référencé par Ihm : :ajouterNouvelleRuche(), Configuration : :chargerRuches(), et IHMNouvelleRuche : :on_buttonBox_accepted().

8.10.3.6 topicTTN

`QString Ruche::topicTTN`

le topic TTN associé à la ruche

Définition à la ligne 20 du fichier ruche.h.

Référencé par Ihm : :ajouterNouvelleRuche(), Configuration : :chargerRuches(), IHMNouvelleRuche : :on_buttonBox_accepted(), et operator==().

La documentation de cette structure a été générée à partir du fichier suivant :

— ruche.h

9 Documentation des fichiers

9.1 Référence du fichier Changelog.md

9.2 Changelog.md

```
00001 \page page_changelog Changelog
00002
00003 Revision: 35
00004 Author: tackermann
00005 Date: vendredi 3 avril 2020 05:44:03
00006 Message:
00007 Tag 0.1, Bouml
00008 ----
00009 Revision: 31
00010 Author: tackermann
00011 Date: jeudi 2 avril 2020 16:33:47
00012 Message:
00013 Suppression des @fn
00014 ----
00015 Revision: 30
00016 Author: tackermann
00017 Date: jeudi 2 avril 2020 16:18:10
00018 Message:
00019 Suppresion des @fn de Doxygen
00020 ----
00021 Revision: 26
00022 Author: tvaira
00023 Date: mercredi 1 avril 2020 16:04:11
00024 Message:
00025 Documentation doxygen
00026 ----
00027 Revision: 25
00028 Author: tackermann
00029 Date: mercredi 1 avril 2020 06:24:12
00030 Message:
00031 tag 0.1 avec documentation doxygen dans un zip, suite documentation 0.2
00032 ----
00033 Revision: 24
00034 Author: tackermann
00035 Date: mercredi 25 mars 2020 16:39:35
00036 Message:
00037 Doxygen, utilisation de PagesIHM dans changerApparenceBouton(), réglage du crash dans
       afficherGraphiques()
00038 ----
00039 Revision: 23
00040 Author: tackermann
00041 Date: mercredi 25 mars 2020 14:30:10
00042 Message:
00043 Ajouts des images manquantes
00044 ----
00045 Revision: 22
00046 Author: tvaira
00047 Date: mercredi 25 mars 2020 09:52:42
00048 Message:
00049 Corrections
00050
00051 ----
00052 Revision: 21
00053 Author: tackermann
00054 Date: mercredi 25 mars 2020 00:57:05
00055 Message:
00056 Décomposition decoderJSON(), changements d'icones, documentation
00057 ----
00058 Revision: 20
00059 Author: tackermann
00060 Date: vendredi 20 mars 2020 19:44:15
00061 Message:
00062 Affichage de l'horodatage sur l'IHM avec la metadata/time
00063 ----
00064 Revision: 19
00065 Author: tackermann
00066 Date: vendredi 20 mars 2020 17:03:51
00067 Message:
00068 Horodatage dans l'IHM pour les dernières valeurs reçu
00069 ----
00070 Revision: 17
00071 Author: tvaira
00072 Date: vendredi 20 mars 2020 09:08:40
00073 Message:
00074 Intégration du device dans les données TTN extraites
00075 Exemple d'affichage de mesures (température intérieures) dans le graphique
00076
00077 ----
00078 Revision: 16
```

```
00079 Author: tvaira
00080 Date: jeudi 19 mars 2020 08:39:37
00081 Message:
00082 Modification desabonnerTopic()
00083
00084 ----
00085 Revision: 15
00086 Author: tackermann
00087 Date: jeudi 19 mars 2020 01:40:51
00088 Message:
00089 Suppression de ruche, modifications IHM, doxygen
00090 ----
00091 Revision: 12
00092 Author: tvaira
00093 Date: lundi 16 mars 2020 13:48:10
00094 Message:
00095 Ajout des noms de ruches connues dans la liste déroulante
00096
00097 ----
00098 Revision: 11
00099 Author: tvaira
00100 Date: lundi 16 mars 2020 13:39:07
00101 Message:
00102 Ajout de la notion de Ruche
00103
00104 ----
00105 Revision: 10
00106 Author: tvaira
00107 Date: lundi 16 mars 2020 09:32:20
00108 Message:
00109 Correction partie Configuration
00110 ----
00111 Revision: 8
00112 Author: tackermann
00113 Date: vendredi 13 mars 2020 18:53:00
00114 Message:
00115 Mise en place de MQTT, decodage de trame, création classe Configuration pour le fichier INI
00116 ----
00117 Revision: 6
00118 Author: tvaira
00119 Date: samedi 7 mars 2020 07:55:21
00120 Message:
00121 Révision du code
00122
00123 ----
00124 Revision: 5
00125 Author: tackermann
00126 Date: jeudi 20 février 2020 02:52:10
00127 Message:
00128 Modification de l'IHM, mise en place de graphiques
00129 ----
00130 Revision: 4
00131 Author: tvaira
00132 Date: samedi 15 février 2020 10:57:35
00133 Message:
00134 Configuration du français pour les boîtes de dialogue Qt
00135 Nommage des classes suivant la convention établie en BTS (Majuscule pour
00136 les noms de classe)
00137 Création d'une énumération pour les pages de l'IHM (pas de valeurs
00138 brutes dans le code)
00139
00140 ----
00141 Revision: 3
00142 Author: tackermann
00143 Date: samedi 8 février 2020 12:39:00
00144 Message:
00145 Ajout d'une IHM pour les réglages de la ruche.
00146 ----
00147 Revision: 2
00148 Author: tackermann
00149 Date: jeudi 6 février 2020 21:01:50
00150 Message:
00151 Création dossier beehoneyt, début de la création de l'IHM
00152 ----
```

9.3 Référence du fichier communication.cpp

Déclaration de la classe Communication.

```
#include "communication.h"
#include "ihm.h"
```

9.3.1 Description détaillée

Déclaration de la classe Communication.

Auteur

ACKERMANN Théo

Version

0.2

Définition dans le fichier communication.cpp.

9.4 communication.cpp

```
00001 #include "communication.h"
00002 #include "ihm.h"
00003
00016 Communication::Communication(QObject *parent) :
      QObject(parent), client(new QMqttClient(this))
00017 {
00018     qDebug() << Q_FUNC_INFO;
00019     connect(client, SIGNAL(stateChanged(ClientState)), this, SLOT(
      changerEtatConnexion()));
00020     connect(client, SIGNAL(messageReceived(const QByteArray &, const QMqttTopicName &)), this, SLOT(
      decoderJson(const QByteArray &)));
00021 }
00022
00026 Communication::~Communication()
00027 {
00028     if(client->state() == QMqttClient::Connected)
00029     {
00030         client->disconnectFromHost();
00031     }
00032     qDebug() << Q_FUNC_INFO;
00033 }
00034
00043 void Communication::connecterTTN(QString hostname, int port, QString username,
      QString password)
00044 {
00045     qDebug() << Q_FUNC_INFO << hostname << port << username << password;
00046     if(client->state() == QMqttClient::Disconnected)
00047     {
00048         client->setHostname(hostname);
00049         client->setPort(port);
00050         client->setUsername(username);
00051         client->setPassword(password);
00052         client->connectToHost();
00053     }
00054     else if(client->state() == QMqttClient::Connected)
00055     {
00056         client->disconnectFromHost();
00057     }
00058 }
00059
00065 void Communication::souscrireTopic(QString topic)
00066 {
00067     if(client->state() == QMqttClient::Connected)
00068     {
00069         subscription = client->subscribe(QMqttTopicFilter(topic));
00070         qDebug() << Q_FUNC_INFO << topic;
00071     }
00072 }
00073
00079 void Communication::desabonnerTopic(QString topic)
00080 {
00081     if(client->state() == QMqttClient::Connected)
00082     {
00083         client->unsubscribe(topic);
00084         qDebug() << Q_FUNC_INFO << topic;
00085     }
00086 }
00087
00093 void Communication::decoderJson(const QByteArray &json)
00094 {
00095     QJsonDocument documentJSON = QJsonDocument::fromJson(json);
00096     QString nomDeLaRuche;
00097     QString horodatage;
00098
```

```
00099     qDebug() << Q_FUNC_INFO << json;
00100     if(!documentJSON.isNull())
00101     {
00102         QJsonObject objetJSON = documentJSON.object();
00103         QStringList listeCles = objetJSON.keys();
00104         // les clés sont triés alphabétiquement
00105         for(int i = 0; i < listeCles.count()-1; i++)
00106         {
00107             if(listeCles.at(i) == "dev_id")
00108             {
00109                 nomDeLaRuche = extraireDeviceID(objetJSON, listeCles, i);
00110             }
00111             if(listeCles.at(i) == "metadata")
00112             {
00113                 horodatage = formaterHorodatage(
      extraireHorodatage(objetJSON[listeCles.at(i)].toObject()));
00114             }
00115             if(listeCles.at(i) == "payload_fields")
00116             {
00117                 QJsonObject objet = objetJSON[listeCles.at(i)].toObject();
00118
00119                 if(objet.contains("temperatureInt"))
00120                 {
00121                     emit nouvelleValeurTemperatureInterieure(
      nomDeLaRuche, extraireTemperatureInterieure(objet), horodatage);
00122                 }
00123                 if(objet.contains("temperatureExt"))
00124                 {
00125                     emit nouvelleValeurTemperatureExterieure(
      nomDeLaRuche, extraireTemperatureExterieure(objet), horodatage);
00126                 }
00127                 if(objet.contains("humiditeInt"))
00128                 {
00129                     emit nouvelleValeurHumiditeInterieure(nomDeLaRuche,
      extraireHumiditeInterieure(objet), horodatage);
00130                 }
00131                 if(objet.contains("humiditeExt"))
00132                 {
00133                     emit nouvelleValeurHumiditeExterieure(nomDeLaRuche,
      extraireHumiditeExterieure(objet), horodatage);
00134                 }
00135                 if(objet.contains("ensoleillement"))
00136                 {
00137                     emit nouvelleValeurEnsoleillement(nomDeLaRuche,
      extraireEnsoleillement(objet), horodatage);
00138                 }
00139                 if(objet.contains("pression"))
00140                 {
00141                     emit nouvelleValeurPression(nomDeLaRuche,
      extrairePression(objet), horodatage);
00142                 }
00143                 if(objet.contains("poids"))
00144                 {
00145                     emit nouvelleValeurPoids(nomDeLaRuche,
      extrairePoids(objet), horodatage);
00146                 }
00147                 if(objet.contains("charge"))
00148                 {
00149                     emit nouvelleValeurCharge(nomDeLaRuche,
      extraireCharge(objet), horodatage);
00150                 }
00151             }
00152         }
00153     }
00154 }
00155
00162 QString Communication::extraireHorodatage(QJsonObject objetJSON)
00163 {
00164     return objetJSON.value(QString("time")).toString();
00165 }
00166
00175 QString Communication::extraireDeviceID(QJsonObject objetJSON, QStringList
      listeCles, int position)
00176 {
00177     return objetJSON[listeCles.at(position)].toString();
00178 }
00179
00186 double Communication::extraireTemperatureInterieure(QJsonObject
       objetJSON)
00187 {
00188     return objetJSON.value(QString("temperatureInt")).toDouble();
00189 }
00190
00197 double Communication::extraireTemperatureExterieure(QJsonObject
       objetJSON)
00198 {
00199     return objetJSON.value(QString("temperatureExt")).toDouble();
00200 }
00201
00208 double Communication::extraireHumiditeInterieure(QJsonObject
      objetJSON)
```

```
00209 {
00210     return objetJSON.value(QString("humiditeInt")).toDouble();
00211 }
00212
00219 double Communication::extraireHumiditeExterieure(QJsonObject
      objetJSON)
00220 {
00221     return objetJSON.value(QString("humiditeExt")).toDouble();
00222 }
00229 int Communication::extraireEnsoleillement(QJsonObject objetJSON)
00230 {
00231     return objetJSON.value(QString("ensoleillement")).toInt();
00232 }
00233
00240 int Communication::extrairePression(QJsonObject objetJSON)
00241 {
00242     return objetJSON.value(QString("pression")).toInt();
00243 }
00244
00251 double Communication::extrairePoids(QJsonObject objetJSON)
00252 {
00253     return objetJSON.value(QString("poids")).toDouble();
00254 }
00255
00262 int Communication::extraireCharge(QJsonObject objetJSON)
00263 {
00264     return objetJSON.value(QString("charge")).toInt();
00265 }
00266
00273 QString Communication::formaterHorodatage(QString horodatageBrut)
00274 {
00275     QDateTime horodatage = QDateTime::fromString(horodatageBrut, Qt::ISODate).toLocalTime();
00276     return horodatage.toString("dd/MM/yyyy HH:mm:ss");
00277 }
00278
00282 void Communication::changerEtatConnexion()
00283 {
00284     QString message;
00285     switch(client->state())
00286     {
00287         case 0:
00288             message = "Déconnecté";
00289             break;
00290         case 1:
00291             message = "En cours de connexion";
00292             break;
00293         case 2:
00294             message = "Connecté";
00295             break;
00296     }
00297     qDebug()<< Q_FUNC_INFO << client->state() << message;
00298     emit nouvelEtatConnexion(client->state());
00299 }
```

