



Projet snirium

Date : 16/12/2021

1 - Présentation du projet.....	3
2 - Partie développement :	4
2.1– Création d’un serveur TCP	4
2.2 – Déplacement du robot	5
2.3 – Connexion au robot en SSH.....	5
2.4 – Lancement de Docker	6
2.5 – Compilation du serveur TCP.....	6
2.6 – Exécution du serveur TCP sur le robot.....	7
2.7 – Ajout d’autres fonctionnalités	7
2.8 – Création de la trame	8
2.9 – Chiffrement de la trame avec XOR.....	8
3 – Présentation de l’IHM	8
3.1 – Création du serveur TCP.....	8
3.2 – Affichage d’erreur	9
3.3 – Déplacement du robot	9
3.4 – Calcul taux SNIRIUM.....	10
3.5 – Calcul d’angle	10
3.6 – Calcul de distance.....	10
3.7 – Création d’un Timer	11
3.7 – Déchiffrage de la Trame.....	11

1 – Présentation du projet

Nous avons travaillé sur un robot lego programmable de ce type pour nous permettre de réaliser les tâches demandées.



Nous nous sommes réparti les tâches en deux parties.

- Une partie développement
- Une partie IHM

2 - Partie développement :

2.1– Création d'un serveur TCP

J'ai tout d'abord créé un serveur TCP pour faciliter ma communication avec le robot.

```
1 // Librairies nécessaires
2 #include <iostream>
3 #include <unistd.h>
4 #include <cerrno>
5 #include <sys/un.h>
6 #include <arpa/inet.h>
7 // Espace de nom utilisé
8 using namespace std;
9 // Point d'entrée du programme
10 int main()
11 {
12     // Création de la socket serveur
13     int sd_serveur = socket(AF_INET, SOCK_STREAM, 0);
14     // Configuration de la socket, notamment le port d'écoute
15     struct sockaddr_in cfg_serveur;
16     cfg_serveur.sin_family = AF_INET;
17     cfg_serveur.sin_addr.s_addr = htonl(INADDR_ANY);
18     cfg_serveur.sin_port = htons(1664);
19     // Attachement de la socket au port défini
20     bind(sd_serveur, (struct sockaddr *)&cfg_serveur, sizeof(cfg_serveur));
21     // Création une file d'attente de connexion
22     listen(sd_serveur, 5);
23     while(1)
24     {
25         // Dès qu'un nouveau client se connecte à notre serveur,
26         // une nouvelle socket est créée pour gérer le client
27         int sd_client = accept(sd_serveur, NULL, NULL);
28         // Réception de la requête du client
29         char buffer[1024];
30         memset(buffer, 0x00, 1024);
31         int nbOctets = recv(sd_client, buffer, sizeof(buffer), 0);
32         string reponse(buffer);
33         cout << reponse << endl;
34         // Envoi de la réponse au client
35         string requete = "Hello world!";
36         send(sd_client, requete.c_str(), requete.size(), 0);
37         // Fermeture de la socket client
38         close(sd_client);
39     }
40     // Fermeture de la socket serveur
41     close(sd_serveur);
42     // Fin du programme
43     return 0;
44 }
```

2.2 – Déplacement du robot

1. Après avoir créé le serveur TCP j'ai fait en sorte qu'il permette au robot de se déplacer.

```
//configuration de la touche pour faire avancer le robot
if (reponse[0] == 'Z' )
{
    monRobot.changerPuissanceMoteurs(0, 0, 0);
    monRobot.changerPuissanceMoteurs(100, 0, 100);
    monRobot.parler("advance", true);
    send(sd_client, requete.c_str(), requete.size(), 0);
}

//configuration de la touche pour faire reculer le robot
if (reponse[0] == 'S' )
{
    monRobot.changerPuissanceMoteurs(0, 0, 0);
    monRobot.changerPuissanceMoteurs(-100, 0, -100);
    monRobot.parler("move back", true);
    send(sd_client, requete.c_str(), requete.size(), 0);
}

//configuration de la touche pour faire tourner à droite le robot
if (reponse[0] == 'Q' )
{
    monRobot.changerPuissanceMoteurs(0, 0, 0);
    monRobot.changerPuissanceMoteurs(-100, 0, 100);
    monRobot.parler("left", true);
    send(sd_client, requete.c_str(), requete.size(), 0);
}

//configuration de la touche pour faire tourner à droite le robot
if (reponse[0] == 'D' )
{
    monRobot.changerPuissanceMoteurs(0, 0, 0);
    monRobot.changerPuissanceMoteurs(100, 0, -100);
    monRobot.parler("right", true);
    send(sd_client, requete.c_str(), requete.size(), 0);
}
```

2.3 – Connexion au robot en SSH

Pour tester chaque nouvelle fonction rajouter au robot j'utilisais le terminal.

