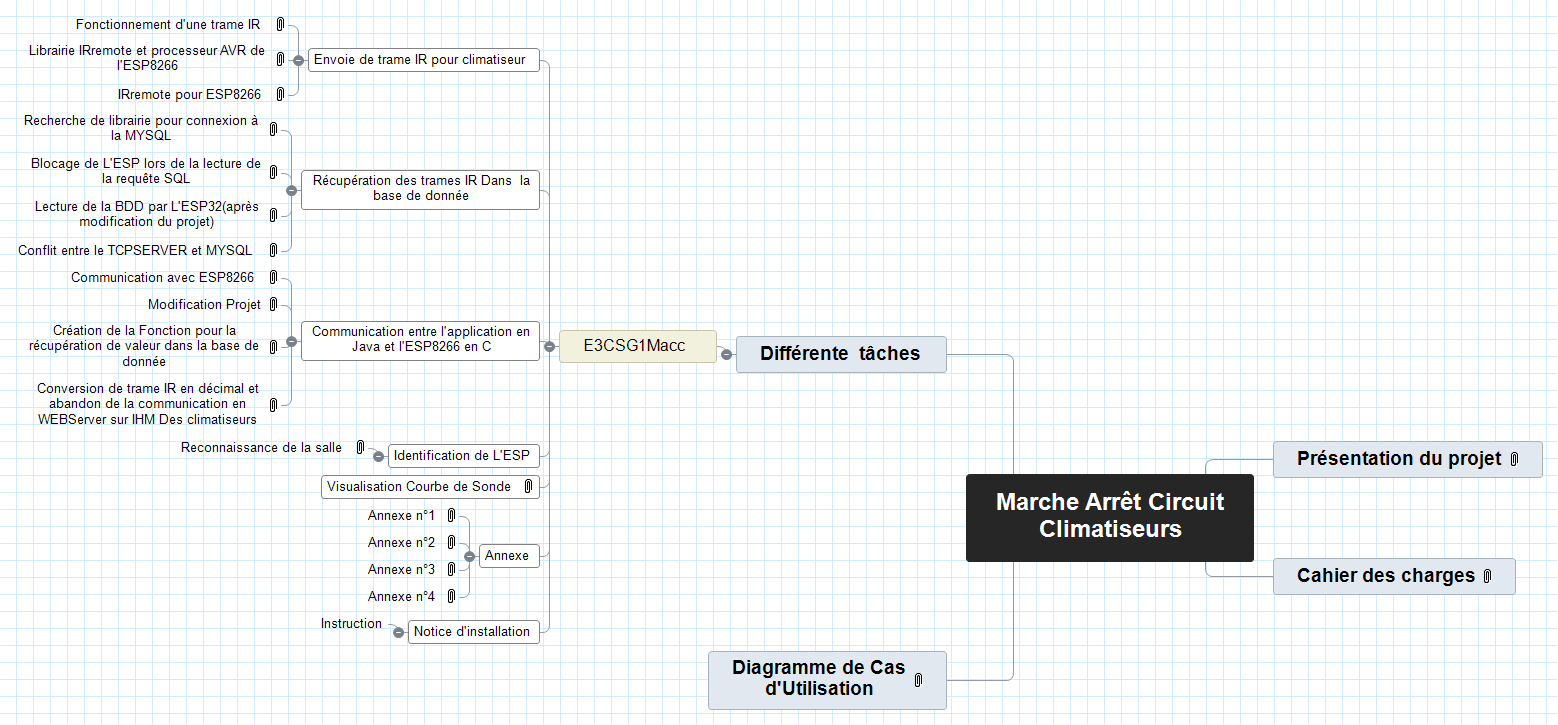
Marche Arrêt Circuit Climatiseurs



[Présentation du projet 2](#_Toc4400094)

[Cahier des charges 2](#_Toc4400095)

[Diagramme de Cas d'Utilisation 3](#_Toc4400096)

[Différente tâches 4](#_Toc4400097)

[E3CSG1Macc 4](#_Toc4400098)

[Envoie de trame IR pour climatiseur 4](#_Toc4400099)

[Récupération des trames IR Dans la base de donnée 11](#_Toc44000100)

[Communication entre l'application en Java et l'ESP8266 en C 13](#_Toc44000101)

[Identification de L'ESP 20](#_Toc44000102)

[Visualisation Courbe de Sonde 20](#_Toc44000103)

[Annexe 20](#_Toc44000104)

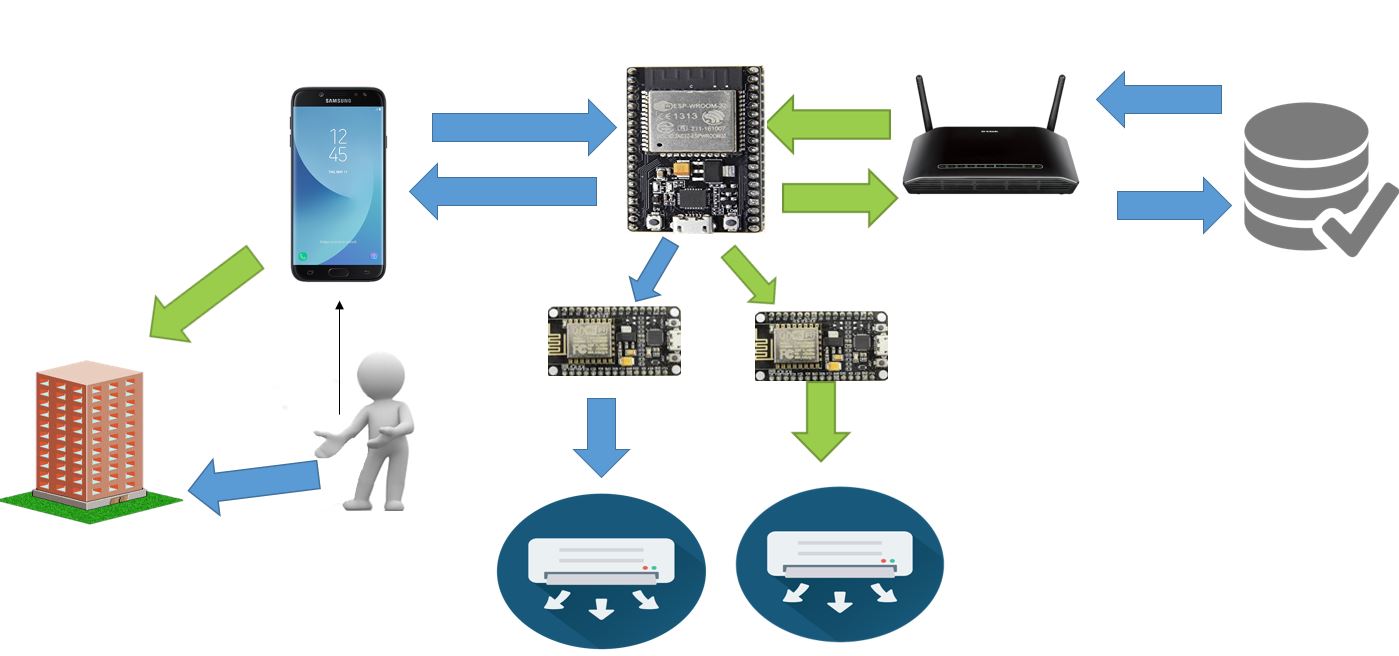
[Notice d'installation 26](#_Toc44000105)