9.5 Référence du fichier communication.h

Déclaration de la classe Communication.

```
#include <QObject>
#include <QtMqtt/QtMqtt>
#include <QDebug>
```

Classes

— class Communication
Permet de recevoir les données.

9.5.1 Description détaillée

Déclaration de la classe Communication.

Auteur

ACKERMANN Théo

Version

0.2

Définition dans le fichier communication.h.

9.6 communication.h

```
00001 #ifndef COMMUNICATION_H
00002 #define COMMUNICATION_H
00003
00011 #include <QObject>
00012 #include <QtMqtt/QtMqtt>
00013 #include <QDebug>
00014
00015 class Ihm;
00016
00024 class Communication  : public QObject
00025 {
00026     Q_OBJECT
00027
00028 public:
00029     Communication(QObject *parent = nullptr);
00030     ~Communication();
00031
00032     void connecterTTN(QString hostname, int port, QString username, QString password);
00033     void souscrireTopic(QString topic);
00034     void desabonnerTopic(QString topic);
00035
00036 private:
00037     QMqttClient *client;
00038     QMqttSubscription *subscription;
00039     Ihm *ihm;
00040
00041     QString extraireHorodatage(QJsonObject objetJSON);
00042     QString extraireDeviceID(QJsonObject objetJSON, QStringList listeCles, int position);
00043     double  extraireTemperatureInterieure(QJsonObject objetJSON);
00044     double  extraireTemperatureExterieure(QJsonObject objetJSON);
00045     double  extraireHumiditeInterieure(QJsonObject objetJSON);
00046     double  extraireHumiditeExterieure(QJsonObject objetJSON);
00047     int     extraireEnsoleillement(QJsonObject objetJSON);
00048     int     extrairePression(QJsonObject objetJSON);
00049     double  extrairePoids(QJsonObject objetJSON);
00050     int     extraireCharge(QJsonObject objetJSON);
00051
00052     QString formaterHorodatage(QString horodatageBrut);
00053
00054 public slots:
00055     void decoderJson(const QByteArray &json);
00056     void changerEtatConnexion();
00057
00058 signals:
00059     void nouvelleValeurTemperatureInterieure(QString nomDeLaRuche,
      double temperatureInterieure, QString horodatage);
00060     void nouvelleValeurTemperatureExterieure(QString nomDeLaRuche,
      double temperatureExterieure, QString horodatage);
00061     void nouvelleValeurEnsoleillement(QString nomDeLaRuche, int ensoleillement,
       QString horodatage);
00062     void nouvelleValeurHumiditeInterieure(QString nomDeLaRuche, double
      humiditeInterieure, QString horodatage);
00063     void nouvelleValeurHumiditeExterieure(QString nomDeLaRuche, double
      humiditeExterieure, QString horodatage);
00064     void nouvelleValeurPression(QString nomDeLaRuche, int pression, QString
      horodatage);
00065     void nouvelleValeurPoids(QString nomDeLaRuche, double poids, QString horodatage);
00066     void nouvelleValeurCharge(QString nomDeLaRuche, int charge, QString horodatage);
00067     void nouvelEtatConnexion(int etat);
00068 };
00069
00070 #endif // COMMUNICATION_H
```

9.7 Référence du fichier configuration.cpp

Déclaration de la classe Configuration.

```
#include "configuration.h"
```

9.7.1 Description détaillée

Déclaration de la classe Configuration.

Auteur

ACKERMANN Théo

Version

0.2

Définition dans le fichier configuration.cpp.

9.8 configuration.cpp

```
00001 #include "configuration.h"
00002
00016 Configuration::Configuration(QObject *parent) :
      QObject(parent), settings(QDir::currentPath() + "/configuration.ini", QSettings::IniFormat)
00017 {
00018     qDebug() << Q_FUNC_INFO;
00019     charger();
00020 }
00021
00025 Configuration::~Configuration()
00026 {
00027     sauvegarder();
00028     qDebug() << Q_FUNC_INFO;
00029 }
00030
00036 void Configuration::setConfigurationTTN(
      ConfigurationTTN configurationTTN)
00037 {
00038     this->configurationTTN = configurationTTN;
00039 }
00040
00049 void Configuration::setConfigurationTTN(QString hostname, int port,
      QString username, QString password)
00050 {
00051     configurationTTN.hostname = hostname;
00052     configurationTTN.port = port;
00053     configurationTTN.username = username;
00054     configurationTTN.password = password;
00055     qDebug() << Q_FUNC_INFO << configurationTTN.hostname <<
      configurationTTN.port <<  configurationTTN.
      username << configurationTTN.password;
00056 }
00057
00063 ConfigurationTTN Configuration::getConfigurationTTN()
       const
00064 {
00065     return configurationTTN;
00066 }
00067
00073 void Configuration::setRuches(QVector<Ruche> ruches)
00074 {
00075     this->ruches = ruches;
00076 }
00077
00083 QVector<Ruche> Configuration::getRuches() const
00084 {
00085     return ruches;
00086 }
00087
00094 QString Configuration::getTopicRuche(QString ruche)
00095 {
00096     ruche = ruche.replace(" ","");
00097     ruche = ruche + "/TopicTTN";
00098     return settings.value(ruche).toString();
00099 }
00103 void Configuration::charger()
00104 {
00105     qDebug() << Q_FUNC_INFO << settings.allKeys() << settings.childKeys();
00106     chargerConfigurationTTN();
00107     chargerRuches();
00108 }
00109
00113 void Configuration::chargerConfigurationTTN()
```

```
00114 {
00115     settings.beginGroup("TTN");
00116     configurationTTN.hostname = settings.value("Hostname").toString();
00117     configurationTTN.port = settings.value("Port").toInt();
00118     configurationTTN.username = settings.value("Username").toString();
00119     configurationTTN.password = settings.value("Password").toString();
00120     settings.endGroup();
00121     settings.sync();
00122     qDebug() << Q_FUNC_INFO << configurationTTN.hostname <<
      configurationTTN.port <<  configurationTTN.
      username << configurationTTN.password;
00123 }
00124
00128 void Configuration::chargerRuches()
00129 {
00130     int nbRuches = settings.value("NbRuches", 0).toInt();
00131     qDebug() << Q_FUNC_INFO << nbRuches;
00132     for(int i = 0; i < nbRuches; i++)
00133     {
00134         Ruche ruche;
00135         QString nomRuche = "Ruche" + QString::number(i+1);
00136         settings.beginGroup(nomRuche);
00137         ruche.nom = settings.value("Nom").toString();
00138         ruche.topicTTN = settings.value("TopicTTN").toString();
00139         ruche.miseEnService = settings.value("MiseEnService").toString();
00140         ruche.adresse = settings.value("Adresse").toString();
00141         ruche.latitude = settings.value("Latitude").toString();
00142         ruche.longitude = settings.value("Longitude").toString();
00143         settings.endGroup();
00144         ruches.push_back(ruche);
00145     }
00146 }
00147
00151 void Configuration::sauvegarder()
00152 {
00153     qDebug() << Q_FUNC_INFO;
00154     sauvegarderConfigurationTTN();
00155     sauvegarderRuches();
00156 }
00157
00161 void Configuration::sauvegarderConfigurationTTN()
00162 {
00163     settings.beginGroup("TTN");
00164     settings.setValue("Hostname", configurationTTN.
      hostname);
00165     settings.setValue("Port", configurationTTN.port);
00166     settings.setValue("Username", configurationTTN.
      username);
00167     settings.setValue("Password", configurationTTN.
      password);
00168     settings.endGroup();
00169 }
00170
00174 void Configuration::sauvegarderRuches()
00175 {
00176     qDebug() << Q_FUNC_INFO << ruches.size();
00177     for(int i = 0; i < ruches.size(); i++)
00178     {
00179         QString nomRuche = "Ruche" + QString::number(i+1);
00180         settings.beginGroup(nomRuche);
00181         settings.setValue("Nom", ruches[i].nom);
00182         settings.setValue("TopicTTN", ruches[i].topicTTN);
00183         settings.setValue("MiseEnService", ruches[i].miseEnService);
00184         settings.setValue("Adresse", ruches[i].adresse);
00185         settings.setValue("Latitude", ruches[i].latitude);
00186         settings.setValue("Longitude", ruches[i].longitude);
00187         settings.endGroup();
00188     }
00189     settings.setValue("NbRuches", ruches.size());
00190 }
```

9.9 Référence du fichier configuration.h

Déclaration de la classe Configuration.

```
#include <QDebug>
#include <QSettings>
#include <QDir>
#include <QDateTime>
#include <QObject>
#include "ruche.h"
```

Classes

— class Configuration
Gère le fichier INI.
— struct ConfigurationTTN
Structure qui définit la configuration MQTT pour se connecter au réseau TheThingsNetwork (TTN)

9.9.1 Description détaillée

Déclaration de la classe Configuration.

Auteur

ACKERMANN Théo

Version

0.2

Définition dans le fichier configuration.h.

