

**SG01 – MakAir**

*Laurent Johan*

**Dossier technique du projet – partie individuelle**

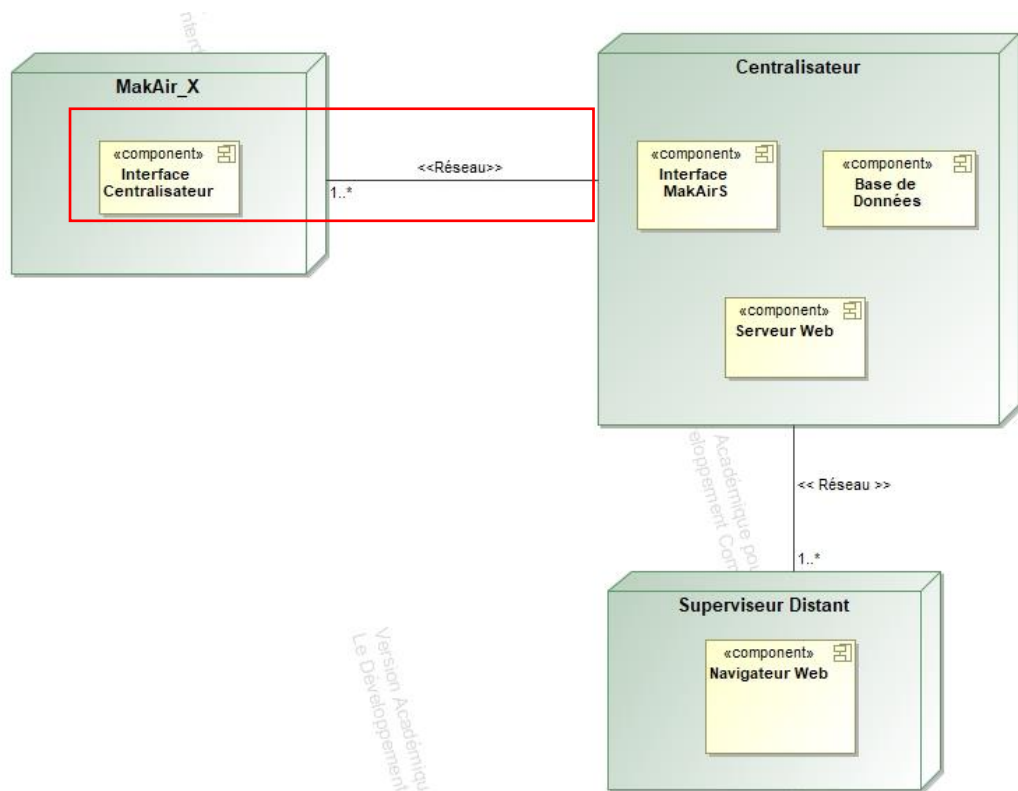
<b>I - SITUATION DANS LE PROJET .....</b>	<b>2</b>
I.1 - INTRODUCTION .....	2
I.2 - SYNOPSIS DE LA REALISATION .....	4
<b>II - REALISATION DE LA FONCTION DU CAS D'UTILISATION UCE1-« COLLECTER LES DONNEE » .....</b>	<b>5</b>
II.1 - CONCEPTION DETAILLEE .....	5
II.1.1 - Mise en place de la solution « minicom ».....	5
II.2 - TESTS UNITAIRES.....	10
II.2.1 - Test unitaire de la récupération de données .....	10
II.2.2 - Test unitaire de la méthode « désérialiser » .....	10
II.2.3 - Problèmes rencontrés.....	11
<b>III - REALISATION DE LA FONCTION DU CAS D'UTILISATION UCE3-« ENVOYER PERIODIQUEMENT LES DONNEES » .....</b>	<b>11</b>
III.1 - CONCEPTION DETAILLEE .....	11
III.2 - TESTS UNITAIRES.....	19
III.2.1 - Test unitaire de la méthode « connexionALaBDD » .....	19
III.2.2 - Problèmes rencontrés.....	19
<b>IV - BILAN DE LA REALISATION PERSONNELLE .....</b>	<b>20</b>
IV.1 - PARTIES RESTANT A DEVELOPPER .....	20
IV.2 - CONCLUSION PERSONNELLE .....	20

## I - Situation dans le projet

### I.1 - Introduction

- *Au sein du projet je dois gérer l'UCE1 (Collecter les données) et l'UCE3 (Envoyer périodiquement les données). L'une des contraintes que j'ai est celle du temps réel. Donc les données doivent s'envoyer sur une durée de l'ordre de la seconde. De manière automatique et complètement autonome, le MakAir collecte et envoie les données de fonctionnement ainsi que celles relatives au patient.*

UC	Description	Critères	Niveau	Flexibilité
<b>UCE1</b>	Collecter les données	Les données sont correctement collectées	N1	F0
<b>UCE2</b>	Enregistrer MakAir	Le MakAir a correctement été enregistré sur le Centralisateur	N1	F0
<b>UCE3</b>	Envoyer périodiquement les données.	Les données sont envoyées périodiquement (à la période T)	N1	F0



*Diagramme de déploiement du système à concevoir*

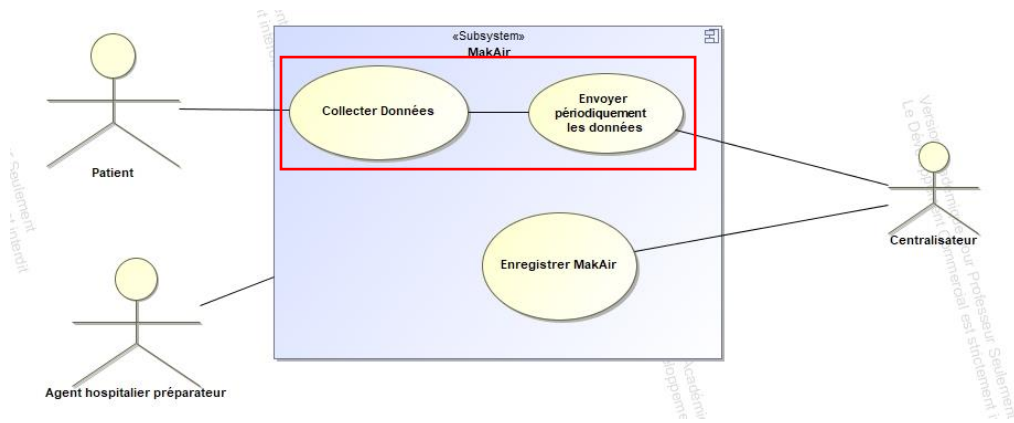


Diagramme des cas d'utilisations (système embarqué)