# Présentation du projet

Objectif : Présenter les objectifs principaux

Marche Arrêt Circuit Climatiseur(MACC)

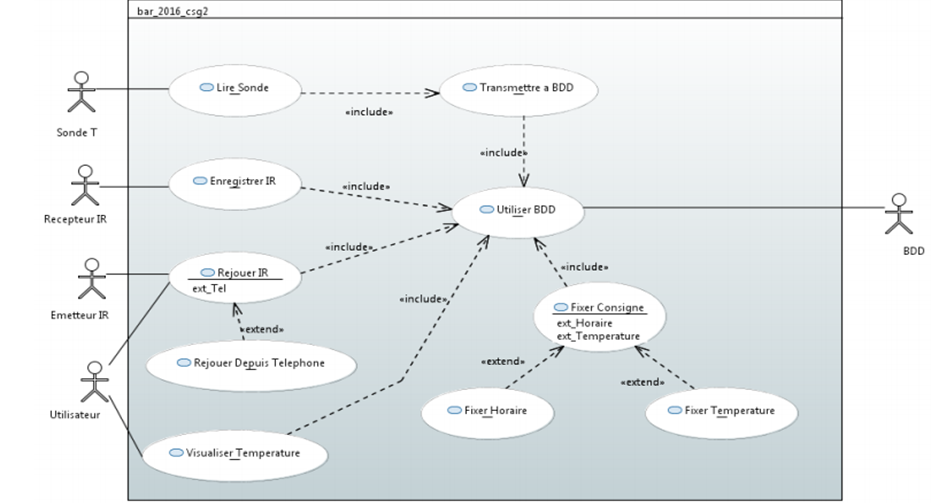
Le système peut être utilisé dans un établissement (hôtels, bureaux, écoles) disposant de nombreuses système de climatisation individuelles disséminée dans des pièces éparses. L’objectif est de réduire les coûts énergétiques attribué à l’oublie de l’arrêt par le personnel des systèmes de climatisation. Une marche ou un arrêt distant sous contrainte, horaire journalier, température ambiante, ou par opérateur direct permettra une diminution sensible du coût des factures énergétiques de l’établissement concerné. Une commande par un smartphone connecté permettra en étant dans la salle concernée de s’affranchir des télécommandes des constructeurs.



# Cahier des charges

|  |  |  |  |
| --- | --- | --- | --- |
| Etudiant 1 | ***CU*** *:Enregistrer IR, Fixer Consigne(ext\_H)*   * Mise en place d’une librairie d’analyse de protocole IR. * Prise en main d’un module de réception IR. * Création d’une BDD associant Message IR aux fonctionnalités d’une télécommande. Marche/arret, Up/down, Timer et climatiseurs associé | Etudiant 3 | ***CU Rejouer IR, Visualiser Temperature***   * Réalisation d’une IHM permettant l’envoie des trames IR à destination d’un ou plusieurs climatiseur.   - Visualisation sélective des relevés des courbes de températures |
| Etudiant 2 | ***CU Lire\_Sonde, Fixer Consigne (ext\_T)***  - Adaptation d’une librairie de lecture de température.  - Récupération de valeur de température pour insertion dans une BDD | Etudiant 4 | CU Rejouer IR  - Réalisation d’une IHM pour appareil mobile. Permettant l’envoie des trames IR pour marche/arrêt, up/down d’un climatiseur |