9.10 configuration.h

```
00001 #ifndef CONFIGURATION_H
00002 #define CONFIGURATION_H
00003
00011 #include <QDebug>
00012 #include <QSettings>
00013 #include <QDir>
00014 #include <QDateTime>
00015 #include <QObject>
00016 #include "ruche.h"
00017
00022 struct ConfigurationTTN
00023 {
00024     QString hostname;
00025     int port;
00026     QString username;
00027     QString password;
00028 };
00029
00037 class Configuration : public QObject
00038 {
00039     Q_OBJECT
00040
00041 public:
00042     Configuration(QObject *parent = nullptr);
00043     ~Configuration();
00044
00045     ConfigurationTTN getConfigurationTTN() const;
00046     QVector<Ruche> getRuches() const;
00047     QString getTopicRuche(QString ruche);
00048
00049 public slots:
00050     void setConfigurationTTN(ConfigurationTTN configurationTTN);
00051     void setConfigurationTTN(QString hostname, int port, QString
      username, QString password);
00052     void setRuches(QVector<Ruche> ruches);
00053
00054 private:
00055     QSettings settings;
00056     ConfigurationTTN configurationTTN;
00057     QVector<Ruche> ruches;
00058
00059     void charger();
00060     void chargerConfigurationTTN();
00061     void chargerRuches();
00062     void sauvegarder();
00063     void sauvegarderConfigurationTTN();
00064     void sauvegarderRuches();
00065 };
00066
00067 #endif // CONFIGURATION_H
```

9.11 Référence du fichier ihm.cpp

Déclaration de la classe Ihm.

```
#include "ihm.h"
#include "ui_ihm.h"
#include "nouvelleruche.h"
#include "reglageruche.h"
#include "communication.h"
#include "configuration.h"
```

9.11.1 Description détaillée

Déclaration de la classe Ihm.

Auteur

ACKERMANN Théo

Version

0.2

Définition dans le fichier ihm.cpp.

