**Comment ça marche ?**

L’IHM WEB se connecte en WEB Socket au programme C++. Ensuite, le code C++ se connecte en TCP à la Moxa. Et enfin, la Moxa se connecte en série à la caméra.

Voici un schéma représentatif :

IHM Web socket C++ (PC) TCP Moxa Série Caméra

**C’est quoi le WEB Socket ?**

Le WebSocket est un protocole de communication bidirectionnel, permettant une connexion continue entre un navigateur web et un serveur. Contrairement au protocole HTTP classique qui fonctionne de manière unidirectionnelle (demande/réponse), WebSocket permet une communication en temps réel et bidirectionnelle. Cela facilite les échanges de données instantanées entre le client et le serveur, ce qui est particulièrement utile pour les applications web nécessitant des mises à jour en temps réel, comme les applications de chat, de jeux en ligne, de flux d'informations en direct, etc.

**Comment fonctionne le WEB Socket ?**

Le protocole WebSocket , décrit dans la spécification RFC 6455 fournit un moyen d'échanger des données entre le navigateur et le serveur via une connexion persistante. Les données peuvent être transmises dans les deux sens sous forme de “paquets”, sans interrompre la connexion et de requêtes HTTP supplémentaires.

**Comment un code C++ se connecte en TCP à Moxa en série ?**

La connexion TCP en C++ à une Moxa se fait généralement en utilisant des sockets. Les sockets sont une API de programmation réseau standard qui permet la communication entre des applications sur des machines différentes. Voici un exemple de code C++ utilisant les sockets pour se connecter en TCP à un périphérique Moxa :

#include <iostream>

#include <WinSock2.h>

#pragma comment(lib, "ws2\_32.lib")

int main() {

// Initialise Winsock

WSADATA wsaData;

if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {

std::cerr << "Erreur lors de l'initialisation de Winsock" << std::endl;

return -1;

}

// Adresse IP et numéro de port du périphérique Moxa

const char\* moxaIP = "192.168.1.1"; // Remplacez par l'adresse IP réelle de votre Moxa

const int moxaPort = 4001; // Remplacez par le port réel de votre Moxa

// Création du socket

SOCKET clientSocket = socket(AF\_INET, SOCK\_STREAM, 0);

if (clientSocket == INVALID\_SOCKET) {

std::cerr << "Erreur lors de la création du socket" << std::endl;

WSACleanup();

return -1;

}

// Configuration de l'adresse du serveur Moxa

sockaddr\_in serverAddress;

serverAddress.sin\_family = AF\_INET;

serverAddress.sin\_port = htons(moxaPort);

if (inet\_pton(AF\_INET, moxaIP, &serverAddress.sin\_addr) <= 0) {

std::cerr << "Adresse IP invalide" << std::endl;

closesocket(clientSocket);

WSACleanup();

return -1;

}

// Connexion au serveur Moxa

if (connect(clientSocket, (struct sockaddr\*)&serverAddress, sizeof(serverAddress)) == SOCKET\_ERROR) {

std::cerr << "Erreur lors de la connexion au serveur Moxa" << std::endl;

closesocket(clientSocket);

WSACleanup();

return -1;

}

// La connexion a réussi, vous pouvez maintenant envoyer et recevoir des données

// Fermeture du socket

closesocket(clientSocket);

WSACleanup();

return 0;

}

**Comment Moxa se connecte à une caméra branchée en série ?**

#include <iostream>

#include <cstring>

#include <unistd.h>

#include <fcntl.h>

#include <termios.h>

int main() {

// Port série auquel la caméra est connectée

const char\* serialPort = "/dev/ttyS0"; // Remplacez par le port série réel de votre Moxa

// Ouvrir le port série

int serial\_fd = open(serialPort, O\_RDWR | O\_NOCTTY | O\_NDELAY);

if (serial\_fd == -1) {

std::cerr << "Erreur lors de l'ouverture du port série" << std::endl;

return -1;

}

// Configuration des paramètres du port série

struct termios serialConfig;

tcgetattr(serial\_fd, &serialConfig);

serialConfig.c\_cflag = B9600 | CS8 | CLOCAL | CREAD;

serialConfig.c\_iflag = IGNPAR;

serialConfig.c\_oflag = 0;

serialConfig.c\_lflag = 0;

tcflush(serial\_fd, TCIFLUSH);

tcsetattr(serial\_fd, TCSANOW, &serialConfig);

// Exemple de commande pour déplacer la caméra vers la droite

const char\* moveRightCommand = "RIGHT\n";

write(serial\_fd, moveRightCommand, strlen(moveRightCommand));

// Attendez un certain temps pour permettre le mouvement (ajustez si nécessaire)

usleep(500000); // 500 ms

// Autres commandes pour le mouvement de la caméra, l'alimentation, etc.

// Fermer le port série

close(serial\_fd);

return 0;

}