# Diagramme de Cas d'Utilisation



# 

# Différente tâches

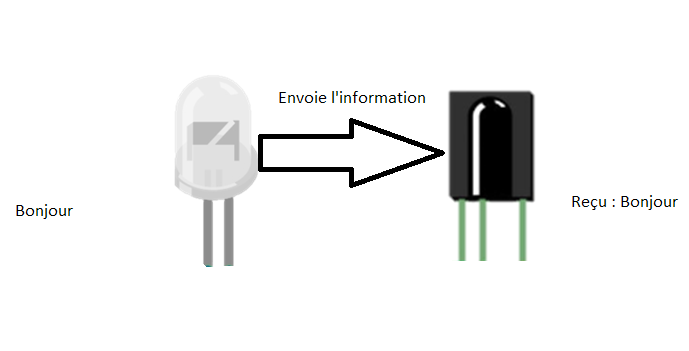
## E3CSG1Macc

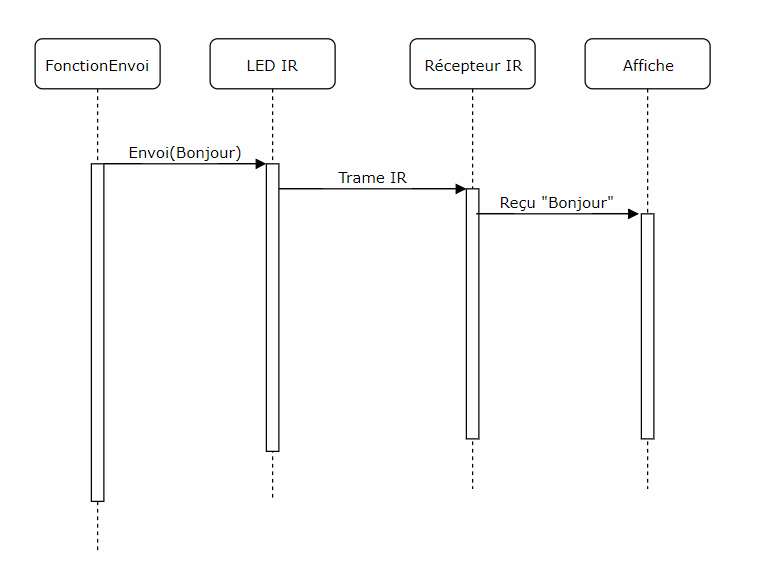
### Envoie de trame IR pour climatiseur

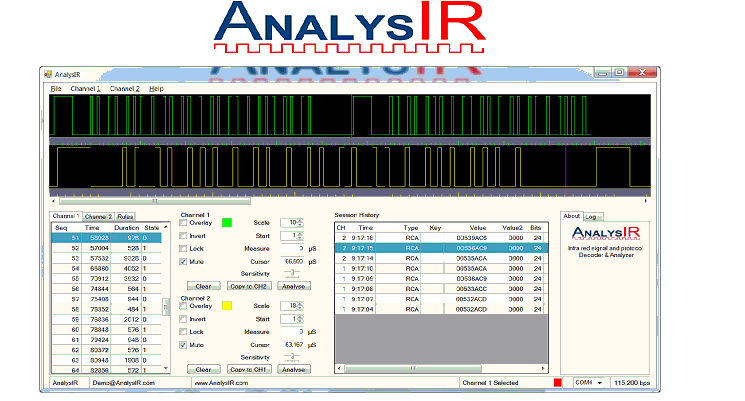
#### Fonctionnement d'une trame IR

Une trame Infrarouge est envoyé par une diode électroluminescente Infrarouge(LED), il est impossible de cette LED s'allumé à l'oeil nu. En effet, l'infrarouge est une onde électromagnétique de fréquence inférieure à celle de la lumière visible peuvent être vu avec un appareil photo s'illuminant tous comme une LED classique.

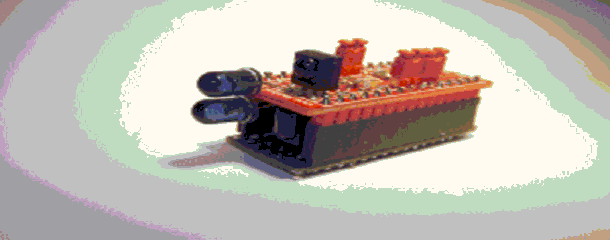
Les appareils recevant le signal (climatiseur, télévision, lecteur DVD...) ont un récepteur pouvant récupéré l'information émise par un rayon infrarouge et vérifie dans à travers sa carte si cette information lui est destinée ou non.







AnalysIR est un software capturant les trames IR reçu, elle permet aussi d’envoyer des trames à travers une carte connectée sous le nom de « A.IR shield » à un arduino NANO. (Voir image ci-dessous)

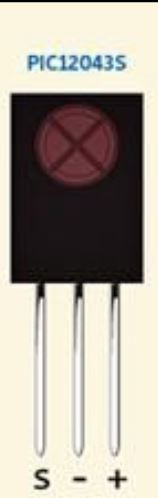


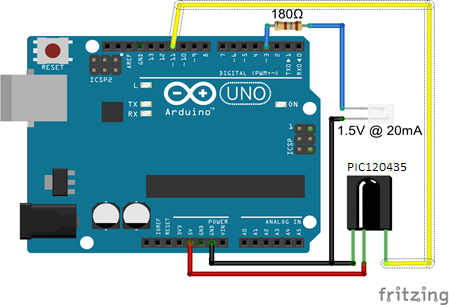
Malheureusement, par manque de licence et une version démonstration pas assez complète, il sera impossible d’utiliser A.IR et AnalysIR.

#### Librairie IRremote et processeur AVR de l'ESP8266

Je rechercha sur internet un moyen d'émettre une Trame infrarouge via une carte de programmation.

La librairie IRremote pour Arduino sur GitHub (<https://github.com/z3t0/Arduino-IRremote>) trouvé lors de la recherche pour utiliser les récepteurs et émetteurs IR. Ci-dessous le câblages d’émetteur et récepteur IR.





Le câblage ci-dessous montre que la sorti du récepteur va dans le connecteur (GPIO) 11, le pôle positif directement sur le 5v de l'arduino et le négatif a terre; pour la LED il faut un résistance de 180 ohms pour limité l'intensité à la sorti du GPIO 3.

Un connecteur GPIO (General Purpose Input/Output) offre à une carte électronique la possibilité de communiquer avec d'autres circuits électroniques ex: un dynamo; une LED; un bouton etc...

Des exemples nous sont Fourni pour une émissions et de réceptions Cependant pour effectuer des test, il nous faut une trame Infrarouge d'un climatiseur en notre possession, nous obtenons une erreur au travers du moniteur série :

"IR code too long. Edit IRremoteInt.h and increase RAWBUF

"

La trame du climatiseurs semble plus longue que la taille prévue par la librairie cela dit il suffit d'aller dans la destination : "\arduino-1.8.5\libraries\IRremote\IRremoteInt.h"

Pour augmenter le buffer à la ligne 43 :

#define RAWBUF 101 // Maximum length of raw duration buffer

Nous passerons la valeur 101 à 255

Après plusieurs essaie, l'arduino ne semble pas avoir trouver l'encodage de la trame mais nous donneras la trame sous sa forme brute.