9.12 ihm.cpp

```
00001 #include "ihm.h"
00002 #include "ui_ihm.h"
00003 #include "nouvelleruche.h"
00004 #include "reglageruche.h"
00005 #include "communication.h"
00006 #include "configuration.h"
00007
00020 Ihm::Ihm(QWidget *parent) :QMainWindow(parent),ui(new Ui::ihm),ihmNouvelleRuche(new
      IHMNouvelleRuche),ihmReglageRuche(new IHMReglageRuche()), iconeEtatSysteme(
      new QSystemTrayIcon(this)), communication(new Communication(this)), configuration(new
      Configuration(this))
00021 {
00022     ui->setupUi(this);
00023     qDebug() << Q_FUNC_INFO;
00024
00025     chargerConfiguration();
00026     chargerIconesBoutons();
00027
00028     initialiserEvenements();
00029     initialiserWidgets();
00030     initialiserGraphiques();
00031
00032     demarrerTTN();
00033
00034     showMaximized();
00035 }
00036
00040 Ihm::~Ihm()
00041 {
00042     delete ihmNouvelleRuche;
00043     delete ihmReglageRuche;
00044     delete ui;
00045     qDebug() << Q_FUNC_INFO;
00046 }
00047
00051 void Ihm::on_pushButton_ruches_clicked()
00052 {
00053     ui->stackedWidget->setCurrentIndex(PagesIHM::PAGE_ACCUEIL);
00054     changerApparenceBouton(PagesIHM::PAGE_ACCUEIL);
00055 }
00056
00060 void Ihm::on_pushButton_mesures_clicked()
00061 {
```

```
00062     ui->stackedWidget->setCurrentIndex(PagesIHM::PAGE_VUE_GLOBALE);
00063     changerApparenceBouton(PagesIHM::PAGE_VUE_GLOBALE);
00064 }
00065
00069 void Ihm::on_pushButton_graphiques_clicked()
00070 {
00071     ui->stackedWidget->setCurrentIndex(PagesIHM::PAGE_GRAPHIQUES);
00072     changerApparenceBouton(PagesIHM::PAGE_GRAPHIQUES);
00073 }
00074
00078 void Ihm::on_pushButton_alertes_clicked()
00079 {
00080     ui->stackedWidget->setCurrentIndex(PagesIHM::PAGE_ALERTES);
00081     changerApparenceBouton(PagesIHM::PAGE_ALERTES);
00082 }
00083
00087 void Ihm::on_pushButton_reglage_ttn_clicked()
00088 {
00089     chargerConfiguration();
00090     ui->stackedWidget->setCurrentIndex(PagesIHM::PAGE_REGLAGES_TTN);
00091     changerApparenceBouton(PagesIHM::PAGE_REGLAGES_TTN);
00092 }
00093
00099 void Ihm::changerApparenceBouton(PagesIHM page)
00100 {
00101     if(page == PagesIHM::PAGE_ACCUEIL)
00102     {
00103         ui->pushButton_ruches->setStyleSheet("background:#666666;");
00104         ui->pushButton_mesures->setStyleSheet("");
00105         ui->pushButton_graphiques->setStyleSheet("");
00106         ui->pushButton_alertes->setStyleSheet("");
00107         ui->pushButton_reglage_ttn->setStyleSheet("");
00108     }
00109     if(page == PagesIHM::PAGE_VUE_GLOBALE)
00110     {
00111         ui->pushButton_ruches->setStyleSheet("");
00112         ui->pushButton_mesures->setStyleSheet("background:#666666;");
00113         ui->pushButton_graphiques->setStyleSheet("");
00114         ui->pushButton_alertes->setStyleSheet("");
00115         ui->pushButton_reglage_ttn->setStyleSheet("");
00116     }
00117     if(page == PagesIHM::PAGE_GRAPHIQUES)
00118     {
00119         ui->pushButton_ruches->setStyleSheet("");
00120         ui->pushButton_mesures->setStyleSheet("");
00121         ui->pushButton_graphiques->setStyleSheet("background:#666666;");
00122         ui->pushButton_alertes->setStyleSheet("");
00123         ui->pushButton_reglage_ttn->setStyleSheet("");
00124     }
00125     if(page == PagesIHM::PAGE_ALERTES)
00126     {
00127         ui->pushButton_ruches->setStyleSheet("");
00128         ui->pushButton_mesures->setStyleSheet("");
00129         ui->pushButton_graphiques->setStyleSheet("");
00130         ui->pushButton_alertes->setStyleSheet("background:#666666;");
00131         ui->pushButton_reglage_ttn->setStyleSheet("");
00132     }
00133     if(page == PagesIHM::PAGE_REGLAGES_TTN)
00134     {
00135         ui->pushButton_ruches->setStyleSheet("");
00136         ui->pushButton_mesures->setStyleSheet("");
00137         ui->pushButton_graphiques->setStyleSheet("");
00138         ui->pushButton_alertes->setStyleSheet("");
00139         ui->pushButton_reglage_ttn->setStyleSheet("background:#666666;");
00140     }
00141 }
00142
00146 void Ihm::on_pushButton_connexion_ttn_clicked()
00147 {
00148     communication->connecterTTN(ui->lineEdit_host->text(),
      ui->spinBox_port->value(), ui->lineEdit_username->text(), ui->lineEdit_password->text());
00149 }
00150
00154 void Ihm::on_pushButton_nouvelle_ruche_clicked()
00155 {
00156     ihmNouvelleRuche->exec();
00157 }
00158
00162 void Ihm::on_pushButton_reglage_clicked()
00163 {
00164     if(ui->comboBox_liste_ruches->currentText() == nullptr)
00165     {
00166         QMessageBox::warning(this,"Erreur","Il n'y a pas de ruche.");
00167         qDebug() << Q_FUNC_INFO << "Il n'y a pas de ruche.";
00168     }
00169     else
00170     {
00171         ihmReglageRuche->recupererInfoRuche(ui->comboBox_liste_ruches->
      currentText());
00172         ihmReglageRuche->exec();
00173     }
```

```
00174 }
00175
00179 void Ihm::on_pushButton_supprimer_ruche_clicked()
00180 {
00181     if(ui->comboBox_liste_ruches->currentText() == nullptr | ui->comboBox_liste_ruches->currentText() =
      = "Nom de la ruche")
00182     {
00183         QMessageBox::warning(this,"Erreur","Il n'y a pas de ruche.");
00184         qDebug() << Q_FUNC_INFO << "Il n'y a pas de ruche.";
00185     }
00186     else
00187     {
00188         QMessageBox::StandardButton reponse;
00189         QString nom_ruche = ui->comboBox_liste_ruches->currentText();
00190         QString question = "Êtes-vous sûr de vouloir supprimer la ruche '" + nom_ruche + "' ?";
00191         reponse = QMessageBox::question(this,"",question,QMessageBox::Yes|QMessageBox::No);
00192
00193         if(reponse == QMessageBox::Yes)
00194         {
00195             qDebug() << Q_FUNC_INFO << "reponse : Oui";
00196             communication->desabonnerTopic(ruches[
      ui->comboBox_liste_ruches->currentIndex()].topicTTN);
00197             ruches.remove(ui->comboBox_liste_ruches->currentIndex());
00198             configuration->setRuches(ruches);
00199             ui->comboBox_liste_ruches->removeItem(ui->comboBox_liste_ruches->currentIndex());
00200         }
00201         else
00202         {
00203             qDebug() << Q_FUNC_INFO << "reponse : Non";
00204         }
00205     }
00206 }
00207
00211 void Ihm::initialiserGraphiques()
00212 {
00213     initialiserGraphiqueTemperatures();
00214     initialiserGraphiqueHumidite();
00215     initialiserGraphiqueEnsoleillement();
00216     initialiserGraphiquePression();
00217     initialiserGraphiquePoids();
00218     //initialiserGraphiqueActivite();
00219
00220     qDebug() << Q_FUNC_INFO;
00221 }
00222
00226 void Ihm::initialiserGraphiqueTemperatures()
00227 {
00228     temperatureInterieure = new QLineSeries();
00229     temperatureInterieure->setName("Intérieure");
00230     temperatureInterieure->setPointsVisible(true);
00231
00232     temperatureExterieure = new QLineSeries();
00233     temperatureExterieure->setName("Extérieure");
00234     temperatureExterieure->setPointsVisible(true);
00235
00236     /* Valeurs de test
00237     temperatureExterieure->append(0, 35);
00238     temperatureExterieure->append(1, 37);
00239     temperatureExterieure->append(2, 35);
00240     temperatureExterieure->append(3, 34);
00241     temperatureExterieure->append(4, 31);*/
00242
00243     QChart *chart = new QChart();
00244     chart->legend()->show();
00245     chart->addSeries(temperatureInterieure);
00246     chart->addSeries(temperatureExterieure);
00247
00248     chart->createDefaultAxes();
00249     chart->setTitle("Températures");
00250     ui->chartView_temperature->setChart(chart);
00251     ui->chartView_temperature->setRenderHint(QPainter::Antialiasing);
00252
00256     QDateTimeAxis *axisX = new QDateTimeAxis();
00257     axisX->setTickCount(7);
00258     axisX->setFormat("dd/MM");
00259     axisX->setTitleText("Jours");
00260     // ou :
00261     //axisX->setTickCount(10);
00262     //axisX->setFormat("hh:mm");
00263     //axisX->setTitleText("Heure");
00264     axisX->setMin(QDateTime::currentDateTime().addDays(-3));
00265     axisX->setMax(QDateTime::currentDateTime().addDays(3));
00266
00267     QValueAxis *axisY = new QValueAxis();
00268     axisY->setTitleText("°C");
00269     axisY->setTickCount((((AXE_TEMPERATURE_MAX - (
      AXE_TEMPERATURE_MIN))*2)/10)+1);
00270     axisY->setMin(AXE_TEMPERATURE_MIN);
00271     axisY->setMax(AXE_TEMPERATURE_MAX);
00272
00273     chart->setAxisY(axisY);
```

```
00274     chart->setAxisX(axisX);
00275 }
00276
00280 void Ihm::initialiserGraphiqueHumidite()
00281 {
00282     humidite = new QLineSeries();
00283     /* Valeurs de test
00284     humidite->append(0, 27);
00285     humidite->append(1, 26);
00286     humidite->append(2, 28);
00287     humidite->append(3, 31);
00288     humidite->append(4, 24);*/
00289
00290     QChart *chart = new QChart();
00291     chart->legend()->hide();
00292     chart->addSeries(humidite);
00293     chart->setTitle("Humidité");
00294     ui->chartView_humidite->setChart(chart);
00295     ui->chartView_humidite->setRenderHint(QPainter::Antialiasing);
00296
00297     QDateTimeAxis *axisX = new QDateTimeAxis();
00298     axisX->setTickCount(7);
00299     axisX->setFormat("dd/MM");
00300     axisX->setTitleText("Jours");
00301     // ou :
00302     //axisX->setTickCount(10);
00303     //axisX->setFormat("hh:mm");
00304     //axisX->setTitleText("Heure");
00305     axisX->setMin(QDateTime::currentDateTime().addDays(-3));
00306     axisX->setMax(QDateTime::currentDateTime().addDays(3));
00307
00308     QValueAxis *axisY = new QValueAxis();
00309     axisY->setTitleText("%");
00310     axisY->setTickCount((((AXE_TEMPERATURE_MAX - (
      AXE_TEMPERATURE_MIN))*2)/10)+1);
00311     axisY->setMin(AXE_TEMPERATURE_MIN);
00312     axisY->setMax(AXE_TEMPERATURE_MAX);
00313
00314     chart->setAxisY(axisY);
00315     chart->setAxisX(axisX);
00316 }
00317
00321 void Ihm::initialiserGraphiqueEnsoleillement()
00322 {
00323     ensoleillement = new QLineSeries();
00324
00325     QChart *chart = new QChart();
00326     chart->legend()->hide();
00327     chart->addSeries(ensoleillement);
00328     chart->setTitle("Ensoleillement");
00329     ensoleillement->setPointsVisible(true);
00330     ui->chartView_ensoleillement->setChart(chart);
00331     ui->chartView_ensoleillement->setRenderHint(QPainter::Antialiasing);
00332
00333     QDateTimeAxis *axisX = new QDateTimeAxis();
00334     axisX->setTickCount(24);
00335     axisX->setFormat("hh");
00336     axisX->setTitleText("Heure");
00337     QDateTime qdatetime;
00338     qint64 msDepuisEpochMin = qdatetime.currentSecsSinceEpoch() - 43200;
00339     qint64 msDepuisEpochMax = qdatetime.currentSecsSinceEpoch() + 43200;
00340
00341     qDebug() << msDepuisEpochMin << msDepuisEpochMax;
00342
00343     //axisX->setMin(qdatetime.fromSecsSinceEpoch(msDepuisEpochMin));
00344     //axisX->setMax(qdatetime.fromSecsSinceEpoch(msDepuisEpochMax));
00345
00346     QValueAxis *axisY = new QValueAxis();
00347     axisY->setTitleText("lux");
00348     axisY->setMin(0);
00349     axisY->setMax(500);
00350
00351     ensoleillement->attachAxis(axisX);
00352     ensoleillement->attachAxis(axisY);
00353
00354     chart->addAxis(axisX, Qt::AlignBottom);
00355     chart->addAxis(axisY, Qt::AlignLeft);
00356     //chart->setAxisY(axisY);
00357     //chart->setAxisX(axisX);
00358
00359     chart->show();
00360 }
00361
00365 void Ihm::initialiserGraphiquePression()
00366 {
00367     pression = new QLineSeries();
00368     // Valeurs de test
00369     pression->append(0, 321);
00370     pression->append(1, 354);
00371     pression->append(2, 396);
00372     pression->append(3, 348);
```

```
00373     pression->append(4, 240);
00374
00375     QChart *chart = new QChart();
00376     chart->legend()->hide();
00377     chart->addSeries(pression);
00378     chart->setTitle("Pression");
00379     ui->chartView_pression->setChart(chart);
00380     ui->chartView_pression->setRenderHint(QPainter::Antialiasing);
00381
00382     QDateTimeAxis *axisX = new QDateTimeAxis();
00383     axisX->setTickCount(7);
00384     axisX->setFormat("dd/MM");
00385     axisX->setTitleText("Jours");
00386     // ou :
00387     //axisX->setTickCount(10);
00388     //axisX->setFormat("hh:mm");
00389     //axisX->setTitleText("Heure");
00390
00391     axisX->setMin(QDateTime::currentDateTime().addDays(-3));
00392     axisX->setMax(QDateTime::currentDateTime().addDays(3));
00393
00394     QValueAxis *axisY = new QValueAxis();
00395     axisY->setTitleText("hPa");
00396     axisY->setMin(0);
00397     axisY->setMax(500);
00398
00399     chart->setAxisY(axisY);
00400     chart->setAxisX(axisX);
00401 }
00402
00406 void Ihm::initialiserGraphiquePoids()
00407 {
00408     poids = new QLineSeries();
00409     // Valeurs de test
00410     poids->append(0, 321);
00411     poids->append(1, 354);
00412     poids->append(2, 396);
00413     poids->append(3, 348);
00414     poids->append(4, 240);
00415
00416     QChart *chart = new QChart();
00417     chart->legend()->hide();
00418     chart->addSeries(poids);
00419     chart->setTitle("Poids");
00420     ui->chartView_poids->setChart(chart);
00421     ui->chartView_poids->setRenderHint(QPainter::Antialiasing);
00422
00423     QDateTimeAxis *axisX = new QDateTimeAxis();
00424     axisX->setTickCount(7);
00425     axisX->setFormat("dd/MM");
00426     axisX->setTitleText("Jours");
00427     // ou :
00428     //axisX->setTickCount(10);
00429     //axisX->setFormat("hh:mm");
00430     //axisX->setTitleText("Heure");
00431
00432     axisX->setMin(QDateTime::currentDateTime().addDays(-3));
00433     axisX->setMax(QDateTime::currentDateTime().addDays(3));
00434
00435     QValueAxis *axisY = new QValueAxis();
00436     axisY->setTitleText("Kg");
00437     axisY->setMin(0);
00438     axisY->setMax(500);
00439
00440     chart->setAxisY(axisY);
00441     chart->setAxisX(axisX);
00442 }
00443
00447 void Ihm::changerAbscisseGraphiques()
00448 {
00449     switch(ui->comboBox_reglages_graphiques->currentIndex())
00450     {
00451         case 0:
00452             qDebug() << Q_FUNC_INFO << "reponse : 1j";
00453             //axisX->setTickCount(10);
00454             //axisX->setFormat("hh:mm");
00455             //axisX->setTitleText("Heure");
00456             break;
00457         case 1:
00458             qDebug() << Q_FUNC_INFO << "reponse : 7j";
00459             //axisX->setTickCount(7);
00460             //axisX->setFormat("dd/MM");
00461             //axisX->setTitleText("Jours");
00462             break;
00463         default:
00464             qDebug() << Q_FUNC_INFO << ui->comboBox_reglages_graphiques->currentIndex();
00465      }
00466 }
00467
00471 void Ihm::changerDonneesVueGlobale()
00472 {
```

```
00473     switch(ui->comboBox_donnees_affiche->currentIndex())
00474     {
00475         case 0:
00476             qDebug() << Q_FUNC_INFO << "Temperature";
00477             break;
00478         case 1:
00479             qDebug() << Q_FUNC_INFO << "Humidité";
00480             break;
00481         default:
00482             qDebug() << Q_FUNC_INFO << ui->comboBox_reglages_graphiques->currentIndex();
00483      }
00484 }
00485
00491 void Ihm::changerEtatConnexion(int etat)
00492 {
00493     switch(etat)
00494     {
00495         case 0:
00496             ui->label_etat_connexion->setPixmap(QPixmap(":/off.png"));
00497             ui->pushButton_connexion_ttn->setText("Connecter");
00498             break;
00499         case 1:
00500             ui->label_etat_connexion->setPixmap(QPixmap(":/connexion.png"));
00501             break;
00502         case 2:
00503             ui->label_etat_connexion->setPixmap(QPixmap(":/on.png"));
00504             ui->pushButton_connexion_ttn->setText("Déconnecter");
00505             connecterRuches();
00506             break;
00507     }
00508 }
00509
00515 void Ihm::closeEvent(QCloseEvent *event)
00516 {
00517     if (iconeEtatSysteme->isVisible())
00518     {
00519         QMessageBox::information(this, NOM_APPLICATION, "Utiliser le menu Quitter pour
      mettre fin à l'application.");
00520         hide();
00521         event->ignore();
00522     }
00523 }
00524
00528 void Ihm::chargerIconesBoutons()
00529 {
00530     ui->pushButton_ruches->setIcon(QIcon(":/ruche.png"));
00531     ui->pushButton_graphiques->setIcon(QIcon(":/graphiques.png"));
00532     ui->pushButton_alertes->setIcon(QIcon(":/alertes.png"));
00533     ui->pushButton_reglage_ttn->setIcon(QIcon(":/reglages.png"));
00534     ui->pushButton_mesures->setIcon(QIcon(":/vue_globale.png"));
00535 }
00536
00540 void Ihm::initialiserWidgets()
00541 {
00542     this->setWindowTitle(NOM_APPLICATION);
00543     ui->label_version->setText(VERSION_APPLICATION);
00544
00545     ui->comboBox_liste_ruches->addItem("Nom de la ruche");
00546
00547     ui->comboBox_reglages_graphiques->addItem("1 jour");
00548     ui->comboBox_reglages_graphiques->addItem("7 jours");
00549
00550     ui->comboBox_donnees_affiche->addItem("Température");
00551     ui->comboBox_donnees_affiche->addItem("Humidité");
00552 }
00553
00557 void Ihm::initialiserEvenements()
00558 {
00559     connect(ui->comboBox_reglages_graphiques, SIGNAL(currentIndexChanged(int)), SLOT(
      changerAbscisseGraphiques()));
00560     connect(ui->comboBox_donnees_affiche, SIGNAL(currentIndexChanged(int)), SLOT(
      changerDonneesVueGlobale()));
00561
00562     // Configuration
00563     connect(this, SIGNAL(sauvegarderConfigurationTTN(QString,int,QString,QString
      )), configuration, SLOT(setConfigurationTTN(QString,int,QString,QString)));
00564
00565     // Ajouter une nouvelle ruche
00566     connect(ihmNouvelleRuche, SIGNAL(nouvelleRuche(Ruche)), this, SLOT(
      ajouterNouvelleRuche(Ruche)));
00567
00568     // Afficher valeur reçue TTN
00569     connect(communication, SIGNAL(nouvelleValeurTemperatureInterieure(QString,double,QString))
      , this, SLOT(setValeurTemperatureInterieure(QString,double,QString)));
00570     connect(communication, SIGNAL(nouvelleValeurTemperatureExterieure(QString,double,QString))
      , this, SLOT(setValeurTemperatureExterieure(QString,double,QString)));
00571
00572     connect(communication, SIGNAL(nouvelleValeurHumiditeInterieure(QString,double,QString)),
      this, SLOT(setValeurHumiditeInterieure(QString,double,QString)));
00573     connect(communication, SIGNAL(nouvelleValeurHumiditeExterieure(QString,double,QString)),
      this, SLOT(setValeurHumiditeExterieure(QString,double,QString)));
```

```
00574
00575     connect(communication, SIGNAL(nouvelleValeurEnsoleillement(QString,int,QString)), this,
      SLOT(setValeurEnsoleillement(QString,int,QString)));
00576     connect(communication, SIGNAL(nouvelleValeurPression(QString,int,QString)), this, SLOT(
      setValeurPression(QString,int,QString)));
00577     connect(communication, SIGNAL(nouvelleValeurPoids(QString,double,QString)), this, SLOT(
      setValeurPoids(QString,double,QString)));
00578     connect(communication, SIGNAL(nouvelleValeurCharge(QString,int,QString)), this, SLOT(
      setValeurCharge(QString,int,QString)));
00579
00580     // Communication
00581     connect(communication, SIGNAL(nouvelEtatConnexion(int)), this, SLOT(
      changerEtatConnexion(int)));
00582 }
00583
00587 void Ihm::initialiserEntreeBarreEtatSysteme()
00588 {
00589     // Crée les actions pour l'application
00590     QAction *actionMinimiser = new QAction(QString::fromUtf8("Minimiser"), this);
00591     QAction *actionMaximiser = new QAction(QString::fromUtf8("Maximiser"), this);
00592     QAction *actionRestaurer = new QAction(QString::fromUtf8("Restaurer"), this);
00593     QAction *actionQuitter = new QAction(QString::fromUtf8("&Quitter"), this);
00594
00595     // Connecte les actions
00596     connect(actionMinimiser, SIGNAL(triggered(bool)), this, SLOT(hide()));
00597     connect(actionMaximiser, SIGNAL(triggered(bool)), this, SLOT(showMaximized()));
00598     connect(actionRestaurer, SIGNAL(triggered(bool)), this, SLOT(showNormal()));
00599     connect(actionQuitter, SIGNAL(triggered(bool)), qApp, SLOT(quit()));
00600
00601     // Crée le menu pour l'entrée dans la barre d'état système
00602     QMenu * menuEtatSysteme = new QMenu(this);
00603     menuEtatSysteme->addAction(actionMinimiser);
00604     menuEtatSysteme->addAction(actionMaximiser);
00605     menuEtatSysteme->addAction(actionRestaurer);
00606     menuEtatSysteme->addSeparator();
00607     menuEtatSysteme->addAction(actionQuitter);
00608
00609     // Crée l'entrée pour la barre d'état système
00610     iconeEtatSysteme->setContextMenu(menuEtatSysteme);
00611     iconeEtatSysteme->setToolTip(NOM_APPLICATION);
00612     // Crée l'icône pour la barre d'état système
00613     QIcon iconeRuche(":/ruche.png");
00614     iconeEtatSysteme->setIcon(iconeRuche);
00615     setWindowIcon(iconeRuche);
00616
00617     iconeEtatSysteme->show();
00618 }
00619
00625 void Ihm::setValeurTemperatureInterieure(QString nomDeLaRuche, double
      temperatureInterieure, QString horodatage)
00626 {
00627     if(ruches[ui->comboBox_liste_ruches->currentIndex()].topicTTN.contains(nomDeLaRuche))
00628     {
00629         ui->lcdNumber_temperature_interieure->display(temperatureInterieure);
00630         QString temps = horodatage;
00631         ui->label_maj_temp_int->setText(temps);
00632     }
00636     QPointF mesure(mesuresTemperatureInterieure.size(),
      temperatureInterieure);
00637     mesuresTemperatureInterieure.push_back(mesure);
00638     qDebug() << Q_FUNC_INFO << nomDeLaRuche << "Nouvelle température intérieure :" <<
      temperatureInterieure;
00639 }
00640
00648 void Ihm::setValeurTemperatureExterieure(QString nomDeLaRuche, double
      temperatureExterieure, QString horodatage)
00649 {
00650     if(ruches[ui->comboBox_liste_ruches->currentIndex()].topicTTN.contains(nomDeLaRuche))
00651     {
00652         ui->lcdNumber_temperature_exterieure->display(temperatureExterieure);
00653         QString temps = horodatage;
00654         ui->label_maj_temp_ext->setText(temps);
00655     }
00656     QPointF mesure(mesuresTemperatureExterieure.size(),
      temperatureExterieure);
00657     mesuresTemperatureExterieure.push_back(mesure);
00658     qDebug() << Q_FUNC_INFO << nomDeLaRuche << "Nouvelle température extérieure :" <<
      temperatureExterieure;
00659 }
00660
00661
00669 void Ihm::setValeurHumiditeInterieure(QString nomDeLaRuche, double
      humidite, QString horodatage)
00670 {
00671     if(ruches[ui->comboBox_liste_ruches->currentIndex()].topicTTN.contains(nomDeLaRuche))
00672     {
00673         ui->lcdNumber_humidite_interieure->display(humidite);
00674         QString temps = horodatage;
00675         ui->label_maj_humidite_interieure->setText(temps);
00676     }
00677     QPointF mesure(mesuresHumidite.size(), humidite);
```

```
00678     mesuresHumidite.push_back(mesure);
00679     qDebug() << Q_FUNC_INFO << nomDeLaRuche << "Nouvelle humidité intérieure:" <<
      humidite;
00680 }
00681
00689 void Ihm::setValeurHumiditeExterieure(QString nomDeLaRuche, double
      humidite, QString horodatage)
00690 {
00691     if(ruches[ui->comboBox_liste_ruches->currentIndex()].topicTTN.contains(nomDeLaRuche))
00692     {
00693         ui->lcdNumber_humidite_exterieure->display(humidite);
00694         QString temps = horodatage;
00695         ui->label_maj_humidite_exterieure->setText(temps);
00696     }
00697     QPointF mesure(mesuresHumidite.size(), humidite);
00698     mesuresHumidite.push_back(mesure);
00699     qDebug() << Q_FUNC_INFO << nomDeLaRuche << "Nouvelle humidité extérieure:" <<
      humidite;
00700 }
00701
00709 void Ihm::setValeurEnsoleillement(QString nomDeLaRuche, int
      ensoleillement, QString horodatage)
00710 {
00711     if(ruches[ui->comboBox_liste_ruches->currentIndex()].topicTTN.contains(nomDeLaRuche))
00712     {
00713         ui->lcdNumber_ensoleillement->display(ensoleillement);
00714         QString temps = horodatage;
00715         ui->label_maj_ensoleillement->setText(temps);
00716     }
00717     QDateTime qdatetime;
00718     qint64 msDepuisEpoch;
00719     msDepuisEpoch = qdatetime.currentSecsSinceEpoch();
00720     QPointF mesure(msDepuisEpoch, ensoleillement);
00721     mesuresEnsoleillement.push_back(mesure);
00722     qDebug() << Q_FUNC_INFO << nomDeLaRuche << "Nouvelle ensoleillement :" <<
      ensoleillement;
00723     qDebug() << mesuresEnsoleillement.size();
00724     for(int i=0;i<mesuresEnsoleillement.size();i++)
00725         qDebug() << mesuresEnsoleillement[i];
00726     actualiserGraphiqueEnsoleillement();
00727 }
00728
00736 void Ihm::setValeurPression(QString nomDeLaRuche, int
      pression, QString horodatage)
00737 {
00738     if(ruches[ui->comboBox_liste_ruches->currentIndex()].topicTTN.contains(nomDeLaRuche))
00739     {
00740         ui->lcdNumber_pression->display(pression);
00741         QString temps = horodatage;
00742         ui->label_maj_pression->setText(temps);
00743     }
00744     QPointF mesure(mesuresPression.size(), pression);
00745     mesuresPression.push_back(mesure);
00746     qDebug() << Q_FUNC_INFO << nomDeLaRuche << "Nouvelle pression :" << pression;
00747 }
00748
00756 void Ihm::setValeurPoids(QString nomDeLaRuche, double poids, QString horodatage)
00757 {
00758     //poids = poids*0.001; // valeur à un dixième
00759     if(ruches[ui->comboBox_liste_ruches->currentIndex()].topicTTN.contains(nomDeLaRuche))
00760     {
00761         ui->lcdNumber_poids->display(poids);
00762         QString temps = horodatage;
00763         ui->label_maj_poids->setText(temps);
00764     }
00765     QPointF mesure(mesuresPoids.size(), poids);
00766     mesuresPoids.push_back(mesure);
00767     qDebug() << Q_FUNC_INFO << nomDeLaRuche << "Nouveau poids :" << poids;
00768 }
00769
00777 void Ihm::setValeurCharge(QString nomDeLaRuche, int charge, QString horodatage)
00778 {
00779     if(ruches[ui->comboBox_liste_ruches->currentIndex()].topicTTN.contains(nomDeLaRuche))
00780     {
00781         ui->lcdNumber_charge->display(charge);
00782         QString temps = horodatage;
00783         ui->label_maj_charge->setText(temps);
00784     }
00785     QPointF mesure(mesuresCharge.size(), charge);
00786     mesuresCharge.push_back(mesure);
00787     qDebug() << Q_FUNC_INFO << nomDeLaRuche << "Nouvelle charge :" << charge;
00788 }
00789
00793 void Ihm::on_pushButton_enregistrer_configuration_ttn_clicked
      ()
00794 {
00795     emit sauvegarderConfigurationTTN(ui->lineEdit_host->text(),
      ui->spinBox_port->value(), ui->lineEdit_username->text(), ui->lineEdit_password->text());
00796 }
00797
00803 void Ihm::ajouterNouvelleRuche(Ruche ruche)
```

```
00804 {
00805     qDebug() << Q_FUNC_INFO << ruche.nom << ruche.topicTTN;
00806     if(ui->comboBox_liste_ruches->currentText() == "Nom de la ruche")
00807         ui->comboBox_liste_ruches->clear();
00808     ui->comboBox_liste_ruches->addItem(ruche.nom);
00809     communication->souscrireTopic(ruche.topicTTN);
00810     ruches.push_back(ruche);
00811     configuration->setRuches(ruches);
00812 }
00813
00817 void Ihm::chargerConfiguration()
00818 {
00819     ConfigurationTTN configurationTTN = configuration->
      getConfigurationTTN();
00820     ui->lineEdit_host->setText(configurationTTN.hostname);
00821     ui->spinBox_port->setValue(configurationTTN.port);
00822     ui->lineEdit_username->setText(configurationTTN.username);
00823     ui->lineEdit_password->setText(configurationTTN.password);
00824
00825     ruches = configuration->getRuches();
00826 }
00827
00831 void Ihm::demarrerTTN()
00832 {
00833     ConfigurationTTN configurationTTN = configuration->
      getConfigurationTTN();
00834     communication->connecterTTN(configurationTTN.
      hostname, configurationTTN.port, configurationTTN.username, configurationTTN.
      password);
00835 }
00836
00840 void Ihm::connecterRuches()
00841 {
00842     qDebug() << Q_FUNC_INFO << ruches.size();
00843     if(ruches.size() > 0)
00844         ui->comboBox_liste_ruches->clear();
00845     for(int i = 0; i < ruches.size(); i++)
00846     {
00847         qDebug() << Q_FUNC_INFO << ruches[i].nom << ruches[i].topicTTN;
00848         communication->souscrireTopic(ruches[i].topicTTN);
00849         ui->comboBox_liste_ruches->addItem(ruches[i].nom);
00850     }
00851 }
00852
00856 void Ihm::afficherGraphiqueTemperatureInterieure()
00857 {
00858     temperatureInterieure->clear();
00859     for(int i=0;i<mesuresTemperatureInterieure.size();i++)
00860         temperatureInterieure->append(
      mesuresTemperatureInterieure[i]);
00861 }
00862
00866 void Ihm::afficherGraphiqueTemperatureExterieure()
00867 {
00868     temperatureExterieure->clear();
00869     for(int i=0;i<mesuresTemperatureExterieure.size();i++)
00870         temperatureExterieure->append(
      mesuresTemperatureExterieure[i]);
00871 }
00872
00876 void Ihm::afficherGraphiqueHumidite()
00877 {
00878     humidite->clear();
00879     for(int i=0;i<mesuresHumidite.size();i++)
00880         humidite->append(mesuresHumidite[i]);
00881 }
00882
00886 void Ihm::actualiserGraphiqueEnsoleillement()
00887 {
00888     //ensoleillement->clear();
00889     ensoleillement->setPointsVisible(true);
00890     ensoleillement->setVisible(true);
00891     ensoleillement->show();
00892     for(int i=0;i<mesuresEnsoleillement.size();i++)
00893     {
00894         ensoleillement->append(mesuresEnsoleillement[i]);
00895         qDebug() << Q_FUNC_INFO << mesuresEnsoleillement[i];
00896     }
00897
00898     qDebug() << Q_FUNC_INFO << "Nouvelle valeur sur le graphique d'ensoleillement";
00899 }
00900
00904 void Ihm::afficherGraphiquePression()
00905 {
00906     pression->clear();
00907     for(int i=0;i<mesuresPression.size();i++)
00908         pression->append(mesuresPression[i]);
00909 }
00910
00914 void Ihm::afficherGraphiquePoids()
00915 {
```

```
00916     poids->clear();
00917     for(int i=0;i<mesuresPoids.size();i++)
00918         poids->append(mesuresPoids[i]);
00919 }
00920
00921 QString Ihm::getNomRuche()
00922 {
00923     return ui->comboBox_liste_ruches->currentText();
00924 }
```