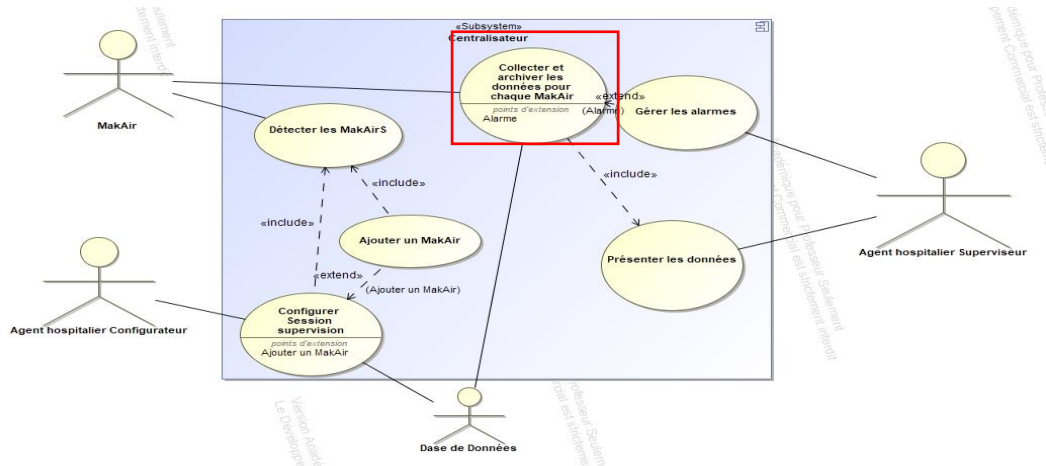
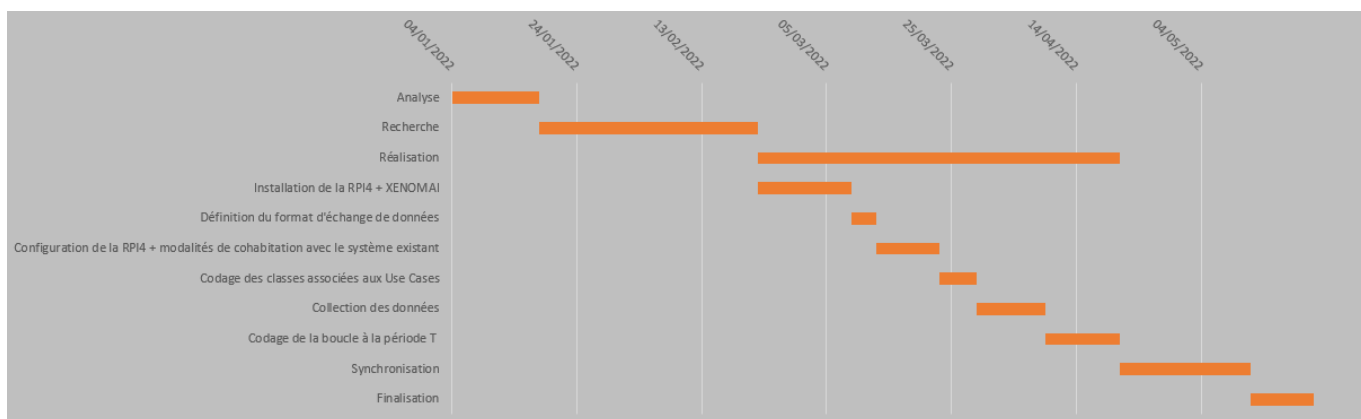


Diagramme de cas d'utilisations (centralisateur)



Gantt des tâches personnelles du projet

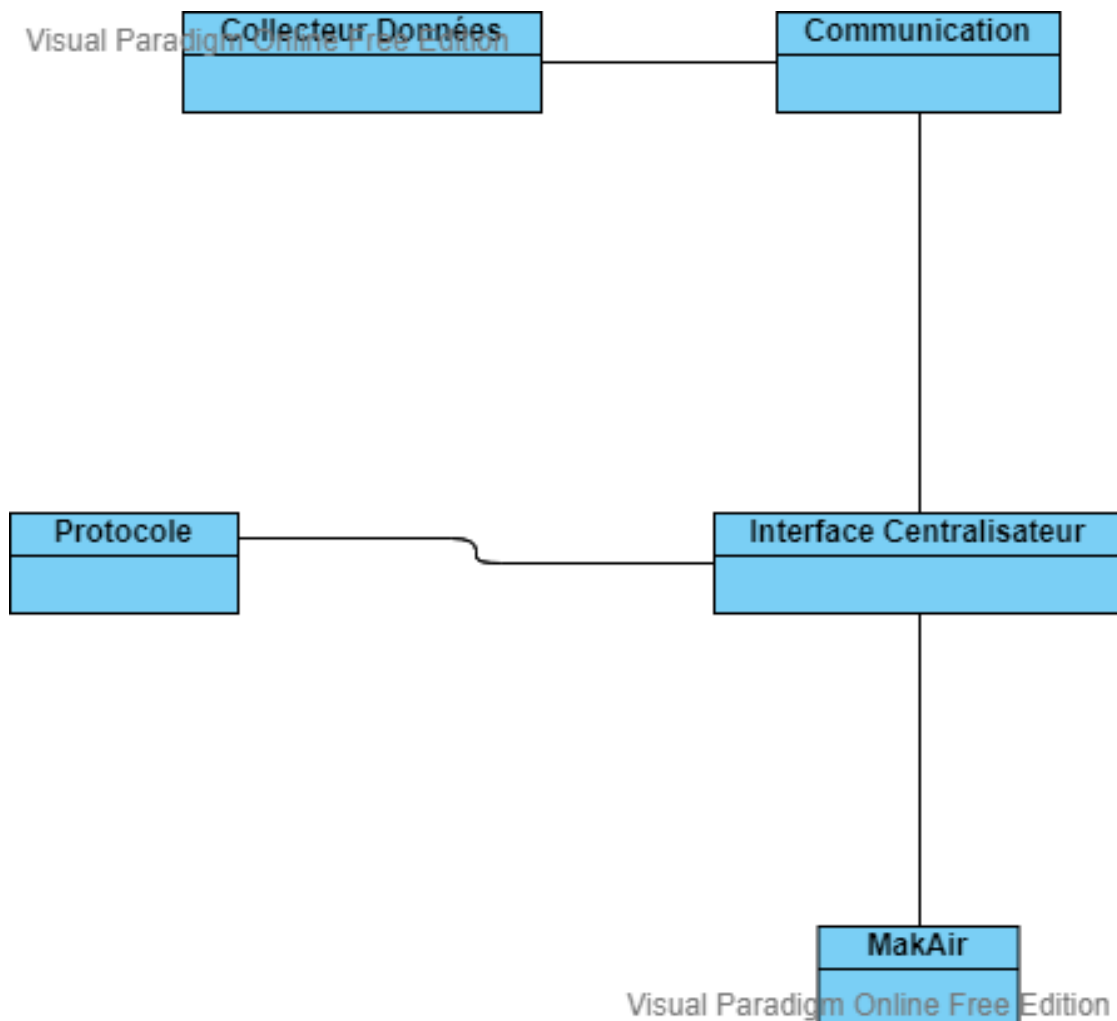
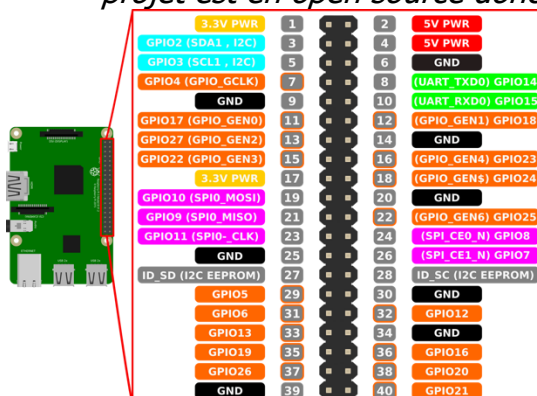


Diagramme de classe (Embarqué)

### I.2 - Synoptique de la réalisation

*Pour la liaison série, on a dû se pencher sur la fiche technique de la RPI 4 et la documentation technique du MakAir que l'on a trouvé sur le git hub étant donné que le projet est en open source donc ouvert à tous. Voici ce que l'on a trouvé :*



*On a donc soudé les broches 7, 14 pour la masse et 29.*

## **II - Réalisation de la fonction du cas d'utilisation UCE1-**

### **« Collecter les donnée »**

- *Ce cas d'utilisation consiste à récupérer des données qui sont situés dans le MakAir. J'ai tout d'abord choisi de travailler en SSH j'utilise ce protocole pour travailler sur la RPI à distance pour ne pas déranger les autres membres du groupe. Ensuite se trouvait le choix entre le TCP et l'UDP l'avantage du TCP est la phase de connexion entre les deux adresses IP cela donne une sécurité d'avoir la totalité des informations, l'avantage de l'UDP est la rapidité d'envoi pour avoir essayé les deux, j'ai préféré choisir le protocole TCP car il n'agit pas sur la contrainte qui m'était donnée qui est l'envoi des données en temps réel et cela me donne une sécurité en plus par rapport à l'UDP. J'ai travaillé en C++ car cela était une contrainte donnée dans mon cahier des charges.*