En utilisant IRsendDemo un exemple présent dans la librairie IrRemote, nous pouvons insérer nos trames encoder ou une trame en brute en l'insérant dans le code en dur.

Suite à cela je téléverse vers l'arduino pour essayer sur le climatiseur, le climatiseur s'alluma pas mais en vérifiant avec un appareil photo, la led emet bien le signal.

Lors de nos cours nous avions intégré l'ESP8266 à arduino IDE pour téléverser vers celui-ci

En essayant de téléverser vers l'ESP8266 il eu une erreur de compilation



Le processeur AVR de l'ESP8266 n'est pas compatible avec la librairie IRremote, donc ne peut pas être téléverser vers celui-ci.

#### IRremote pour ESP8266

En continuant mes recherches sur l'émission infrarouge via ESP8266 sur GitHub la librairie IRremote a été rendu compatible par la communauté avec les ESP8266 et intégrant des encodages pour climatiseurs donc reçois l'intégralité du signal des télécommandes des climatiseurs sans avoir a modifier le Buffer comme précédemment avec IRremote. Cette librairie fut appelée. https://github.com/markszabo/IRremoteESP8266

La librairie Irremote-ESP8266 permet aussi d'envoyer des trames pour les climatiseurs en BRUT ou encoder en prenant comme base l'exemple "IRrecvDemoV2" et émettre un signal IR avec "IRsendDemo".

Voici un décodage de trame de climatiseur avec IrremoteESP8266

IRrecvDumpV2 is now running and waiting for IR input on Pin 14

Timestamp : 000083.750

Encoding : COOLIX

Code : B23F20 (24 bits)

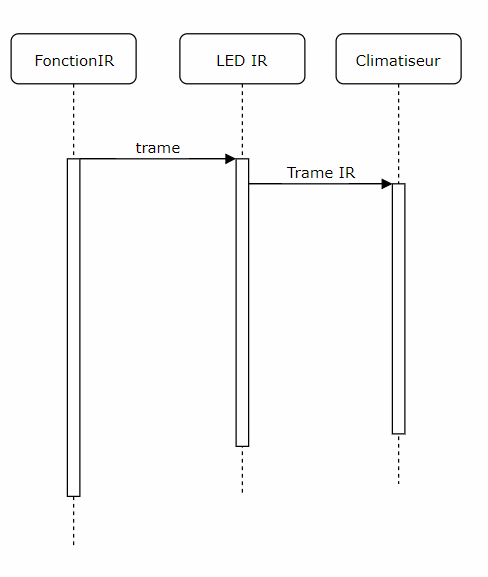
Library : v2.3.2

uint16\_t rawData[199] = {4570, 4466, 556, 1608, 556, 500, 556, 1610, 556, 1610, 558, 498, 556, 500, 556, 1610, 556, 500, 556, 500, 556, 1610, 554, 500, 556, 500, 556, 1610, 554, 1610, 580, 476, 556, 1610, 556, 500, 580, 1584, 554, 1610, 554, 1612, 556, 1608, 556, 500, 554, 1610, 556, 1610, 556, 1610, 554, 500, 556, 500, 554, 502, 556, 500, 554, 1610, 556, 500, 556, 500, 554, 1616, 552, 1610, 556, 1610, 554, 502, 556, 500, 556, 500, 554, 502, 556, 500, 556, 500, 556, 500, 554, 500, 554, 1612, 556, 1610, 556, 1610, 556, 1608, 554, 1610, 556, 5310, 4542, 4490, 556, 1610, 554, 502, 556, 1610, 556, 1610, 554, 502, 554, 502, 554, 1612, 554, 502, 554, 500, 554, 1610, 556, 500, 554, 500, 556, 1608, 556, 1610, 554, 500, 556, 1610, 554, 502, 554, 1612, 554, 1612, 554, 1610, 554, 1610, 554, 500, 554, 1612, 556, 1610, 554, 1610, 554, 504, 552, 502, 554, 502, 556, 500, 554, 1610, 554, 500, 554, 502, 556, 1610, 554, 1610, 554, 1612, 554, 502, 552, 504, 554, 500, 554, 502, 554, 502, 552, 504, 554, 500, 554, 502, 554, 1612, 552, 1612, 530, 1634, 556, 1610, 554, 1612, 552}; // COOLIX B23F20

uint64\_t data = 0xB23F20;

On peut donc voir que le climatiseur utilise l'encodage COOLIX sur 24 bit

Le test devant le climatiseur fonctionna avec l'exemple "IRsendDemo"( code en Annexe n°1)



Je constate qu'à chaque température la trame a un changement, le système de température ne suit pas une simple incrémentation comme pour le volume d'un téléviseur