Pour me connecter au robot j'utilisais la commande ssh et j'entrais son nom et son adresse IP.

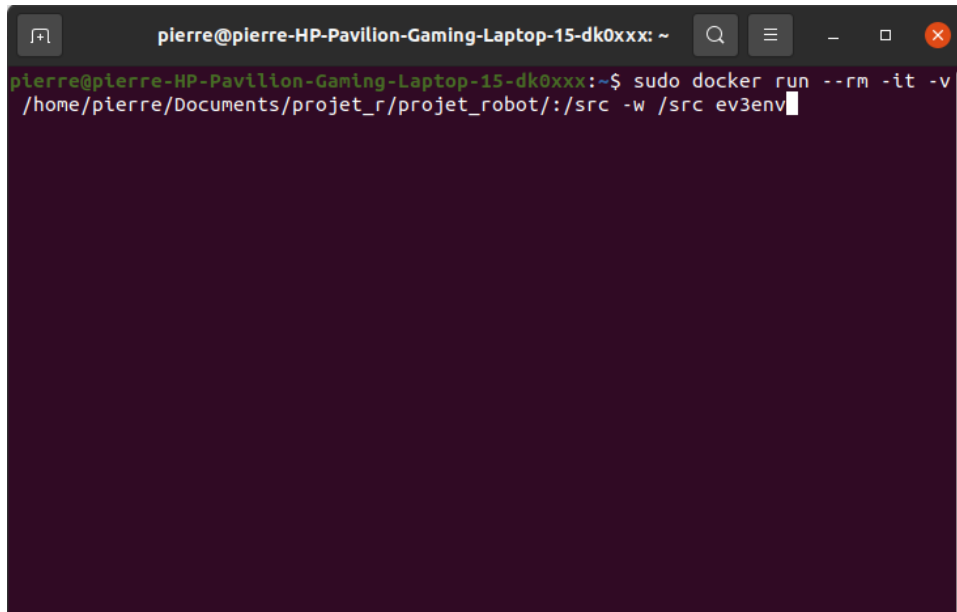
Ainsi que le mot de passe "maker".



PIERRE ALLAIRE
MATHEO GONCALVES

2.4 – Lancement de Docker

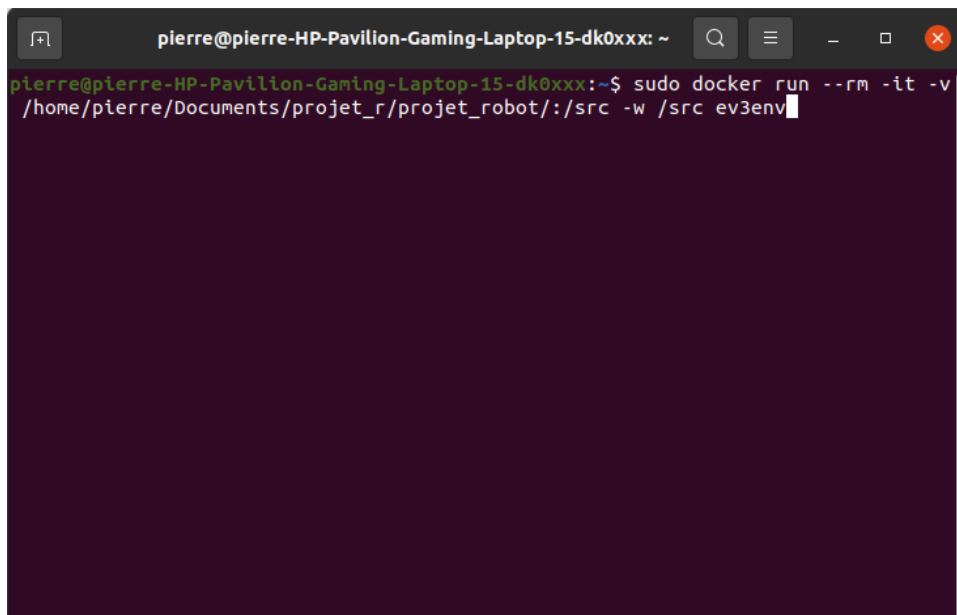
Une fois connecté au robot je lançais Docker avec une autre fenêtre de terminal.