### II.1 - Conception détaillée

#### II.1.1 - Mise en place de la solution « minicom »

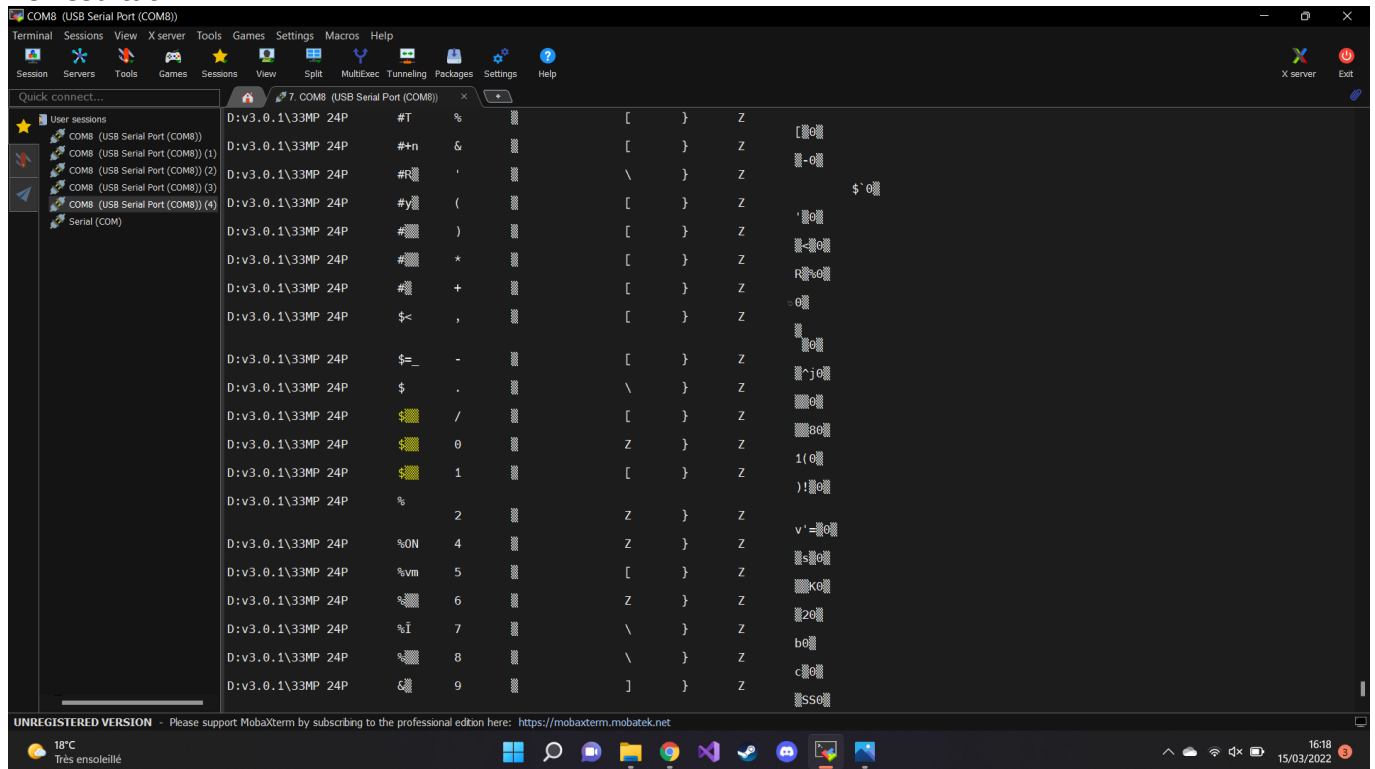
- *Tout d'abord j'ai dû commencer par installer minicom sur ma RPI 4 la procédure est la suivante :*

```
sudo apt-get install minicom
```

*Ensuite pour vérifier le bon fonctionnement de la liaison série il me suffisait de faire la commande suivante :*

*Sudo minicom -d/dev/ttyUSB0*

*Donc sudo pour avoir l'accès aux commandes de superutilisateur. Puis le -d nous sert à préciser le port ici /dev/ttyUSB0.*

*Le résultat :*

*Après cela je savais que je recevais bien les données seulement elles n'étaient pas lisibles. Je me suis donc renseigné sur le GitHub du MakAir puis j'ai trouvé cela :*

```
void sendDataSnapshot(uint16_t centileValue,
                      int16_t pressureValue,
                      CyclePhases phase,
                      uint8_t blowerValvePosition,
                      uint8_t patientValvePosition,
                      uint8_t blowerRpm,
                      uint8_t batteryLevel,
                      int16_t inspiratoryFlowValue,
                      int16_t expiratoryFlowValue) {
    uint8_t phaseValue;
    if (phase == CyclePhases::INHALATION) {
        phaseValue = 17u; // 00010001
    } else if (phase == CyclePhases::EXHALATION) {
        phaseValue = 68u; // 01000100
    } else {
        phaseValue = 0u;
    }

    Serial6.write(header, HEADER_SIZE);
    CRC32 crc32;
    Serial6.write("D:", 2);
    crc32.update("D:", 2);
    Serial6.write((uint8_t)PROTOCOL_VERSION); // Communication protocol version
    crc32.update((uint8_t)PROTOCOL_VERSION);

    Serial6.write(static_cast<uint8_t>(strlen(VERSION)));
    crc32.update(static_cast<uint8_t>(strlen(VERSION)));
    Serial6.print(VERSION);
    crc32.update(VERSION, strlen(VERSION));
    Serial6.write(deviceId, 12);
    crc32.update(deviceId, 12);

    Serial6.print("\t");
    crc32.update("\t", 1);

    byte systick[8]; // 64 bits
    toBytes64(systick, computeSystick());
    Serial6.write(systick, 8);
    crc32.update(systick, 8);

    Serial6.print("\t");
    crc32.update("\t", 1);
}
```

*On peut donc voir ici les valeurs qui se transmettent, les valeurs sont donc :*

- centileValue (valeur en pourcentage)
- pressureValue (valeur de la pression)
- phase
- blowerValvePosition (position de la vanne de soufflage)
- patientValvePosition (position de la vanne du patient)
- blowerRpm (souffleur en rythme par minute)
- batteryLevel (niveau de la batterie)
- inspiratoryFlowValue (Valeur du débit inspiratoire)
- expiratoryFlowValue ((Valeur du débit expiratoire)

*Suite à cela j'ai donc pu coder mon tableau avec le nombre de valeurs besoin comme ceci :*

```
void Communication::sendToDb(string Tab[10])
{
    try {
        sql::ResultSet *res;
        sql::Statement *stmt;
        stmt = con->createStatement();

        // COMMANDE SQL
        unsigned int sizeOfTable = 0;

        res = stmt->executeQuery("SELECT COUNT(*) FROM Datas");
        while (res->next()) {
            cout << res->getInt(1) << endl;
            sizeOfTable = res->getInt(1);
        }

        if (sizeOfTable < 150) {
            const string comQuery = "(idMakair, centileValue, pressureValue, phase ,blowerValvePosition ,patientValvePosition, blowerRpm, batteryLevel, inspiratoryFlowValue, expiratoryFlowValue ) ";
            const string values = "VALUES(" + Tab[0] + "," + Tab[1] + "," + Tab[2] + "," + Tab[3] + "," + Tab[4] + "," + Tab[5] + "," + Tab[6] + "," + Tab[7] + "," + Tab[8] + "," + Tab[9] + ")";
        }
    }
}
```



### codeDeserialiser.txt

#### codeDeserialiser.txt

```
1  /**
2   * Convert an array of 2 bytes to a u16
3   *
4   * @param bytes Array of 2 elements
5   * @return The corresponding u16 number
6   */
7   // cppcheck-suppress unusedFunction
8   uint16_t toU16(byte bytes[]) {
9       uint16_t num = (bytes[0] << 8) + bytes[1];
10      return num;
11  }
12
13  /**
14   * Convert an array of 4 bytes to a u32
15   *
16   * @param bytes Array of 4 elements
17   * @return The corresponding u32 number
18   */
19  // cppcheck-suppress unusedFunction
20  uint32_t toU32(byte bytes[]) {
21      uint32_t num = (bytes[0] << 24) + (bytes[1] << 16) + (bytes[2] << 8) + bytes[3];
22      return num;
23  }
```