B23F20(24 bits) 20°c

B23F60(24 bits) 21°c

B23F70(24 bits) 22°c

B23F50(24 bits) 23°c

B23F40(24 bits) 24°c

B23FC0(24 bits) 25°c

### Récupération des trames IR Dans la base de donnée

#### Recherche de librairie pour connexion à la MYSQL

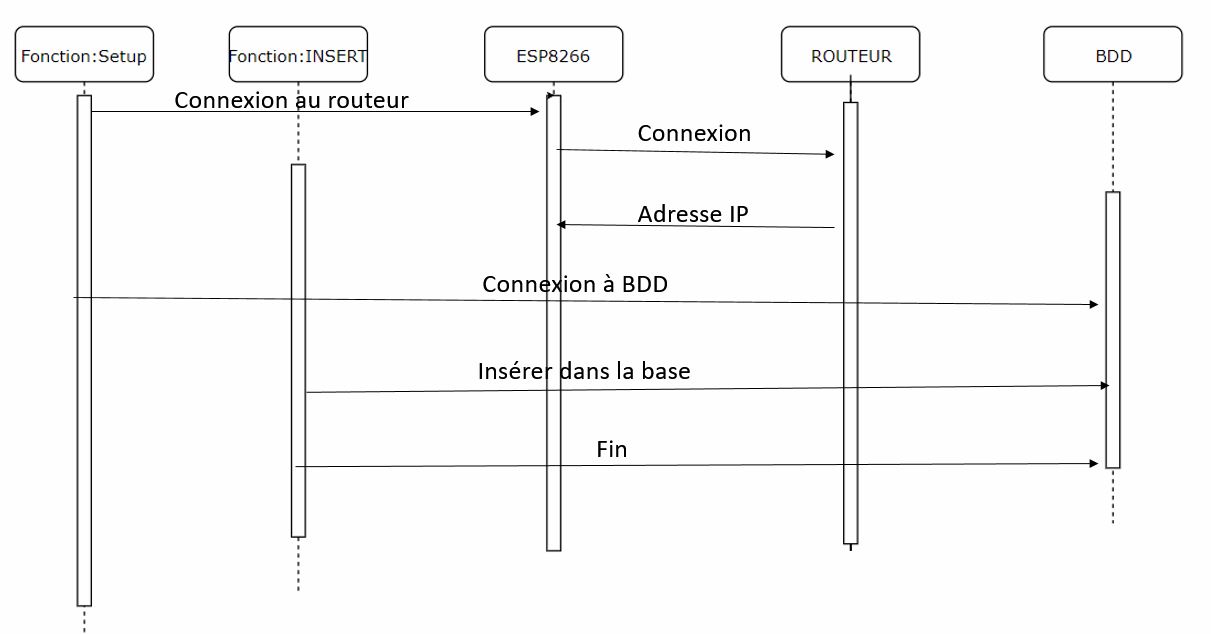
Pour me connecter à la Base de donnée via L'ESP8266, j'émet la supposition qu'il me faudra une librairie permettant les requêtes SQL.

En recherchant dans le gestionnaire de bibliothèque je trouve "MYSQL Connector Arduino" comme son nom l'indique les exemples sont uniquement pour arduino.

Il me fallu vérifié si la librairie peut-être compatible avec ESP8266 en intégrant c'est librairie et la connexion au réseaux.

Je remarque qu'il y a un code pour l'insertion mais aussi un code pour effectuer une lecture dans la base de donnée,

Ci-dessous un diagramme représentant l'insertion:

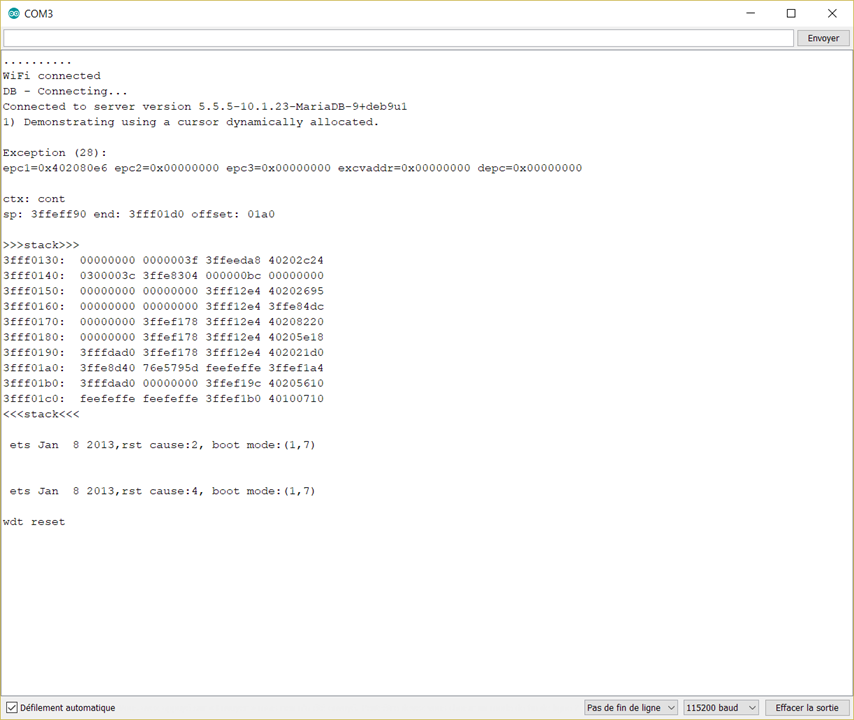


En vérifiant dans la base l'insertion est correctement réalisé

Le code appliqué sera dans Annexe n°2

#### Blocage de L'ESP lors de la lecture de la requête SQL

Pour la lecture du code, ESP8266 se bloque systématiquement lors de la requête SQL



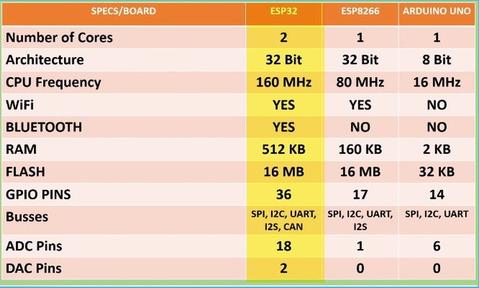
Ceci est très problématique car sans effectuer de lecture à la base de donnée impossible de récupéré les trames dans la base.

#### Lecture de la BDD par L'ESP32(après modification du projet)

Suite à la modification du projet, IHM de salle(ESP32) doit récupérée les valeurs telle que les trames Infrarouges pour les climatiseurs qui seront en tant que client de l'IHM salle.

Les trames infrarouges sont situé dans la base de donnée; il doit donc se connecter à la base de donnée et passée insérér en mémoire vive (dans la RAM).

L'image ci-dessous sont les spécifications de l'ESP32(utilisé pour l'IHM de salle); L'ESP8266(IHM des climatiseurs) et ARDUINO UNO (carte de programmation en C++);



Nous pouvons constatée que L'ESP32 a en sa possession 3 fois plus de Mémoire vive ( 512KB au lieu de 160) se qui nous permettra de stocké beaucoup plus de trame directement en mémoire.