9.13 Référence du fichier ihm.h

Déclaration de la classe Ihm.

```
#include <QtWidgets>
#include <QSystemTrayIcon>
#include <QtCharts>
#include <QDebug>
#include <QMessageBox>
#include "ruche.h"
```

Classes

— class Ihm
La fenêtre principale de l'application.

Espaces de nommage

— Ui

Macros

— #define AXE_TEMPERATURE_MAX 50
— #define AXE_TEMPERATURE_MIN -10
— #define NOM_APPLICATION "Bee Honey't"
— #define VERSION_APPLICATION "v0.2"

Énumérations

— enum PagesIHM {
PAGE_ACCUEIL, PAGE_VUE_GLOBALE, PAGE_GRAPHIQUES, PAGE_ALERTES,
PAGE_REGLAGES_TTN }
Définit les numéros de page de l'IHM.

9.13.1 Description détaillée

Déclaration de la classe Ihm.

Auteur

ACKERMANN Théo

Version

0.2

Définition dans le fichier ihm.h.

9.13.2 Documentation des macros

9.13.2.1 AXE_TEMPERATURE_MAX

```
#define AXE_TEMPERATURE_MAX 50
```

Définition à la ligne 22 du fichier ihm.h.

Référencé par Ihm : :initialiserGraphiqueHumidite(), et Ihm : :initialiserGraphiqueTemperatures().

9.13.2.2 AXE_TEMPERATURE_MIN

```
#define AXE_TEMPERATURE_MIN -10
```

Définition à la ligne 21 du fichier ihm.h.

Référencé par Ihm : :initialiserGraphiqueHumidite(), et Ihm : :initialiserGraphiqueTemperatures().

9.13.2.3 NOM_APPLICATION

```
#define NOM_APPLICATION "Bee Honey't"
```

Définition à la ligne 18 du fichier ihm.h.

Référencé par Ihm : :closeEvent(), Ihm : :initialiserEntreeBarreEtatSysteme(), et Ihm : :initialiserWidgets().

9.13.2.4 VERSION_APPLICATION

```
#define VERSION_APPLICATION "v0.2"
```

Définition à la ligne 19 du fichier ihm.h.

Référencé par Ihm : :initialiserWidgets().

9.13.3 Documentation du type de l'énumération

9.13.3.1 PagesIHM

```
enum PagesIHM
```

Définit les numéros de page de l'IHM.

Valeurs énumérées

PAGE_ACCUEIL	Page principale.
PAGE_VUE_GLOBALE	Page de la vue globale.
PAGE_GRAPHIQUES	Page des graphiques.
PAGE_ALERTES	Page des dernières alertes.
PAGE_REGLAGES_TTN	Page des réglages de TTN.

Définition à la ligne 28 du fichier ihm.h.

```
00029 {
00030     PAGE_ACCUEIL,
00031     PAGE_VUE_GLOBALE,
00032     PAGE_GRAPHIQUES,
00033     PAGE_ALERTES,
00034     PAGE_REGLAGES_TTN
00035 };
```

9.14 ihm.h

```
00001 #ifndef IHM_H
00002 #define IHM_H
00003
00011 #include <QtWidgets>
00012 #include <QSystemTrayIcon>
00013 #include <QtCharts>
00014 #include <QDebug>
00015 #include <QMessageBox>
00016 #include "ruche.h"
00017
00018 #define NOM_APPLICATION     "Bee Honey't"
00019 #define VERSION_APPLICATION "v0.2"
00020
00021 #define AXE_TEMPERATURE_MIN -10
00022 #define AXE_TEMPERATURE_MAX 50
00023
00028 enum PagesIHM
00029 {
00030     PAGE_ACCUEIL,
00031     PAGE_VUE_GLOBALE,
00032     PAGE_GRAPHIQUES,
00033     PAGE_ALERTES,
00034     PAGE_REGLAGES_TTN
00035 };
00036
00037 class IHMNouvelleRuche;
00038 class IHMReglageRuche;
00039 class Communication;
00040 class Configuration;
00041
00042 namespace Ui {
00043 class ihm;
00044 }
00045
00053 class Ihm : public QMainWindow
00054 {
00055     Q_OBJECT
00056
00057 public:
00058     explicit Ihm(QWidget *parent = nullptr);
00059     ~Ihm();
00060
00061     void initialiserGraphiques();
00062
00063     void initialiserGraphiqueTemperatures();
00064     void initialiserGraphiqueHumidite();
00065     void initialiserGraphiqueEnsoleillement();
00066     void initialiserGraphiquePression();
00067     void initialiserGraphiquePoids();
00068     void initialiserGraphiqueActivite();
00069
00070     QString getNomRuche();
00071
00072 public slots:
00073     void setValeurTemperatureInterieure(QString nomDeLaRuche, double temperatureInterieure, QString
      horodatage);
00074     void setValeurTemperatureExterieure(QString nomDeLaRuche, double temperatureExterieure, QString
```

```
     horodatage);
00075
00076     void setValeurHumiditeInterieure(QString nomDeLaRuche, double humiditeExterieure, QString horodatage);
00077     void setValeurHumiditeExterieure(QString nomDeLaRuche, double humiditeInterieure, QString horodatage);
00078
00079     void setValeurEnsoleillement(QString nomDeLaRuche, int ensoleillement, QString horodatage);
00080     void setValeurPression(QString nomDeLaRuche, int pression, QString horodatage);
00081     void setValeurPoids(QString nomDeLaRuche, double poids, QString horodatage);
00082     void setValeurCharge(QString nomDeLaRuche, int charge, QString horodatage);
00083
00084 private slots:
00085     void on_pushButton_ruches_clicked();
00086     void on_pushButton_mesures_clicked();
00087     void on_pushButton_graphiques_clicked();
00088     void on_pushButton_alertes_clicked();
00089     void on_pushButton_nouvelle_ruche_clicked();
00090     void on_pushButton_reglage_clicked();
00091     void on_pushButton_supprimer_ruche_clicked();
00092     void on_pushButton_reglage_ttn_clicked();
00093     void on_pushButton_connexion_ttn_clicked();
00094     void on_pushButton_enregistrer_configuration_ttn_clicked();
00095
00096     void changerAbscisseGraphiques();
00097     void changerDonneesVueGlobale();
00098     void changerEtatConnexion(int etat);
00099
00100     void ajouterNouvelleRuche(Ruche ruche);
00101
00102 protected:
00103     void closeEvent(QCloseEvent *event);
00104
00105 private:
00106     Ui::ihm *ui;
00107     IHMNouvelleRuche *ihmNouvelleRuche;
00108     IHMReglageRuche *ihmReglageRuche;
00109     QSystemTrayIcon *iconeEtatSysteme;
00110     Communication *communication;
00111     Configuration *configuration;
00112     QVector<Ruche> ruches;
00113
00114     QLineSeries *temperatureInterieure;
00115     QVector<QPointF> mesuresTemperatureInterieure;
00116
00117     QLineSeries *temperatureExterieure;
00118     QVector<QPointF> mesuresTemperatureExterieure;
00119
00120     QLineSeries *humidite;
00121     QVector<QPointF> mesuresHumidite;
00122
00123     QLineSeries *ensoleillement;
00124     QVector<QPointF> mesuresEnsoleillement;
00125
00126     QLineSeries *pression;
00127     QVector<QPointF> mesuresPression;
00128
00129     QLineSeries *poids;
00130     QVector<QPointF> mesuresPoids;
00131
00132     QLineSeries *charge;
00133     QVector<QPointF> mesuresCharge;
00134
00135     void chargerIconesBoutons();
00136     void changerApparenceBouton(PagesIHM);
00137     void initialiserWidgets();
00138     void initialiserEvenements();
00139     void initialiserEntreeBarreEtatSysteme();
00140
00141     void chargerConfiguration();
00142     void demarrerTTN();
00143     void connecterRuches();
00144
00145     void afficherGraphiqueTemperatureInterieure();
00146     void afficherGraphiqueTemperatureExterieure();
00147     void afficherGraphiqueHumidite();
00148     void actualiserGraphiqueEnsoleillement();
00149     void afficherGraphiquePression();
00150     void afficherGraphiquePoids();
00151
00152 signals:
00153     void sauvegarderConfigurationTTN(QString hostname, int port, QString username, QString password);
00154
00155 };
00156
00157 #endif // IHM_H
```

9.15 Référence du fichier main.cpp

Programme principal de l'application desktop qui crée et affiche l'ihm principale.

```
#include "ihm.h"
#include <QApplication>
#include <QTranslator>
#include <QLocale>
#include <QLibraryInfo>
```

Fonctions

— int main (int argc, char ∗argv[ ])

9.15.1 Description détaillée

Programme principal de l'application desktop qui crée et affiche l'ihm principale.

Auteur

ACKERMANN Théo

Version

0.2

Définition dans le fichier main.cpp.

9.15.2 Documentation des fonctions

9.15.2.1 main()

```
int main (
            int argc,
            char * argv[] )
```

Définition à la ligne 14 du fichier main.cpp.

```
00015 {
00016     QApplication a(argc, argv);
00017     QString locale = QLocale::system().name().section('_', 0, 0);
00018     QTranslator translator;
00019     translator.load(QString("qt_") + locale, QLibraryInfo::location(QLibraryInfo::TranslationsPath));
00020     a.installTranslator(&translator);
00021
00022     Ihm w;
00023     w.show();
00024
00025     return a.exec();
00026 }
```

9.16 main.cpp

```
00001 #include "ihm.h"
00002 #include <QApplication>
00003 #include <QTranslator>
00004 #include <QLocale>
00005 #include <QLibraryInfo>
00006
00014 int main(int argc, char *argv[])
00015 {
00016     QApplication a(argc, argv);
00017     QString locale = QLocale::system().name().section('_', 0, 0);
00018     QTranslator translator;
00019     translator.load(QString("qt_") + locale, QLibraryInfo::location(QLibraryInfo::TranslationsPath));
00020     a.installTranslator(&translator);
00021
00022     Ihm w;
00023     w.show();
00024
00025     return a.exec();
00026 }
```

9.17 Référence du fichier nouvelleruche.cpp

Déclaration de la classe IHMNouvelleRuche.

```
#include "nouvelleruche.h"
#include "ui_nouvelleruche.h"
```

9.17.1 Description détaillée

Déclaration de la classe IHMNouvelleRuche.