*Ceci est le code pour désérialiser la trame, je l'ai donc pris en l'état et l'ai inséré dans mon code. En ajoutant une variable « trameTest » et en lui assignant une trame que j'ai copier-coller.*



```
int main()
{
    Communication client;

    string frameTest = "2.00AD:v3.0.1\33MP 24P 0yi! ¥ ^ D } ^ d";
    //string lastFrame;

    unsigned int cursor = 0;
```

*Voici le résultat obtenu :*

```
192.168.114.100
Terminal Sessions View X server Tools Games Settings Macros Help
Quick connect...
home/ubuntu
Name
.ccache
.local
.csh
.c++
.counter
.doc
.doccounter
.pstTest
.projects
.rtsid
.testspopen
.testKOP
.testKOP
.rtsid
.bash_history
.bash_logout
.bashrc
.profile
.sudo_as_admin_successful
.xauthority
.monscript
.frame.sh
.frame.txt
Remote monitoring
Follow terminal folder
idValue = P 24P
Systick = 24P
Value at [0] =
Result at [1] uint16_t = 0
Value at [1] =
Result at [2] int16_t = 8192
Value at [2] = 2
Result at [3] uint8_t = 50
Value at [3] = 4
Result at [4] uint8_t = 52
Value at [4] = P
Result at [5] uint8_t = 80
Value at [5] =
Result at [6] uint8_t = 140
Value at [6] =
Result at [7] uint8_t = 255
Value at [7] =
Result at [8] int16_t = -4608
Value at [8] =
Result at [9] int16_t = 8448
Value at [9] =
Value at [10] =
Value at [11] = D
Value at [12] = }
Value at [13] = ^
```

II.2 - Tests unitairesII.2.1 - Test unitaire de la récupération de données

Élément testé :	Test récupération données			
Objectif du test :	Test si on récupère bien les données			
Nom du testeur :	Johan		Date :06/04/22	
Moyens mis en œuvre :	Logiciel : moba Xterm	Matériel : RPI, makAir	Outil de développement : aucun	
Procédure du test : Vérifier grâce à la commande « minicom –device ttyUSB0 »				
Id	Description du vecteur de test	Résultat attendu	Résultat obtenu	Validation (O/N)
	Tester la récupération de données avec le port liaison série bien branché	Lisibilité des trames	Lisibilité des trames	O
	Tester la récupération de données sans le port liaison série branché	Non lisibilité des trames	Non lisibilité des trames	O
Conclusion du test :		Le test est concluant.		

II.2.2 - Test unitaire de la méthode « désérialiser »

Élément testé :	Test correspondance données			
Objectif du test :	Tester si la désérialisation fonctionne avec un trame test			
Nom du testeur :	Johan		Date :06/04/22	
Moyens mis en œuvre :	Logiciel : Visual studio	Matériel : RPI, trame test	Outil de développement : Visual studio	
Procédure du test : Vérifier grâce à la commande « minicom –device ttyUSB0 »				
Id	Description du vecteur de test	Résultat attendu	Résultat obtenu	Validation (O/N)
	Tester la correspondance avec une bonne trame test	Listing avec correspondance des valeurs de la trame	Listing avec correspondance des valeurs de la trame	O
	Tester la correspondance avec une mauvaise trame test	Message erreur	Pas de message erreur	N
Dossier technique Johan Laurent			23 mai 22	Page 10

Conclusion du test :	Il faut coder un message d'erreur si la trame est mauvaise
----------------------	--

### II.2.3 - Problèmes rencontrés

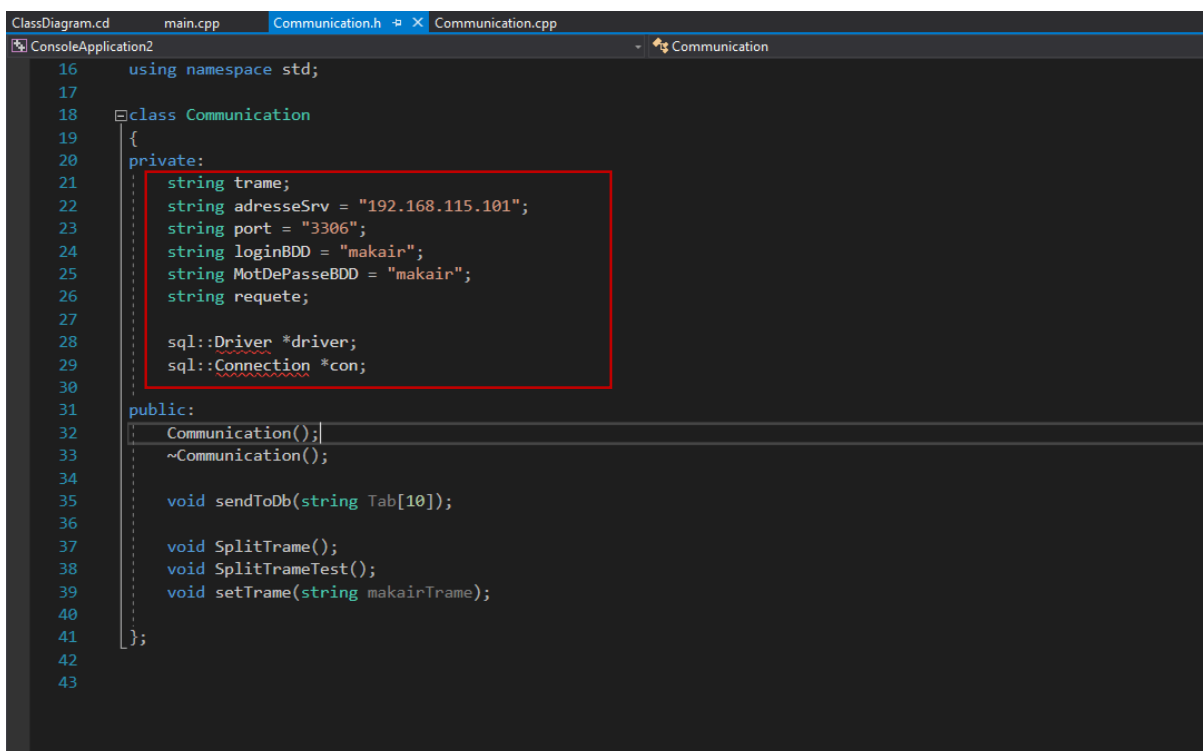
*J'ai rencontré un problème sur la liaison série qui n'était pas la bonne, j'avais pris la liaison série qui se situe or du MackAir. La solution était donc de faire la soudure comme dit précédemment.*

## **III - Réalisation de la fonction du cas d'utilisation UCE3- « Envoyer périodiquement les données »**

*Dans ce cas d'utilisation, le principe est simple une fois la récupération de données effectuer, je dois les envoyer périodiquement à la base de données. Cependant, la contrainte du temps réel m'était donnée.*

### III.1 - Conception détaillée

*Une fois les données désérialisées, il faut maintenant les stocker dans une base de données. Pour cela il fallait déjà se connecter à la base de données.*