Pour la récupération de Trame dans la base de donnée, la librairie MYSQL\_CONNECTOR disponible dans la bibliothèque Arduino semble ne pas avoir de problème avec l'ESP32.

Nous allons récupéré une trames pour vérifier le fonctionnement. Voir Annexe n°3

#### Conflit entre le TCPSERVER et MYSQL

La connexion à la base de donnée doit être coupé pour utilisé le TCPServer sous peine de ne pas recevoir la réponse de l'application.

Il est nécessaire d'effectuer une déconnections à la base de donnée avec la commande

conn.close();

J'en conclue qu'à cause du TCPServer communicant avec l'application; aucune connexion en temps que client et autre serveur ne pourrais communiqué avec l'ESP8266 sans interrompre le TCPServer avec l'application.

Ceci rend problématique la communication avec les ESP8266 des Climatiseurs car à chaque commande instruction destiné a celle-ci, la communication avec l'application aura lieu!

Pour chaque instruction, l'application devra rétablir le TCPServer.

### Communication entre l'application en Java et l'ESP8266 en C

#### Communication avec ESP8266

L'objectif étant la communication entre ESP8266 pour permettre à l'application d'envoyer des valeurs afin que l'IHM exécute la demande pour en mettre une trame IR en fonction de la valeur réceptionner.

Dans nos connaissances acquis lors des années, nous avons appris en language JAVA,(language utilisé pour l'application).

les Sockets permette de communiqué entre un serveur(l'ESP) et un client(l'application).

Les sockets dans le language Arduino sont assez différente du language java :

#include <ESP8266WiFi.h>

#include <WiFiClient.h>

// Hardcode WiFi parameters as this isn't going to be moving around.

const char\* ssid = "<>";

const char\* password = "<>";

// Start a TCP Server on port 5045

WiFiServer server(5045);

void setup() {

Serial.begin(115200);

WiFi.begin(ssid,password);

Serial.println("");

//Wait for connection

while(WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.print("Connected to "); Serial.println(ssid);

Serial.print("IP Address: "); Serial.println(WiFi.localIP());

// Start the TCP server

server.begin();

}

void loop() {

TCPServer();

}

void TCPServer () {

WiFiClient client = server.available();

if (client) {

if (client.connected()) {

Serial.println("Connected to client");

}

if (client.available() > 0) {

// Read incoming message

char inChar = client.read();

// Send back something to the clinet

server.write(inChar);

// Echo input on Serial monitor

Serial.write(inChar);

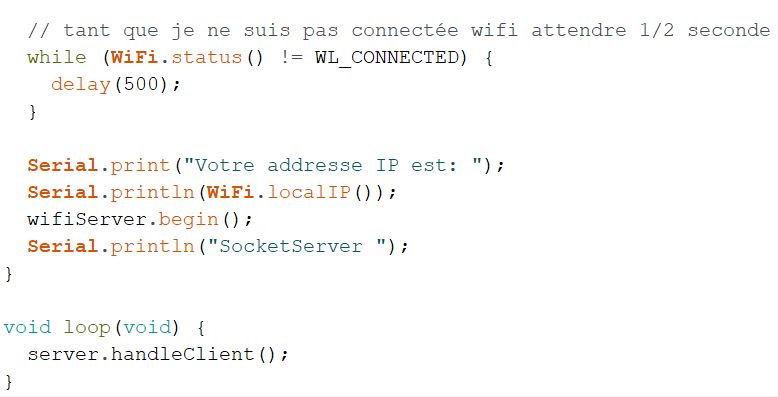
}

L'application envoie le socket mais il n'y pas de réponse:

Après quelque recherche, il semblerait qu'il y a une méthode passant par un serveur web situer sur l'IHM, l'application se connectera et passant des paramètres, exemple : https://192.168.1.6/ir?climV13=allume







Cette méthode est la plus courante pour la communication entre les applications Android et le language Arduino.

#### Modification Projet

Un problème ayant été soulevé, les climatiseurs peuvent être contrôlé par des personnes ne se trouvant pas dans la salle dû a sa connections sur le wifi.

Nous devons utiliser ESP32 pour connecter les ESP8266 équipé d'un module infrarouge par climatiseurs nécessite qu'une seule connexion du smartphone à l'ESP32 pour contrôler les climatiseurs de salle et limité la porté du WiFi.

Cela veut donc dire que l'ESP32 devra se faire office de routeur pour laissé le smartphone se connectée à a la base de donnée ainsi que les ESP8266.

#### Création de la Fonction pour la récupération de valeur dans la base de donnée

Étant passer à ESP32, il devra récupéré des valeurs dans la base de donnée et mettre ses valeurs en mémoire, dans une de ses variables pour évité de demander à la base de donnée la même information.

Plusieurs variable seront demander, il est donc nécessaire d'utilisé le code de lecture dans une fonction avec au moins un paramètre et une sortie pour stocker la valeur.

#include <WiFi.h>

#include <MySQL\_Connection.h>

#include <MySQL\_Cursor.h>

#include <esp\_event.h>

#include <esp\_event\_loop.h>

const char\* ssidWIFI = "NOM-DU-WIFI";

const char\* pWIFI = "MOTDEPASSE";

IPAddress server\_addr(192, 168, 137, 127); // IP of the MySQL \*server\* here

char user[] = "pi"; // MySQL user login username

char password[] = "Simconolat"; // MySQL user login password

char EXEC\_SQL[128]; // Buffer de creation requete

char SELECT\_SQL[] = "SELECT message FROM testsSnir\_2018.test\_arduino WHERE client=%d";

int cli=0;

char\* sel;

WiFiClient client;

MySQL\_Connection conn((Client \*)&client);

byte mac\_addr[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };

// Create an instance of the cursor passing in the connection

void setup() {

Serial.begin(115200);

delay(10);

// Connect to WiFi network

Serial.println();

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssidWIFI);