Auteur

ACKERMANN Théo

Version

0.2

Définition dans le fichier nouvelleruche.cpp.

9.18 nouvelleruche.cpp

```
00001 #include "nouvelleruche.h"
00002 #include "ui_nouvelleruche.h"
00003
00016 IHMNouvelleRuche::IHMNouvelleRuche(QWidget *parent) :
00017     QDialog(parent),
00018     ui(new Ui::nouvelleRuche)
00019 {
00020     ui->setupUi(this);
00021     qDebug() << Q_FUNC_INFO;
00022     ui->dateEdit_mise_en_service->setDate(QDate::currentDate());
00023     QPushButton *ok = ui->buttonBox->button(QDialogButtonBox::Ok);
00024     ok->setEnabled(false);
00025     connect(ui->lineEdit_ttn, SIGNAL(textChanged(QString)), this, SLOT(
      verifier()));
00026 }
00027
00031 IHMNouvelleRuche::~IHMNouvelleRuche()
00032 {
00033     delete ui;
00034     qDebug() << Q_FUNC_INFO;
00035 }
00036
00042 void IHMNouvelleRuche::closeEvent(QCloseEvent *event)
```

```
00043 {
00044     qDebug() << Q_FUNC_INFO;
00045     if(ui->lineEdit_ttn->text().isEmpty())
00046     {
00047         event->ignore();
00048         ui->label_affichage_erreur->setText("Veuillez renseigner un topic TTN.");
00049     }
00050 }
00051
00055 void IHMNouvelleRuche::on_buttonBox_accepted()
00056 {
00057     Ruche ruche;
00058     ruche.nom = ui->lineEdit_nom->text();
00059     ruche.topicTTN = "mes-ruches/devices/" + ui->lineEdit_ttn->text() + "/up";
00060     ruche.adresse = ui->lineEdit_adresse->text();
00061     ruche.miseEnService = ui->dateEdit_mise_en_service->date().toString("dd/MM/yyyy");
00062     ruche.latitude = ui->lineEdit_latitude->text();
00063     ruche.longitude = ui->lineEdit_longitude->text();
00064     emit nouvelleRuche(ruche);
00065     nettoyerIHM();
00066 }
00067
00071 void IHMNouvelleRuche::on_buttonBox_rejected()
00072 {
00073     nettoyerIHM();
00074 }
00075
00079 void IHMNouvelleRuche::verifier()
00080 {
00081     QPushButton *ok = ui->buttonBox->button(QDialogButtonBox::Ok);
00082     if(!ui->lineEdit_ttn->text().isEmpty())
00083     {
00084         ok->setEnabled(true);
00085     }
00086     else
00087     {
00088         ok->setEnabled(false);
00089     }
00090 }
00091
00095 void IHMNouvelleRuche::nettoyerIHM()
00096 {
00097     ui->lineEdit_nom->clear();
00098     ui->lineEdit_ttn->clear();
00099     ui->lineEdit_adresse->clear();
00100     ui->lineEdit_longitude->clear();
00101     ui->lineEdit_latitude->clear();
00102
00103     ui->label_affichage_erreur->clear();
00104 }
```

9.19 Référence du fichier nouvelleruche.h

Déclaration de la classe IHMNouvelleRuche.

```
#include <QDialog>
#include <QCloseEvent>
#include <QPushButton>
#include <QMessageBox>
#include <QDebug>
#include "ruche.h"
```

Classes

— class IHMNouvelleRuche
La fenêtre pour créer une nouvelle ruche.

Espaces de nommage

— Ui

9.19.1 Description détaillée

Déclaration de la classe IHMNouvelleRuche.

Auteur

ACKERMANN Théo

Version

0.2

Définition dans le fichier nouvelleruche.h.

9.20 nouvelleruche.h

```
00001 #ifndef NOUVELLERUCHE_H
00002 #define NOUVELLERUCHE_H
00003
00011 #include <QDialog>
00012 #include <QCloseEvent>
00013 #include <QPushButton>
00014 #include <QMessageBox>
00015 #include <QDebug>
00016 #include "ruche.h"
00017
00018 namespace Ui {
00019 class nouvelleRuche;
00020 }
00021
00022
00030 class IHMNouvelleRuche : public QDialog
00031 {
00032     Q_OBJECT
00033
00034 public:
00035     explicit IHMNouvelleRuche(QWidget *parent = nullptr);
00036     ~IHMNouvelleRuche();
00037
00038 protected:
00039     void closeEvent(QCloseEvent *event);
00040
00041 private slots:
00042     void on_buttonBox_accepted();
00043     void on_buttonBox_rejected();
00044     void verifier();
00045
00046 private:
00047     Ui::nouvelleRuche *ui;
00048     void nettoyerIHM();
00049
00050 signals:
00051     void nouvelleRuche(Ruche);
00052 };
00053
00054 #endif // NOUVELLERUCHE_H
```

9.21 Référence du fichier README.md

9.22 README.md

```
00001 \mainpage Le projet
00002
00003 \tableofcontents
00004
00005 Système autonome permettant de connaître à distance certains paramètres d'une ruche afin d'assurer son
       suivi et d'évaluer la santé des abeilles.
00006
00007 \section section_tdm Table des matières
00008 - \ref page_README
00009 - \ref page_changelog
00010 - \ref page_about
00011 - \ref page_licence
```

```
00012
00013 \section section_infos Informations
00014
00015 \author Théo Ackermann <theoackrm@gmail.com>
00016 \date 2020
00017 \version 0.1
00018 \see https://svn.riouxsvn.com/
00019
00020
00021 \page page_README README
00022
00023 [TOC]
00024
00025 # Projet {#projet}
00026
00027 ## Présentation {#presentation}
00028
00029 Système autonome permettant de connaître à distance certains paramètres d'une ruche afin d'assurer son
       suivi et d'évaluer la santé des abeilles.
00030
00031 Version : PC (*desktop*)
00032
00033 ## Fichier de configuration
00034
00035 Le fichier `configuration.ini` contient :
00036
00037 * les paramètres de connexion au serveur TTN
00038 * le nombre de ruches
00039 * les paramètres de chaque ruche
00040
00041 ```ini
00042 [General]
00043 NbRuches=2
00044
00045 [Ruche1]
00046 Adresse=
00047 Latitude=
00048 Longitude=
00049 MiseEnService=
00050 Nom=Ruche 1
00051 TopicTTN=mes_ruches/devices/ruche_1/up
00052
00053 [Ruche2]
00054 Adresse=
00055 Latitude=
00056 Longitude=
00057 MiseEnService=25/03/2020
00058 Nom=Ruche 2
00059 TopicTTN=mes_ruches/devices/ruche_2/up
00060
00061 [TTN]
00062 Hostname=eu.thethings.network
00063 Password=
00064 Port=1883
00065 Username=mes_ruches
00066 ```
00067
00068 ## Recette de l'application PC (desktop) {#recette}
00069
00070 * Consulter les données d'une ruche
00071 * Recevoir les données actuelles des ruches (MQTT/Json)
00072 * Éditer les ruches
00073
00074 ## Informations {#informations}
00075
00076 \author Théo Ackermann <theoackrm@gmail.com>
00077 \date 2020
00078 \version 0.2
00079 \see https://svn.riouxsvn.com/bee-honey-t
00080
00081
00082 \page page_about A propos
00083
00084 \author Théo Ackermann <theoackrm@gmail.com>
00085 \date 2020
00086 \version 0.2
00087 \see https://svn.riouxsvn.com/bee-honey-t
00088
00089
00090 \page page_licence Licence GPL
00091
00092 This program is free software; you can redistribute it and/or modify
00093 it under the terms of the GNU General Public License as published by
00094 the Free Software Foundation; either version 2 of the License, or
00095 (at your option) any later version.
00096
00097 This program is distributed in the hope that it will be useful,
00098 but WITHOUT ANY WARRANTY; without even the implied warranty of
00099 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00100 GNU General Public License for more details.
00101
```

```
00102 You should have received a copy of the GNU General Public License
00103 along with this program; if not, write to the Free Software
00104 Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
```

9.23 Référence du fichier reglageruche.cpp

Déclaration de la classe IHMReglageRuche.

```
#include "reglageruche.h"
#include "ui_reglageruche.h"
#include "ihm.h"
```

9.23.1 Description détaillée

Déclaration de la classe IHMReglageRuche.

Auteur

ACKERMANN Théo

Version

0.2

Définition dans le fichier reglageruche.cpp.

9.24 reglageruche.cpp

```
00001 #include "reglageruche.h"
00002 #include "ui_reglageruche.h"
00003 #include "ihm.h"
00004
00017 IHMReglageRuche::IHMReglageRuche(QWidget *parent) :
00018     QDialog(parent),
00019     ui(new Ui::reglageRuche)
00020 {
00021     ui->setupUi(this);
00022     qDebug() << Q_FUNC_INFO;
00023 }
00024
00028 IHMReglageRuche::~IHMReglageRuche()
00029 {
00030     delete ui;
00031     qDebug() << Q_FUNC_INFO;
00032 }
00033
00034 void IHMReglageRuche::recupererInfoRuche(QString nomRuche)
00035 {
00036     qDebug() << Q_FUNC_INFO;
00037     ui->lineEdit_nomRuche->setText(nomRuche);
00038
00039 }
```

9.25 Référence du fichier reglageruche.h

Déclaration de la classe IHMReglageRuche.

```
#include <QDialog>
#include <QDebug>
```

Classes

— class IHMReglageRuche
La fenêtre pour changer les réglages de la ruche.

Espaces de nommage

— Ui

9.25.1 Description détaillée

Déclaration de la classe IHMReglageRuche.

Auteur

ACKERMANN Théo

Version

0.2

Définition dans le fichier reglageruche.h.

9.26 reglageruche.h

```
00001 #ifndef REGLAGERUCHE_H
00002 #define REGLAGERUCHE_H
00003
00011 #include <QDialog>
00012 #include <QDebug>
00013
00014 class Ihm;
00015
00016 namespace Ui {
00017 class reglageRuche;
00018 }
00019
00027 class IHMReglageRuche : public QDialog
00028 {
00029     Q_OBJECT
00030
00031 public:
00032     explicit IHMReglageRuche(QWidget *parent = nullptr);
00033     ~IHMReglageRuche();
00034
00035     void recupererInfoRuche(QString);
00036
00037 private:
00038     Ui::reglageRuche *ui;
00039     Ihm *ihm;
00040 };
00041
00042 #endif // REGLAGERUCHE_H
```

9.27 Référence du fichier ruche.h

Déclaration de la structure Ruche.

```
#include <QString>
```

Classes

— struct Ruche
Structure qui définit une ruche.

9.27.1 Description détaillée

Déclaration de la structure Ruche.

Auteur

ACKERMANN Théo

Version

0.2

Définition dans le fichier ruche.h.