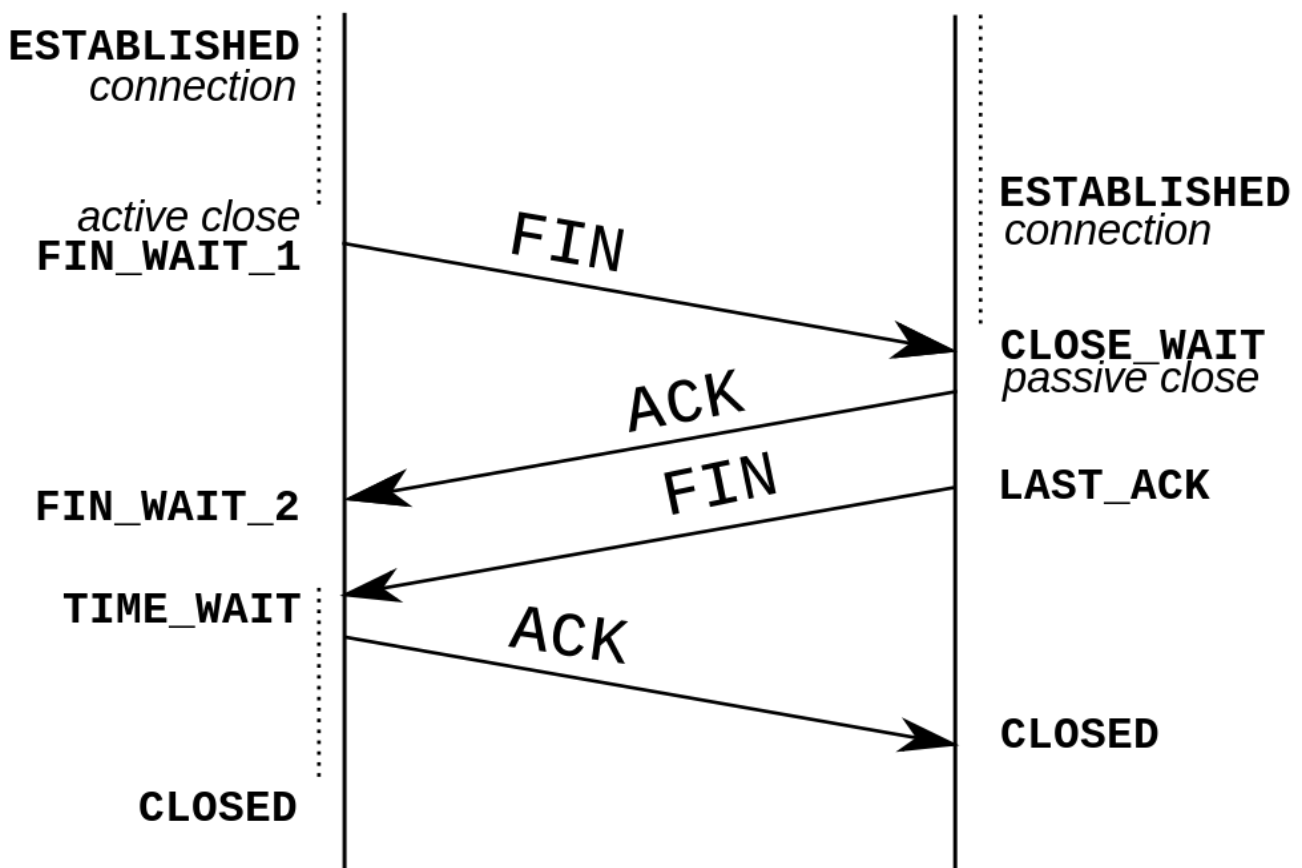


```
16 using namespace std;
17
18 class Communication
19 {
20 private:
21     string trame;
22     string adresseSrv = "192.168.115.101";
23     string port = "3306";
24     string loginBDD = "makair";
25     string MotDePasseBDD = "makair";
26     string requete;
27
28     sql::Driver *driver;
29     sql::Connection *con;
30
31 public:
32     Communication();
33     ~Communication();
34
35     void sendToDb(string Tab[10]);
36
37     void SplitTrame();
38     void SplitTrameTest();
39     void setFrame(string makairTrame);
40
41 };
42
43
```

On peut voir dans le .h de Communication adresse du serveur ou est la base de données, le port utilisé pour la connection, le login et le mot de passe. Tout cela est mis dans le private pour des raisons évidentes de sécurité.

Ensuite une fois connecter à la base de données, il fallait donc envoyer périodiquement les données. Comme dit précédemment j'ai dû faire un choix pour la communication entre les données du MakAir et la base de données. Mon choix était donc le TCP pour des raisons de sécurité dans l'envoi des données avec la phase de connexion du protocole TCP.

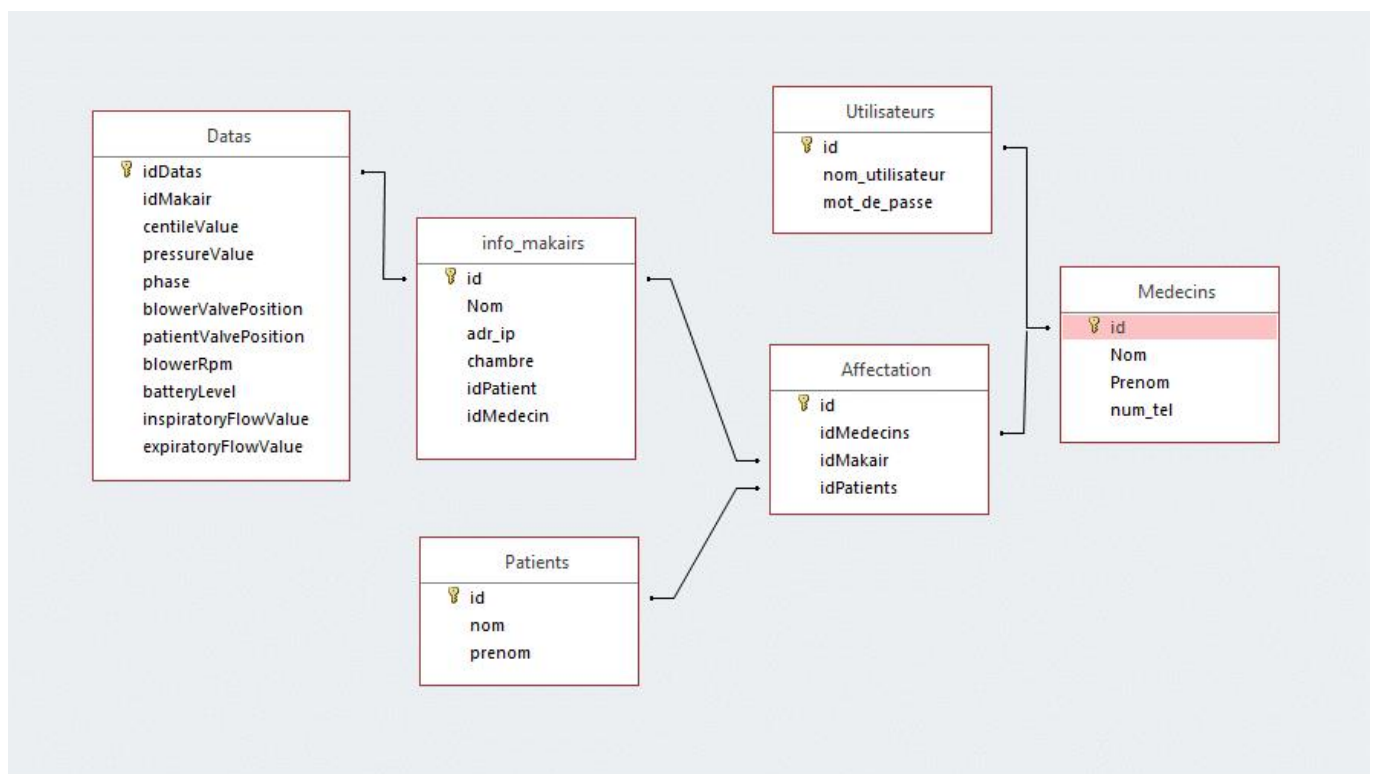
# Initiator                      Receiver














































































Dans mon code on peut le voir dans le Communication.cpp au niveau du constructeur de Communication

```
Communication::Communication() {  
    requete = "tcp://" + adresseSrv + ":" + port;  
  
    /* Create a connection */  
    driver = sql::mysql::get_driver_instance();  
    con = driver->connect(requete, loginBDD, MotDePasseBDD);  
    /* Connect to the MySQL test database */  
    con->setSchema("makair");  
}  
  
Communication::~~Communication() {  
    delete con;  
}
```

Cependant pour envoyer les données à la base de données il faut savoir comment la base de données est faite :