A terminal window titled 'pierre@pierre-HP-Pavilion-Gaming-Laptop-15-dk0xxx: ~' with search, menu, and window control icons. The command 'sudo docker run --rm -it -v /home/pierre/Documents/projet_r/projet_robot/:/src -w /src ev3env' is entered and the cursor is at the end of the line.

```
pierre@pierre-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~$ sudo docker run --rm -it -v /home/pierre/Documents/projet_r/projet_robot/:/src -w /src ev3env
```

2.5 – Compilation du serveur TCP

Une fois lancé je compilais avec la commande make all.

A terminal window titled 'pierre@pierre-HP-Pavilion-Gaming-Laptop-15-dk0xxx: ~' with search, menu, and window control icons. The command 'sudo docker run --rm -it -v /home/pierre/Documents/projet_r/projet_robot/:/src -w /src ev3env' is entered and the cursor is at the end of the line.

```
pierre@pierre-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~$ sudo docker run --rm -it -v /home/pierre/Documents/projet_r/projet_robot/:/src -w /src ev3env
```

Parfois je devais utiliser la commande scp au début de l'utilisation du robot.



```
compiler@54da9ce51460: /src
pierre@pierre-HP-Pavilion-Gaming-Laptop-15-dk0xxx:~$ sudo docker run --rm -it -v
/home/pierre/Documents/projet_r/projet_robot/:/src -w /src ev3env
[sudo] Mot de passe de pierre :
compiler@54da9ce51460:/src$ scp serveur_TCP robot@192.168.1.171:/home/robot/snr
2/
```

2.6 – Exécution du serveur TCP sur le robot

Une fois la compilation effectuée je n'avais plus qu'à exécuter mon serveur sur le robot avec la commande `./` suivit du nom de mon serveur donc pour moi `./serveur_TCP`.

2.7 – Ajout d'autres fonctionnalités

J'ai par la suite ajouté d'autres fonctionnalités comme faire bouger les fourches du robot ou l'arrêter.

```
//configuration de la touche pour faire monter les fourches du robot
if (reponse[0] == 'A' )
{
    monRobot.changerPuissanceMoteurs(0, 0, 0);
    monRobot.changerPuissanceMoteurs([0, 50, 0]);
    monRobot.parler("get up", true);
    send(sd_client, requete.c_str(), requete.size(), 0);
}

//configuration de la touche pour descendre les fourches du robot
if (reponse[0] == 'E' )
{
    monRobot.changerPuissanceMoteurs(0, 0, 0);
    monRobot.changerPuissanceMoteurs(0, -50, 0);
    monRobot.parler("get off", true);
    send(sd_client, requete.c_str(), requete.size(), 0);
}

//configuration de la touche permettant de stopper le robot
if (reponse[0] == 'T' )
{
    monRobot.changerPuissanceMoteurs(0, 0, 0);
    monRobot.parler("stop", true);
    send(sd_client, requete.c_str(), requete.size(), 0);
}
```

2.8 – Création de la trame

J’ai ensuite créé une trame contenant les méthodes à retournés par le robot.

```
//configuration de la touche permettant de retranscrire la trame du robot
if (reponse[0] == 'Y' )
{
    // Construction de la chaîne de caractères en mémoire
    ostringstream preparation;

    //Retranscription de la trame
    preparation << monRobot.recupererGyroscopeAngle() << ";" << monRobot.recupererDistance() << ";" << monRobot.recupererLumiereReflechie() << ";" << monRobot.recupererPositionDuMoteur(Robot:

    // Transformation de la chaîne préparée en string
    string chaine = preparation.str();
```

2.9 – Chiffrement de la trame avec XOR

Après avoir réussi à afficher les données sur l’ihm j’ai chiffré la trame avec le protocole de chiffrement XOR.

```
//configuration de la touche permettant de retranscrire la trame du robot
if (reponse[0] == 'Y' )
{
    // Construction de la chaîne de caractères en mémoire
    ostringstream preparation;

    //Retranscription de la trame
    preparation << monRobot.recupererGyroscopeAngle() << ";" << monRobot.recupererDistance() << ";" << monRobot.recupererLumiereReflechie() << ";" << monRobot.recupererPositionDuMoteur(Robot:

    //Création variables
    string enc;
    char key[3] = {4,8,3};

    // Transformation de la chaîne préparée en string
    string chaine = preparation.str();

    //boucle permettant de chiffrer la trame
    for (int i=0; i < chaine.size() ; i++)
    {
        enc+= chaine[i] ^ (int(key[i%3]) + i)% 20;
    }

    cout << enc << endl;
    send(sd_client, enc.c_str(), enc.size(), 0);
```