9.28 ruche.h

```
00001 #ifndef RUCHE_H
00002 #define RUCHE_H
00003
00011 #include <QString>
00012
00017 struct Ruche
00018 {
00019     QString nom;
00020     QString topicTTN;
00021     QString miseEnService;
00022     QString adresse;
00023     QString latitude;
00024     QString longitude;
00025     bool operator==(const Ruche &r) const
00026     {
00027         // le nom aussi ?
00028         if(this->topicTTN != r.topicTTN)
00029                 return false;
00030         return true;
00031     }
00032     bool operator!=(const Ruche &r) const
00033     {
00034         return !(*this == r);
00035     }
00036 };
00037
00038 #endif // RUCHE_H
```

Index

~Communication
Communication, 7
~Configuration
Configuration, 21
~IHMNouvelleRuche
IHMNouvelleRuche, 61
~IHMReglageRuche
IHMReglageRuche, 65
~Ihm
Ihm, 33

AXE_TEMPERATURE_MAX
ihm.h, 90
AXE_TEMPERATURE_MIN
ihm.h, 90
actualiserGraphiqueEnsoleillement
Ihm, 33
adresse
Ruche, 69
afficherGraphiqueHumidite
Ihm, 33
afficherGraphiquePoids
Ihm, 34
afficherGraphiquePression
Ihm, 34
afficherGraphiqueTemperatureExterieure
Ihm, 34
afficherGraphiqueTemperatureInterieure
Ihm, 35
ajouterNouvelleRuche
Ihm, 35

Changelog.md, 71
changerAbscisseGraphiques
Ihm, 36
changerApparenceBouton
Ihm, 36
changerDonneesVueGlobale
Ihm, 37
changerEtatConnexion
Communication, 7
Ihm, 37
charge
Ihm, 54
charger
Configuration, 21
chargerConfiguration
Ihm, 38
chargerConfigurationTTN
Configuration, 21
chargerIconesBoutons
Ihm, 38
chargerRuches
Configuration, 22
client
Communication, 18
closeEvent
IHMNouvelleRuche, 61
Ihm, 38
Communication, 4
~Communication, 7
changerEtatConnexion, 7
client, 18
Communication, 7
connecterTTN, 8
decoderJson, 9
desabonnerTopic, 10
extraireCharge, 10
extraireDeviceID, 11
extraireEnsoleillement, 11
extraireHorodatage, 12
extraireHumiditeExterieure, 12
extraireHumiditeInterieure, 13
extrairePoids, 13
extrairePression, 14
extraireTemperatureExterieure, 14
extraireTemperatureInterieure, 14
formaterHorodatage, 15
ihm, 18
nouvelEtatConnexion, 15
nouvelleValeurCharge, 15
nouvelleValeurEnsoleillement, 16
nouvelleValeurHumiditeExterieure, 16
nouvelleValeurHumiditeInterieure, 16
nouvelleValeurPoids, 16
nouvelleValeurPression, 16
nouvelleValeurTemperatureExterieure, 17
nouvelleValeurTemperatureInterieure, 17
souscrireTopic, 17
subscription, 18
communication
Ihm, 55
communication.cpp, 72
communication.h, 75
Configuration, 18
~Configuration, 21
charger, 21
chargerConfigurationTTN, 21
chargerRuches, 22
Configuration, 20
configurationTTN, 26
getConfigurationTTN, 22
getRuches, 23
getTopicRuche, 23
ruches, 26
sauvegarder, 24
sauvegarderConfigurationTTN, 24
sauvegarderRuches, 24
setConfigurationTTN, 25
setRuches, 26
settings, 27
configuration
Ihm, 55
configuration.cpp, 76
configuration.h, 78
ConfigurationTTN, 27

hostname, 28
password, 28
port, 28
username, 28
configurationTTN
Configuration, 26
connecterRuches
Ihm, 39
connecterTTN
Communication, 8

decoderJson
Communication, 9
demarrerTTN
Ihm, 39
desabonnerTopic
Communication, 10

ensoleillement
Ihm, 55
extraireCharge
Communication, 10
extraireDeviceID
Communication, 11
extraireEnsoleillement
Communication, 11
extraireHorodatage
Communication, 12
extraireHumiditeExterieure
Communication, 12
extraireHumiditeInterieure
Communication, 13
extrairePoids
Communication, 13
extrairePression
Communication, 14
extraireTemperatureExterieure
Communication, 14
extraireTemperatureInterieure
Communication, 14

formaterHorodatage
Communication, 15

getConfigurationTTN
Configuration, 22
getNomRuche
Ihm, 39
getRuches
Configuration, 23
getTopicRuche
Configuration, 23

hostname
ConfigurationTTN, 28
humidite
Ihm, 55

IHMNouvelleRuche, 59
~IHMNouvelleRuche, 61
closeEvent, 61
IHMNouvelleRuche, 60
nettoyerIHM, 62
nouvelleRuche, 62
on_buttonBox_accepted, 62
on_buttonBox_rejected, 62
ui, 63
verifier, 63
IHMReglageRuche, 64
~IHMReglageRuche, 65
IHMReglageRuche, 65
ihm, 66
recupererInfoRuche, 66
ui, 66
iconeEtatSysteme
Ihm, 55
Ihm, 29
~Ihm, 33
actualiserGraphiqueEnsoleillement, 33
afficherGraphiqueHumidite, 33
afficherGraphiquePoids, 34
afficherGraphiquePression, 34
afficherGraphiqueTemperatureExterieure, 34
afficherGraphiqueTemperatureInterieure, 35
ajouterNouvelleRuche, 35
changerAbscisseGraphiques, 36
changerApparenceBouton, 36
changerDonneesVueGlobale, 37
changerEtatConnexion, 37
charge, 54
chargerConfiguration, 38
chargerIconesBoutons, 38
closeEvent, 38
communication, 55
configuration, 55
connecterRuches, 39
demarrerTTN, 39
ensoleillement, 55
getNomRuche, 39
humidite, 55
iconeEtatSysteme, 55
Ihm, 32
ihmNouvelleRuche, 56
ihmReglageRuche, 56
initialiserEntreeBarreEtatSysteme, 40
initialiserEvenements, 40
initialiserGraphiqueActivite, 41
initialiserGraphiqueEnsoleillement, 41
initialiserGraphiqueHumidite, 42
initialiserGraphiquePoids, 43
initialiserGraphiquePression, 44
initialiserGraphiqueTemperatures, 45
initialiserGraphiques, 44
initialiserWidgets, 46
mesuresCharge, 56
mesuresEnsoleillement, 56
mesuresHumidite, 56
mesuresPoids, 57
mesuresPression, 57
mesuresTemperatureExterieure, 57
mesuresTemperatureInterieure, 57
on_pushButton_alertes_clicked, 46
on_pushButton_connexion_ttn_clicked, 46

on_pushButton_enregistrer_configuration_ttn_clicked, 47
on_pushButton_graphiques_clicked, 47
on_pushButton_mesures_clicked, 47
on_pushButton_nouvelle_ruche_clicked, 48
on_pushButton_reglage_clicked, 48
on_pushButton_reglage_ttn_clicked, 48
on_pushButton_ruches_clicked, 49
on_pushButton_supprimer_ruche_clicked, 49
poids, 57
pression, 58
ruches, 58
sauvegarderConfigurationTTN, 49
setValeurCharge, 50
setValeurEnsoleillement, 50
setValeurHumiditeExterieure, 51
setValeurHumiditeInterieure, 51
setValeurPoids, 52
setValeurPression, 53
setValeurTemperatureExterieure, 53
setValeurTemperatureInterieure, 54
temperatureExterieure, 58
temperatureInterieure, 58
ui, 58
ihm
Communication, 18
IHMReglageRuche, 66
ihm.cpp, 80
ihm.h, 89
AXE_TEMPERATURE_MAX, 90
AXE_TEMPERATURE_MIN, 90
NOM_APPLICATION, 90
PagesIHM, 90
VERSION_APPLICATION, 90
ihmNouvelleRuche
Ihm, 56
ihmReglageRuche
Ihm, 56
initialiserEntreeBarreEtatSysteme
Ihm, 40
initialiserEvenements
Ihm, 40
initialiserGraphiqueActivite
Ihm, 41
initialiserGraphiqueEnsoleillement
Ihm, 41
initialiserGraphiqueHumidite
Ihm, 42
initialiserGraphiquePoids
Ihm, 43
initialiserGraphiquePression
Ihm, 44
initialiserGraphiqueTemperatures
Ihm, 45
initialiserGraphiques
Ihm, 44
initialiserWidgets
Ihm, 46

latitude
Ruche, 69
longitude
Ruche, 69

main
main.cpp, 93
main.cpp, 92
main, 93
mesuresCharge
Ihm, 56
mesuresEnsoleillement
Ihm, 56
mesuresHumidite
Ihm, 56
mesuresPoids
Ihm, 57
mesuresPression
Ihm, 57
mesuresTemperatureExterieure
Ihm, 57
mesuresTemperatureInterieure
Ihm, 57
miseEnService
Ruche, 70

NOM_APPLICATION
ihm.h, 90
nettoyerIHM
IHMNouvelleRuche, 62
nom
Ruche, 70
nouvelEtatConnexion
Communication, 15
nouvelleRuche
IHMNouvelleRuche, 62
nouvelleValeurCharge
Communication, 15
nouvelleValeurEnsoleillement
Communication, 16
nouvelleValeurHumiditeExterieure
Communication, 16
nouvelleValeurHumiditeInterieure
Communication, 16
nouvelleValeurPoids
Communication, 16
nouvelleValeurPression
Communication, 16
nouvelleValeurTemperatureExterieure
Communication, 17
nouvelleValeurTemperatureInterieure
Communication, 17
nouvelleruche.cpp, 94
nouvelleruche.h, 95

on_buttonBox_accepted
IHMNouvelleRuche, 62
on_buttonBox_rejected
IHMNouvelleRuche, 62
on_pushButton_alertes_clicked
Ihm, 46
on_pushButton_connexion_ttn_clicked
Ihm, 46
on_pushButton_enregistrer_configuration_ttn_clicked

Ihm, 47
on_pushButton_graphiques_clicked
Ihm, 47
on_pushButton_mesures_clicked
Ihm, 47
on_pushButton_nouvelle_ruche_clicked
Ihm, 48
on_pushButton_reglage_clicked
Ihm, 48
on_pushButton_reglage_ttn_clicked
Ihm, 48
on_pushButton_ruches_clicked
Ihm, 49
on_pushButton_supprimer_ruche_clicked
Ihm, 49
operator !=
Ruche, 68
operator==
Ruche, 69

PagesIHM
ihm.h, 90
password
ConfigurationTTN, 28
poids
Ihm, 57
port
ConfigurationTTN, 28
pression
Ihm, 58

QDialog, 67
QMainWindow, 67
QObject, 67

README.md, 96
recupererInfoRuche
IHMReglageRuche, 66
reglageruche.cpp, 98
reglageruche.h, 98
Ruche, 68
adresse, 69
latitude, 69
longitude, 69
miseEnService, 70
nom, 70
operator !=, 68
operator==, 69
topicTTN, 70
ruche.h, 99
ruches
Configuration, 26
Ihm, 58

sauvegarder
Configuration, 24
sauvegarderConfigurationTTN
Configuration, 24
Ihm, 49
sauvegarderRuches
Configuration, 24
setConfigurationTTN
Configuration, 25
setRuches
Configuration, 26
setValeurCharge
Ihm, 50
setValeurEnsoleillement
Ihm, 50
setValeurHumiditeExterieure
Ihm, 51
setValeurHumiditeInterieure
Ihm, 51
setValeurPoids
Ihm, 52
setValeurPression
Ihm, 53
setValeurTemperatureExterieure
Ihm, 53
setValeurTemperatureInterieure
Ihm, 54
settings
Configuration, 27
souscrireTopic
Communication, 17
subscription
Communication, 18

temperatureExterieure
Ihm, 58
temperatureInterieure
Ihm, 58
topicTTN
Ruche, 70

Ui, 4
ui
IHMNouvelleRuche, 63
IHMReglageRuche, 66
Ihm, 58
username
ConfigurationTTN, 28

VERSION_APPLICATION
ihm.h, 90
verifier
IHMNouvelleRuche, 63