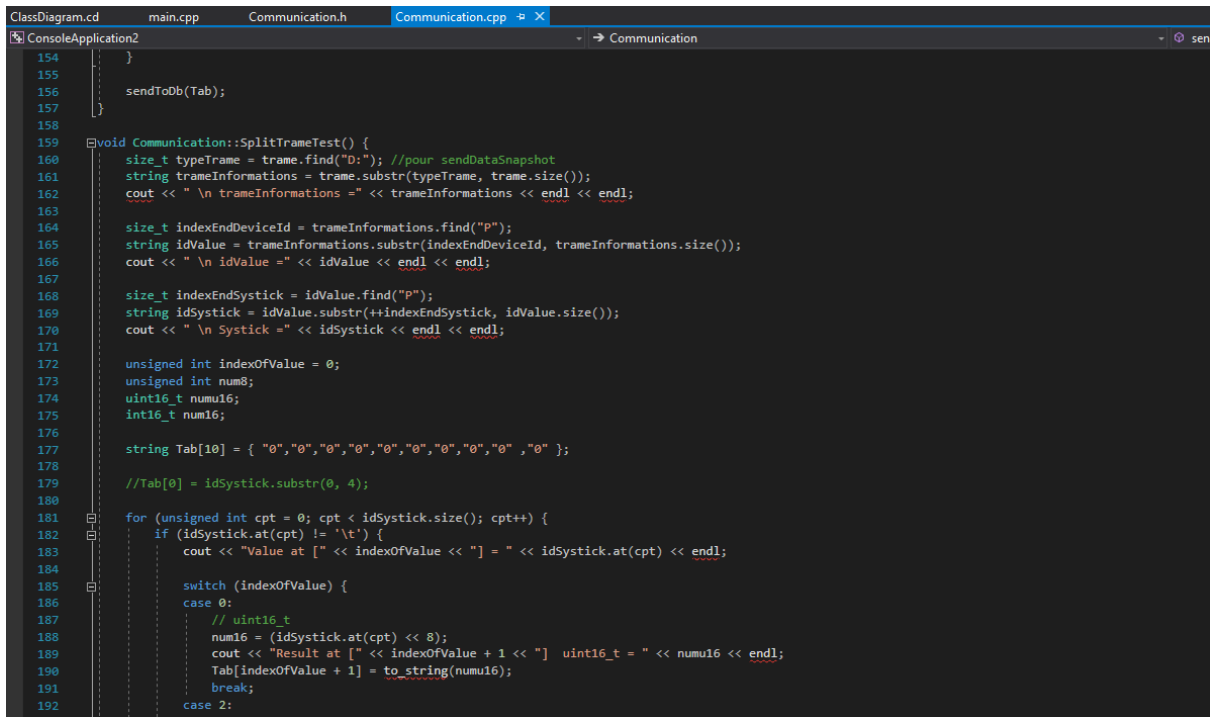


Une fois la communication établie il ne reste plus qu'à savoir si la base de données récupère bien les données.

idDatas 1 idMakair centileValue pressureValue phase blowerValvePosition patientValvePosition blowerRpm batteryLevel inspiratoryFlowValue expiratoryFlowValue														
<input type="checkbox"/>	 Editor	 Copier	 Supprimer	1	0	0	8192	50	52	80	140	255	-4608	8448
<input type="checkbox"/>	 Editor	 Copier	 Supprimer	2	0	0	8192	50	52	80	140	255	-4608	8448
<input type="checkbox"/>	 Editor	 Copier	 Supprimer	3	0	0	8192	50	52	80	140	255	-4608	8448
<input type="checkbox"/>	 Editor	 Copier	 Supprimer	4	0	0	8192	50	52	80	140	255	-4608	8448
<input type="checkbox"/>	 Editor	 Copier	 Supprimer	5	0	0	8192	50	52	80	140	255	-4608	8448
<input type="checkbox"/>	 Editor	 Copier	 Supprimer	6	0	0	8192	50	52	80	140	255	-4608	8448
<input type="checkbox"/>	 Editor	 Copier	 Supprimer	7	0	0	8192	50	52	80	140	255	-4608	8448
<input type="checkbox"/>	 Editor	 Copier	 Supprimer	8	0	0	8192	50	52	80	140	255	-4608	8448
<input type="checkbox"/>	 Editor	 Copier	 Supprimer	9	0	0	8192	50	52	80	140	255	-4608	8448
<input type="checkbox"/>	 Editor	 Copier	 Supprimer	10	0	0	8192	50	52	80	140	255	-4608	8448
<input type="checkbox"/>	 Editor	 Copier	 Supprimer	11	0	0	8192	50	52	80	140	255	-4608	8448
<input type="checkbox"/>	 Editor	 Copier	 Supprimer	12	0	0	8192	50	52	80	140	255	-4608	8448
<input type="checkbox"/>	 Editor	 Copier	 Supprimer	13	0	0	8192	50	52	80	140	255	-4608	8448
<input type="checkbox"/>	 Editor	 Copier	 Supprimer	14	0	0	8192	50	52	80	140	255	-4608	8448
<input type="checkbox"/>	 Editor	 Copier	 Supprimer	15	0	0	8192	50	52	80	140	255	-4608	8448
<input type="checkbox"/>	 Editor	 Copier	 Supprimer	16	0	0	8192	50	52	80	140	255	-4608	8448
<input type="checkbox"/>	 Editor	 Copier	 Supprimer	17	0	0	8192	50	52	80	140	255	-4608	8448
<input type="checkbox"/>	 Editor	 Copier	 Supprimer	18	0	0	8192	50	52	80	140	255	-4608	8448
<input type="checkbox"/>	 Editor	 Copier	 Supprimer	19	0	0	8192	50	52	80	140	255	-4608	8448
<input type="checkbox"/>	 Editor	 Copier	 Supprimer	20	0	0	8192	50	52	80	140	255	-4608	8448
<input type="checkbox"/>	 Editor	 Copier	 Supprimer	21	0	0	8192	50	52	80	140	255	-4608	8448
<input type="checkbox"/>	 Editor	 Copier	 Supprimer	22	0	0	8192	50	52	80	140	255	-4608	8448
<input type="checkbox"/>	 Editor	 Copier	 Supprimer	23	0	0	8192	50	52	80	140	255	-4608	8448
<input type="checkbox"/>	 Editor	 Copier	 Supprimer	24	0	0	8192	50	52	80	140	255	-4608	8448
<input type="checkbox"/>	 Editor	 Copier	 Supprimer	25	0	0	8192	50	52	80	140	255	-4608	8448

On peut donc voir qu'un champ dans la table Data est égale à une valeur transmise par le MakAir.