WiFi.mode(WIFI\_STA);

WiFi.begin(ssidWIFI, pWIFI);

WiFi.setHostname("ESP\_Salle");

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

Serial.println("DB - Connecting...");

//Attente de connexion a la BDD

while (conn.connect(server\_addr, 3306, user, password) != true) {

delay(500);

Serial.print ( "." );

}

delay(1000);

sprintf(EXEC\_SQL, SELECT\_SQL, cli);

sel = SELECT(EXEC\_SQL); // exécute la fonction SELECT

}

void loop() {

delay(10000);

Serial.println(sel);

}

char\* SELECT(char\* lol)

{

// Serial.println("> Execution de SELECT with dynamically supplied parameter");

MySQL\_Cursor \*cur\_mem = new MySQL\_Cursor(&conn);

// preparer le code sql

// Execute the query

cur\_mem->execute(lol);

// Serial.println(EXEC\_SQL);

// Fetch the columns and print them

column\_names \*cols = cur\_mem->get\_columns();

// Serial.print("ColName:");

for (int f = 0; f < cols->num\_fields; f++) {

// Serial.print(cols->fields[f]->name);

if (f < cols->num\_fields - 1) {

// Serial.print(',');

}

}

// Serial.println();

// Read the rows and print them

row\_values \*row = NULL;

do {

row = cur\_mem->get\_next\_row();

if (row != NULL) {

for (int f = 0; f < cols->num\_fields; f++) {

if (f == 0) {

// Serial.print("\t");

}

char\* test = row->values[f];

//Serial.println(test);

return test;

if (f < cols->num\_fields - 1) {

// Serial.print(',');

}

}

// Serial.println();

}

}

while (row != NULL);

// Deleting the cursor also frees up memory used

delete cur\_mem;

}

#### Conversion de trame IR en décimal et abandon de la communication en WEBServer sur IHM Des climatiseurs

Les trames Infrarouges sont en hexadécimal mais à-travers le lien HTTP il a fallu effectuée une conversion en décimal afin de pouvoir transféré la trame infrarouge

Par exemple, la trame "0xB27BE0" deviendra "178123224".

Mais pour des trames trop longue, les trames sont trop longue et sont en puissance de 10 et sont beaucoup trop longue pour passer a travers l'url.

LA METHODE EST ABANDONÉE.

Nous faisons donc un TCPClient et Serveur sur l'ESP SALLE pour recevoir les valeurs de l'application et envoyée la trame à l'ESP CLIM; Le ESP CLIM aura donc un TCPServer actif en permanence pour recevoir les Trames IR de l'ESP SALLE.

Voici le code pour ESP-CLIM: Voir Annexe n°4

### Identification de L'ESP

#### Reconnaissance de la salle

Cette partie reconnaissance avait été prévue pour localisé l'ESP8266 pour l'application l'ajout de l'ESP32.

pour listé les ESP32 situés dans les Bâtiments, j'ai eu l'idée de faire une insertion dans la base de donnée comprenant l'adresse MAC comme clé primaire et d'indiqué la salle et le bâtiment ainsi que son Adresse IP lors du téléversement.

Le code se trouve dans Annexe n°4

### Visualisation Courbe de Sonde

La Visualisation de la courbe se fait sur la sonde, elle contient les 4 températures ambiante prélevés dans la salle, d'une intervalle de 15 min chacun.

ses valeurs sont récupérée dans la base de Donnée et stockée par l'IHM salle contenant un capteur de température qui sera pour notre projet, le DHT22

### Annexe

#### Annexe n°1

#ifndef UNIT\_TEST

#include <Arduino.h>

#endif

#include <IRremoteESP8266.h>

#include <IRsend.h>

IRsend irsend(4); // LED IR est contrôlée par GPIO pin 4 (D2)

void setup() {

irsend.begin();

Serial.begin(115200, SERIAL\_8N1, SERIAL\_TX\_ONLY); //démarre le moniteur série

}

void loop() {

#if SEND\_COOLIX

Serial.println("allume le climatiseur à 20°C");

irsend.sendCOOLIX(0xB23F20, 24, 2); //trame IR, 24 bit, répété 2 le signal

#endif // SEND\_COOLIX

delay(2000);

}

#### Annexe n°2

#include <MySQL\_Connection.h>

#include <MySQL\_Cursor.h>

#include <ESP8266WiFi.h>

#include <WiFiClient.h>

char ssidWIFI[] = "SSID";

char passwordWIFI[] = "PASSWORD;

char nomHOST[] = "MACC IR";

IPAddress server\_addr(192,168,1,1); // IP of the MySQL \*server\* here

char user[] = "pi"; // MySQL user login username

char password[] = "Simconolat"; // MySQL user login password

// Sample query

char query[] = "insert to bddtest";

WiFiClient client;

MySQL\_Connection conn((Client \*)&client);

void setup() {

Serial.begin(115200);

delay(10);

// Connect to WiFi network

Serial.println();

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssidWIFI);

WiFi.mode(WIFI\_STA);

WiFi.begin(ssidWIFI, passwordWIFI);

WiFi.hostname(nomHOST);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

Serial.println("DB - Connecting...");

//Attente de connexion a la BDD

while (conn.connect(server\_addr, 3306, user, password) != true) {

delay(500);

Serial.print ( "." );

}

};

void loop() {

delay(2000);

Serial.println("\nRunning SELECT and printing results\n");

// Initiate the query class instance

MySQL\_Cursor \*cur\_mem = new MySQL\_Cursor(&conn);

// Execute the query

cur\_mem->execute(query);

// Fetch the columns and print them

column\_names \*cols = cur\_mem->get\_columns();

for (int f = 0; f < cols->num\_fields; f++) {

Serial.print(cols->fields[f]->name);

if (f < cols->num\_fields-1) {

Serial.print(", ");

}

}

Serial.println();

// Read the rows and print them

row\_values \*row = NULL;

do {

row = cur\_mem->get\_next\_row();

if (row != NULL) {

for (int f = 0; f < cols->num\_fields; f++) {

Serial.print(row->values[f]);

if (f < cols->num\_fields-1) {

Serial.print(", ");