3 – Présentation de l’IHM

3.1 – Création du serveur TCP

```
// Instanciation de la socket
tcpSocket = new QTcpSocket(this);

// Attachement d'un slot qui sera appelé à chaque fois que des données arrivent
connect(tcpSocket, SIGNAL(readyRead()), this, SLOT(gerer_donnees()));

// Idem pour les erreurs
connect(tcpSocket, SIGNAL(error(QAbstractSocket::SocketError)), this, SLOT(afficher_erreur(QAbstractSocket::SocketError)));
```


3.2 – Affichage d'erreur

Cela permet de voir si l'envoi de la trame a bien effectué

```
174
175 void MainWindow::afficher_erreur(QAbstractSocket::SocketError socketError)
176 {
177     switch (socketError)
178     {
179         case QAbstractSocket::RemoteHostClosedError:
180             break;
181         case QAbstractSocket::HostNotFoundError:
182             QMessageBox::information(this, tr("Client TCP"),
183                                     tr("Hôte introuvable"));
184             break;
185         case QAbstractSocket::ConnectionRefusedError:
186             QMessageBox::information(this, tr("Client TCP"),
187                                     tr("Connexion refusée"));
188             break;
189         default:
190             QMessageBox::information(this, tr("Client TCP"),
191                                     tr("Erreur : %1.")
192                                     .arg(tcpSocket->errorString()));
193     }
194 }
```

3.3 – Déplacement du robot

Pour déplacer le robot nous avons utilisé les touches du clavier Z pour avancer, S pour reculer, Q pour tourner à gauche, D pour tourner à droite, A pour monter le bras, E pour descendre le bras.

```
132
133 void MainWindow::on_Av_clicked()
134 {
135     tcpSocket->write("Z");
136 }
137
138
139
140 void MainWindow::on_Ar_clicked()
141 {
142     tcpSocket->write("S");
143 }
144
145
146 void MainWindow::on_Dr_clicked()
147 {
148     tcpSocket->write("D");
149 }
150
151
152 void MainWindow::on_Ga_clicked()
153 {
154     tcpSocket->write("Q");
155 }
156
157
158 void MainWindow::on_St_clicked()
159 {
160     tcpSocket->write("T");
161 }
162
163
164 void MainWindow::on_MonterBras_clicked()
165 {
166     tcpSocket->write("A");
167 }
168
169
170 void MainWindow::on_BaisserBras_clicked()
171 {
172     tcpSocket->write("E");
173 }
174
```

3.4 – Calcul taux SNIRIUM

Cela permet de calculer le taux de SNIRIUM

```
97 void MainWindow::gerer_donnees()
98 {
99     //Récupération des données
100     QByteArray reponse = tcpSocket->readAll();
101     QString enc = dechiffrer_trame(reponse);
102
103     //Découpage de la trame
104     QStringList trameDecoupee = enc.split(";");
105     qDebug() << reponse << "\n";
106     qDebug() << enc << "\n";
107
108     if (trameDecoupee.size() >= 4)
109     {
110         //taux SNIRIUM
111         unsigned short tauxSNIRIUM = trameDecoupee[2].toShort();
112         ui->RepTauxSnirium->setText(QString("%1").arg(tauxSNIRIUM));
113     }
```

3.5 – Calcul d'angle

Cela permet de calculer l'angle par rapport aux roues

```
113
114     //Angle
115     short angle = trameDecoupee[0].toShort();
116     if (angle > 360 || angle < -360)
117     {
118         angle = angle % 360;
119     }
120     ui->RepAngle->setText(QString("%1").arg(angle));
121 }
```

3.6 – Calcul de distance

Cela permet de calculer la distance entre le robot et un obstacle.

```
121
122     //Distance
123     float distance = trameDecoupee[1].toFloat();
124     ui->RepPosition->setText(QString("%1").arg(distance));
125     if (distance <= 10)
126     {
127         tcpSocket->write("Y");
128     }
```

3.7 – Création d'un Timer

On a créé un timer pour pouvoir récupérer la trame toute les 2 secondes.

```
//création d'un timer pour récupérer la trame
pTimer = new QTimer;
connect(pTimer, SIGNAL(timeout()), this, SLOT(demander_Trame()));
```

3.7 – Déchiffrage de la Trame

```
80
81 ▼ QString MainWindow::dechiffrer_trame(QString enc)
82 {
83     QString decoder_trame;
84     int key[3] = {4,8,3};
85
86 ▼     for (int i = 0; i < enc.size() ; i++)
87     {
88         QChar trame( enc[i].toLatin1() ^ (key[i%3]+i) %20);
89         decoder_trame += trame;
90     }
91
92     return decoder_trame;
93 }
94
```