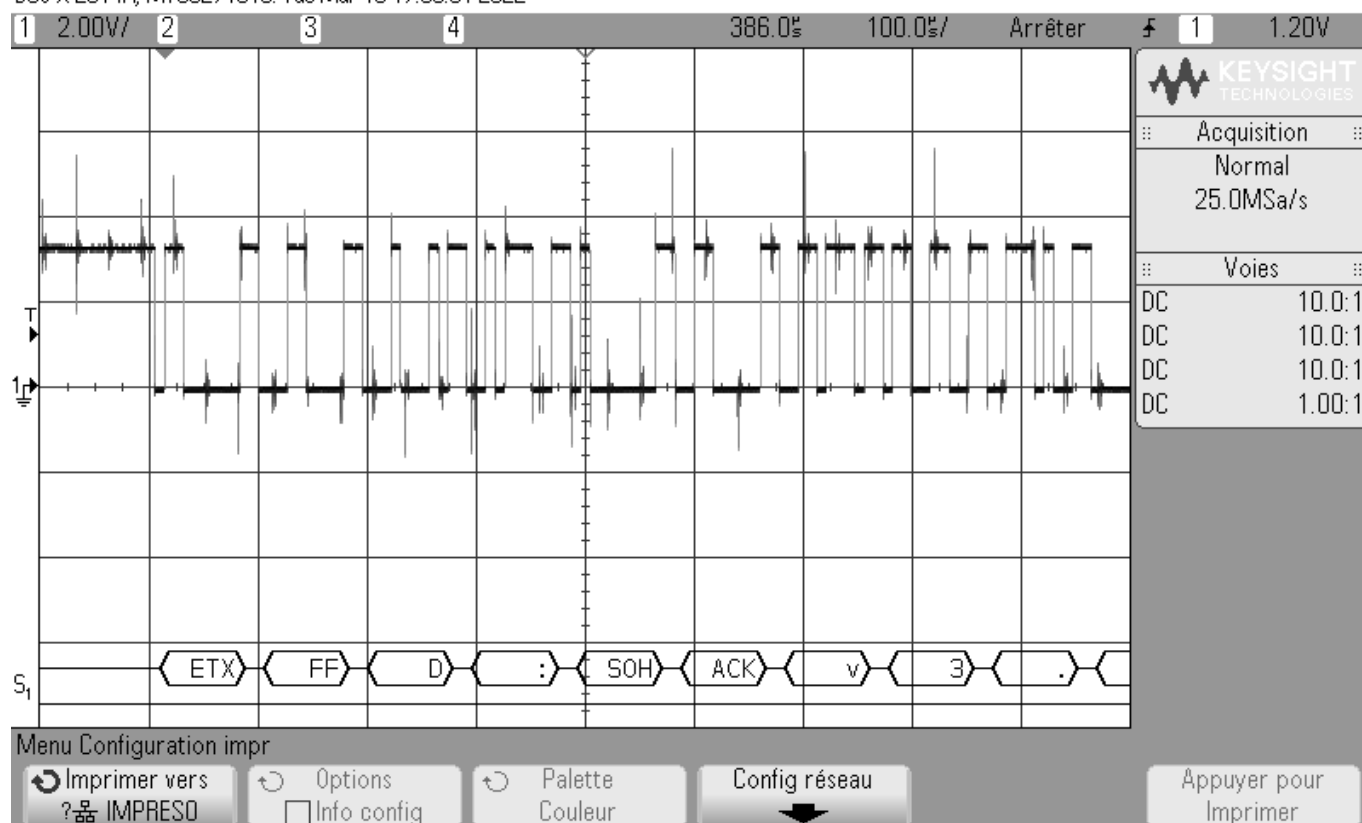
Voici le code qui effectue le travail de séparer les valeurs de la trame :



```
154 }
155
156     sendToDb(Tab);
157 }
158
159 void Communication::SplitFrameTest() {
160     size_t typeFrame = frame.find("D:"); //pour sendDataSnapshot
161     string frameInformations = frame.substr(typeFrame, frame.size());
162     cout << " \n frameInformations =" << frameInformations << endl << endl;
163
164     size_t indexEndDeviceId = frameInformations.find("P");
165     string idValue = frameInformations.substr(indexEndDeviceId, frameInformations.size());
166     cout << " \n idValue =" << idValue << endl << endl;
167
168     size_t indexEndSystick = idValue.find("P");
169     string idSystick = idValue.substr(++indexEndSystick, idValue.size());
170     cout << " \n Systick =" << idSystick << endl << endl;
171
172     unsigned int indexOfValue = 0;
173     unsigned int num8;
174     uint16_t num16;
175     int16_t num16;
176
177     string Tab[10] = { "0","0","0","0","0","0","0","0","0","0" };
178
179     //Tab[0] = idSystick.substr(0, 4);
180
181     for (unsigned int cpt = 0; cpt < idSystick.size(); cpt++) {
182         if (idSystick.at(cpt) != '\t') {
183             cout << "Value at [" << indexOfValue << "] = " << idSystick.at(cpt) << endl;
184
185             switch (indexOfValue) {
186                 case 0:
187                     // uint16_t
188                     num16 = (idSystick.at(cpt) << 8);
189                     cout << "Result at [" << indexOfValue + 1 << "] uint16_t = " << num16 << endl;
190                     Tab[indexOfValue + 1] = to_string(num16);
191                     break;
192                 case 2:
193                     // uint16_t
```

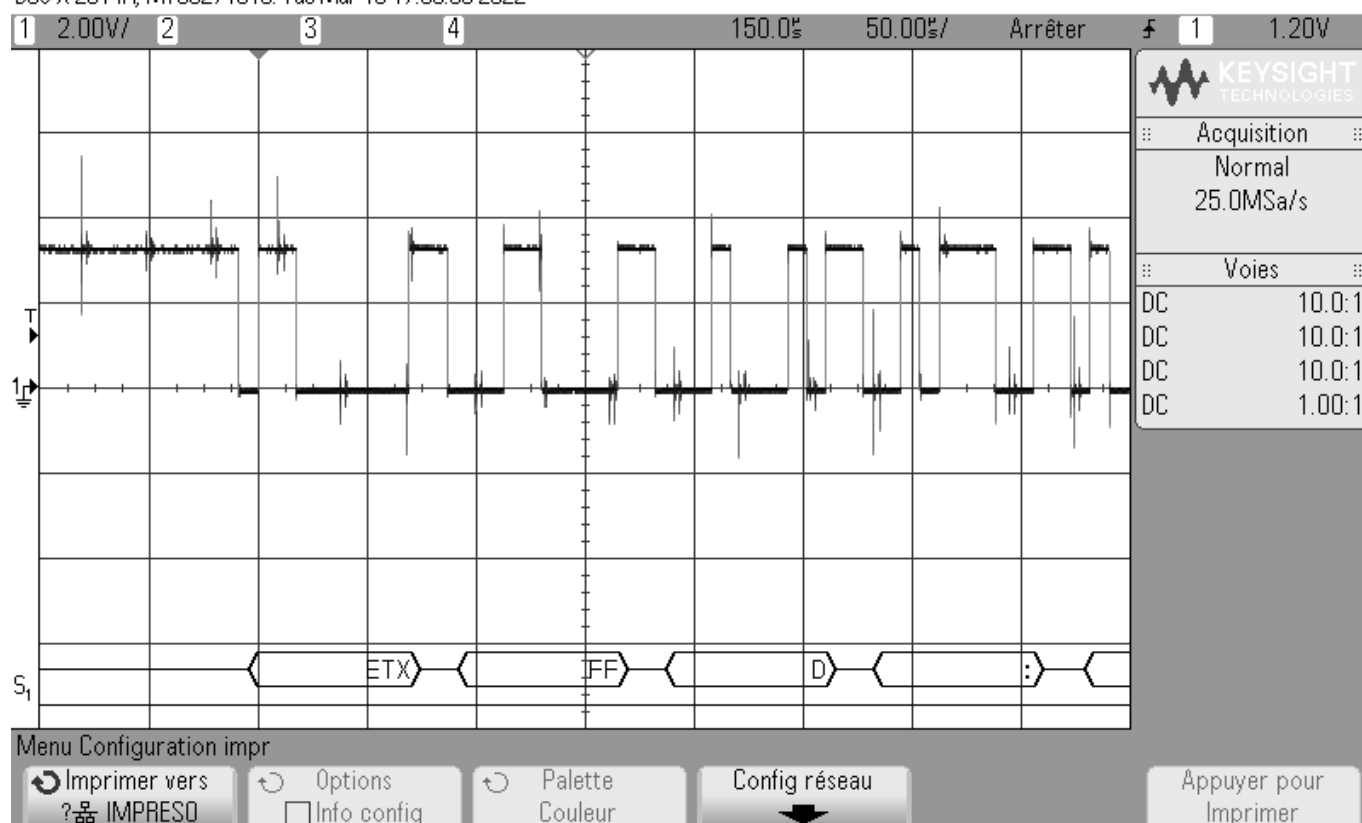
La méthode se situe dans « Communication.cpp » et se nomme « splitFrameTest » car elle sépare les valeurs se trouvant dans la trame de test. Pour pouvoir séparer la trame j'ai dû me renseigner sur cette dernière. Pour cela j'ai observé la trame à l'oscilloscope :