}

}

Serial.println();

}

} while (row != NULL);

// Deleting the cursor also frees up memory used

delete cur\_mem;

}

#### Annexe n°3

#include <WiFi.h>

#include <MySQL\_Connection.h>

#include <MySQL\_Cursor.h>

#include <esp\_event.h>

#include <esp\_event\_loop.h>

const char\* ssidWIFI = "DESKTOP-GLPMSI";

const char\* pWIFI = "97197170";

IPAddress server\_addr(192, 168, 1, 8); // Addresse de la BDD

char user[] = "pi"; // IDENTIFIANT

char password[] = MDP"; // mot de passe

char EXEC\_SQL[128]; // Buffer de creation requete

char SELECT\_SQL[] = "SELECT message FROM testsSnir\_2018.test\_arduino WHERE client=%d";

int cli=0;

char\* sel;

WiFiClient client;

MySQL\_Connection conn((Client \*)&client);

byte mac\_addr[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };

// Create an instance of the cursor passing in the connection

void setup() {

Serial.begin(115200);

delay(10);

// Connexion au WiFi

Serial.println();

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssidWIFI);

WiFi.mode(WIFI\_STA);

WiFi.begin(ssidWIFI, pWIFI);

WiFi.setHostname("ESP\_Salle");

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

Serial.println("DB - Connecting...");

//Attente de connexion a la BDD

while (conn.connect(server\_addr, 3306, user, password) != true) {

delay(500);

Serial.print ( "." );

}

delay(1000);

}

void loop() {

delay(10000);

MySQL\_Cursor \*cur\_mem = new MySQL\_Cursor(&conn);

// preparer le code sql

sprintf(EXEC\_SQL, SELECT\_SQL, cli);

// Execute the query

cur\_mem->execute(EXEC\_SQL);

// Serial.println(EXEC\_SQL);

// Fetch the columns and print them

column\_names \*cols = cur\_mem->get\_columns();

// Serial.print("ColName:");

for (int f = 0; f < cols->num\_fields; f++) {

// Serial.print(cols->fields[f]->name);

if (f < cols->num\_fields - 1) {

// Serial.print(',');

}

}

// Serial.println();

// Read the rows and print them

row\_values \*row = NULL;

do {

row = cur\_mem->get\_next\_row();

if (row != NULL) {

for (int f = 0; f < cols->num\_fields; f++) {

if (f == 0) {

// Serial.print("\t");

}

char\* test = row->values[f];

Serial.println(test);

if (f < cols->num\_fields - 1) {

// Serial.print(',');

}

}

// Serial.println();

}

}

while (row != NULL);

// Deleting the cursor also frees up memory used

delete cur\_mem;

}

#### Annexe n°4

#include <WiFi.h>

#include <MySQL\_Connection.h>

#include <MySQL\_Cursor.h>

#include <esp\_event.h>

#include <esp\_event\_loop.h>

const char\* ssidWIFI = "SSID";

const char\* pWIFI = "PASSWORD";

int SALLE = 13; // NUMERO DE LA SALLE

char\* BATIMENT ="BTF"; // BATIMENT

uint8\_t MAC\_array[6];

char MAC\_char[18];

IPAddress server\_addr(192, 168, 1, 1); // IP du serveur MySQL

char user[] = "pi"; // utilisateur BDD

char password[] = "password"; // mot de passe BDD

char EXEC\_SQL[128]; // Buffer de creation requete

char EXE\_SQL[128];

char INSERT\_SQL[] = "INSERT INTO `MACC1`.`ESPSALLE` (`NUM\_SALLE`, `IP\_CLIM`,`ADRESSE\_MACESP`, `NOM\_BAT`) VALUES ('%d', '%s', '%s', '%s');";

char UPDATE\_SQL[] = "UPDATE `MACC1`.`ESPSALLE` SET `IP\_CLIM`='%s' WHERE `ADRESSE\_MACESP`='%s';";

WiFiClient client;

MySQL\_Connection conn((Client \*)&client);

void setup() {

Serial.begin(115200);

delay(10);

// Connect to WiFi network

Serial.println();

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssidWIFI);

WiFi.mode(WIFI\_STA);

WiFi.begin(ssidWIFI, pWIFI);

WiFi.setHostname("ESP\_Salle");

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

Serial.println("DB - Connecting...");

//Attente de connexion a la BDD

while (conn.connect(server\_addr, 3306, user, password) != true) {

delay(500);

Serial.print ( "." );

}

WiFi.macAddress(MAC\_array);

for (int i = 0; i < sizeof(MAC\_array); ++i){

sprintf(MAC\_char,"%s%02x:",MAC\_char,MAC\_array[i]);

}

Serial.println(MAC\_char);

Serial.println(WiFi.localIP());

sprintf(EXEC\_SQL, INSERT\_SQL, SALLE, WiFi.localIP().toString().c\_str(), MAC\_char, BATIMENT);

Serial.println(EXEC\_SQL);

INSERT(EXEC\_SQL); //rajoute dans table ESPSALLE la carte si cela est sa première connexion

delay(500);

Serial.println(MAC\_char);

sprintf(EXE\_SQL, UPDATE\_SQL, WiFi.localIP().toString().c\_str(), MAC\_char);

Serial.println(EXE\_SQL);

}

void loop() {

delay(900000); // attendre 15 min

INSERT(EXE\_SQL); //met a jour l'adresse IP dans la base de donnée

}

void INSERT(char\* requete)

{

// Initialise l'instance de la classe de requête

MySQL\_Cursor \*cur\_mem = new MySQL\_Cursor(&conn);

// Execute the query

cur\_mem->execute(requete);

delete cur\_mem;

//comme il n'y a aucun résultat, nous n'avons pas besoin de lire de données

   // La suppression du curseur libère également la mémoire utilisée

}

### Notice d'installation

#### Instruction