DSO-X 2014A, MY56271913: Tue Mar 15 17:05:31 2022



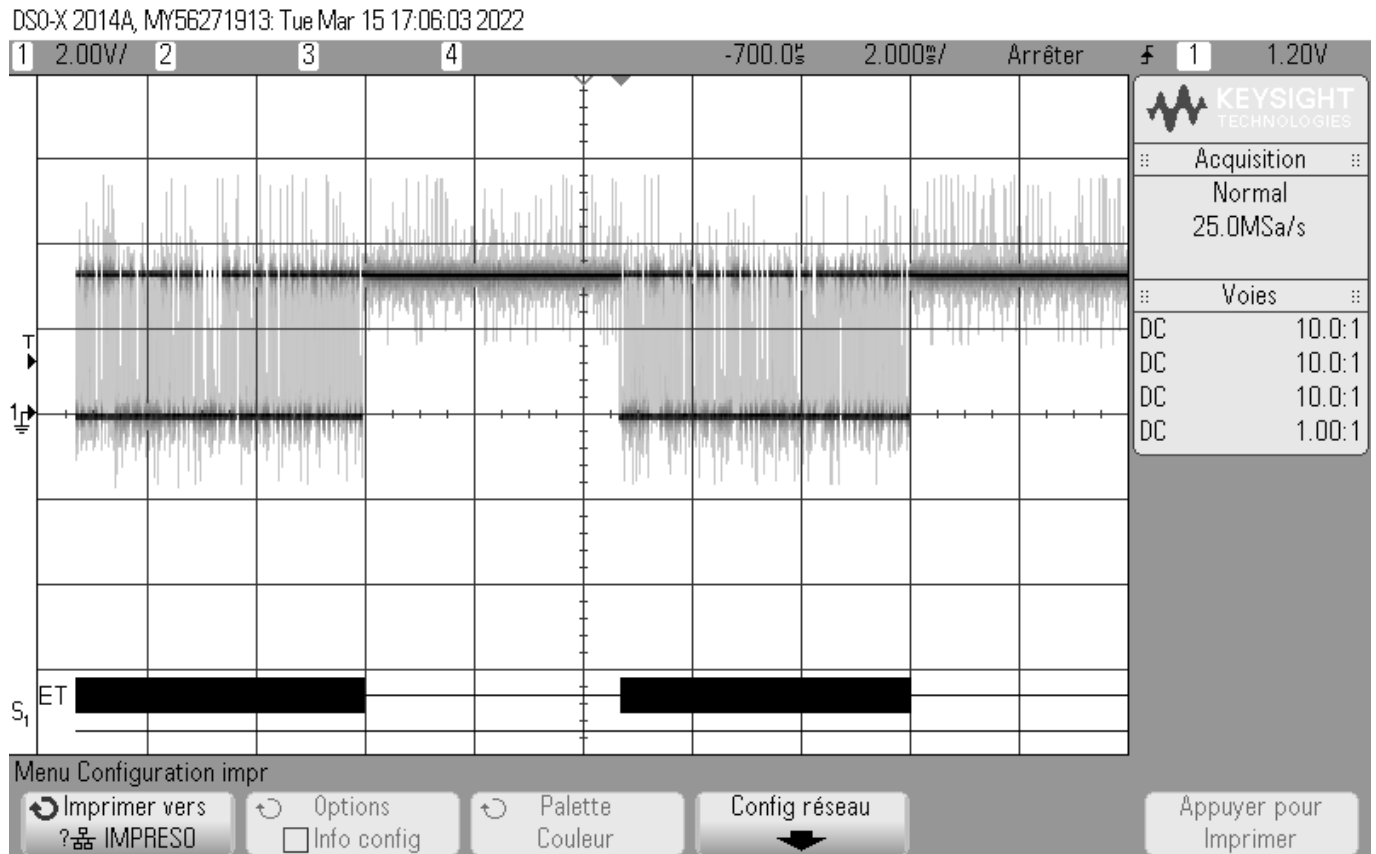
Voici le début de la trame

DSO-X 2014A, MY56271913: Tue Mar 15 17:06:30 2022





Ci-dessus, on peut retrouver les deux bits de stop.



On peut voir ci-dessus la longueur d'une trame.

Ensuite, il a fallu que je me renseigne sur comment était codée chaque valeur et surtout sur combien d'octet, j'ai donc regardé encore une fois le GitHub du MakAir dans le code « sendDataSnapshot » :

```
void sendDataSnapshot(uint16_t centileValue,
                     int16_t pressureValue,
                     CyclePhases phase,
                     uint8_t blowerValvePosition,
                     uint8_t patientValvePosition,
                     uint8_t blowerRpm,
                     uint8_t batteryLevel,
                     int16_t inspiratoryFlowValue,
                     int16_t expiratoryFlowValue) {
```

On peut donc voir que par exemple centileValue est codé sur 16 bits donc 2 octets.

Ensuite je me suis penché sur le début de la trame, on peut donc voir quelque chose de redondant dans les trames :



« D : » : code pour dire que le système n'a pas de problème

« v3.0.1 » : version du MakAir

Ensuite il me reste plus qu'à envoyer les valeurs qui sont situées dans un tableau dans la base de données :

```
void Communication::sendToDb(string Tab[10])
{
    try {
        sql::ResultSet *res;
        sql::Statement *stmt;
        stmt = con->createStatement();

        // COMMANDE SQL
        unsigned int sizeOfTable = 0;

        res = stmt->executeQuery("SELECT COUNT(*) FROM Datas");
        while (res->next()) {
            cout << res->getInt(1) << endl;
            sizeOfTable = res->getInt(1);
        }

        if (sizeOfTable < 150) {
            const string comQuery = "(idMakair, centileValue, pressureValue, phase ,blowerValvePosition ,patientValvePosition, blowerRpm, batteryLevel, inspiratoryFlowValue, expiratoryFlowValue ) ";
            const string values = "VALUES(" + Tab[0] + "," + Tab[1] + "," + Tab[2] + "," + Tab[3] + "," + Tab[4] + "," + Tab[5] + "," + Tab[6] + "," + Tab[7] + "," + Tab[8] + "," + Tab[9] + " )";

            cout << "INSERT INTO Datas " + comQuery + values << endl;

            res = stmt->executeQuery("INSERT INTO Datas " + comQuery + values);
            while (res->next()) {
                cout << "\t... MySQL replies: ";
                cout << res << endl;
                cout << "\t... MySQL says it again: ";
                cout << res->getString(1) << endl;
            }
        }
    }
}
```

La méthode « sendToDb » situé dans Communication.cpp permet de le faire grâce à des commande sql insérer dans le code C++ (INSERT INTO Datas).

## III.2 - Tests unitaires

### III.2.1 - Test unitaire de la méthode « connexionALaBDD »

Élément testé :	Connexion à la Base de Données			
Objectif du test :	Tester la connexion à la base de données			
Nom du testeur :	Johan		Date :06/04/22	
Moyens mis en œuvre :	Logiciel : Visual studio	Matériel : RPI, trame test		Outil de développement : Visual studio
Procédure du test : Vérifier grâce au constructeur de Communication et à la méthode sendToDb				
Id	Description du vecteur de test	Résultat attendu	Résultat obtenu	Validation (O/N)
	Tester la récupération des données dans la base de données avec la trame test	Toutes les valeurs sont mises dans les champs correspondants	Toutes les valeurs sont mises dans les champs correspondants	O
	Tester la récupération des données dans la base de données avec une nouvelle trame test	Les valeurs au sein des champs sont bien changées	Les valeurs au sein des champs sont bien changées	O
Conclusion du test :		L'envoi des données s'effectue bien		

### III.2.2 - Problèmes rencontrés

Le plus gros problème rencontré dans ce cas d'utilisation se trouvait encore une fois dans la mauvaise liaison série.

Un autre problème est parvenu c'est celui de la saturation de la base de données, la base de données devenait très rapidement saturée. La solution était donc de supprimer les plus vieilles valeurs pour laisser place aux dernières valeurs.

## **IV - Bilan de la réalisation personnelle**

- *UCE1 « collecter les données » n'est pas totalement validé car finalement je travaille encore avec une trame test.*
- *Il faut donc travailler avec les trames de la liaison série tout en restant en temps réel.*

### IV.1 - Parties restant à développer

- *Je pourrais encore travailler en cherchant de nouvelles valeurs à transmettre du MackAir vers le centralisateur.*

### IV.2 - Conclusion personnelle

- *Tout d'abord, j'ai trouvé ce projet très intéressant d'un part par son contexte et ensuite par la complexité du travail. J'ai aimé son contexte car en plus d'être d'actualité il est très complet. Le fait de présenter le projet au CHU de Cholet montre que notre travail aboutit à quelque chose de très concret ce qui a mis une pression en plus mais aussi une motivation. J'adore personnellement le travail en groupe, l'entente au sein du groupe était parfaite, chacun était très motivés ce qui a rendu le projet encore plus intéressant. J'ai pu apprendre beaucoup de chose et ainsi découvrir ce que j'aime le moins et ce que j'aimais le plus.*