```cpp
#include "robot.h"

Robot::Robot() : flux(&service)
{
  connect(&service, SIGNAL(readyRead()), this, SLOT(reponse()));
  connect(&service, SIGNAL(connected()), this, SLOT(connexion()));
  connect(&service, SIGNAL(disconnected()), this, SLOT(deconnexion()));
  connect(&service, SIGNAL(error(QAbstractSocket::SocketError)), this, SLOT(erreur()));
}

QString Robot::hexa2(int decimal)
{
  QString chaine;
  chaine.setNum(decimal, 16);
  if (chaine.length()<2) chaine = QString("0%1").arg(chaine);
  return chaine;
}

QString Robot::hexa4(int decimal)
{
  QString chaine;
  chaine.setNum(decimal, 16);
  while (chaine.length()<4) chaine = QString("0%1").arg(chaine);
  return QString("%1%2
%3%4").arg(chaine[2]).arg(chaine[3]).arg(chaine[0]).arg(chaine[1]);
}

void Robot::playTone(int frequence, int duree)
{
  // 00 Sans retour
  // 03 PLAYTONE
  // %1 Fréquene en Hertz
  // %2 Durée en millisecondes
  flux << QString("00 03 %1 %2").arg(hexa4(frequence)).arg(hexa4(duree)) << endl;
}

void Robot::demarrerMoteur(Moteur moteur, int vitesse)
{
  int calcul = (256+vitesse) % 256;
  // 00 Sans retour
  // 04 SETOUTPUTSTATE
  // %1 Port (0-2)
  // %2 Puissance (-100 à 100)
  // 05 (4+1) Activation du moteur en mode régulation
  // 01 Régulation de vitesse
  // 00 Ratio (-100 à 100)
  // 20 Démarrage du moteur à sa pleine vitesse sans rampe d'accélération
  // 00 00 00 00 Limite vitesse
  flux << QString("00 04 %1 %2 05 01 00 20 00 00 00
00").arg(hexa2(moteur)).arg(hexa2(calcul)) << endl;
}

void Robot::arreterMoteur(Moteur moteur)
{
  // 00 Sans retour
  // 04 SETOUTPUTSTATE
  // %1 Port (0-2)
  // 00 Tout à zéro pour arrêter le moteur sélectionné
    flux << QString("00 04 %1 00 00 00 00 00 00 00 00 00").arg(hexa2(moteur)) << endl;
}

void Robot::initialisationCapteurs()
{
  // 00 Sans retour
  // 05 SETINPUTMODE
  // 02 Entrée 3 (Fin de course)
  // 0B Type de capteur (01 --> SWITCHED)
  // 20 Mode de capture (20 --> BOOLEANMODE)
  flux << QString("00 05 02 01 20") << endl;
```

```
  // 00 Sans retour
  // 05 SETINPUTMODE
  // 02 Entrée 1 (ULTRASON)
  // 01 Type de capteur (0B --> Utilisation du bus I2C)
  // 20 Mode de capture (00 --> RAWMODE) prise en compte des données brutes
  flux << QString("00 05 00 0B 00") << endl;
  // 00 Sans retour
  // 0F LSWRITE
  // 00 Entrée 1 (ULTRASON)
  // 03 Longueur de la commande
  // 00 Longueur de la réponse
  // 02 41 02 Fixe le capteur ULTRASON en mode de lecture en continu
  flux << QString("00 0F 00 03 00 02 41 02") << endl;
}

bool Robot::finDeCourse()
{
  reception = false;
  // 10 Taille du retour en Hexadécimal (16 octets)
  // 07 GETINPUTVALUES
  // 02 Entrée 3 (Fin de course)
  flux << QString("10 07 02") << endl;
  while (!reception) msleep(30);
  QString B12 = resultat.split(" ").at(12);
  return B12.toInt() == 1;
}

int Robot::distance()
{
  reception = false;
  // 00 Sans retour
  // 0F LSWRITE
  // 00 Entrée 1 (ULTRASON)
  // 02 Longueur de la commande
  // 08 Longueur de la réponse
  // 02 42 Spécifie l'adresse des variables du capteur ULTRASON où se situe la mesure
  flux << QString("00 0F 00 02 08 02 42") << endl;
  while (!reception) msleep(20);
  reception = false;
  // 14 Lecture de 20 octets
  // 10 LSREAD
  // 00 Entrée 1 (ULTRASON)
  flux << QString("14 10 00") << endl;
  while (!reception) msleep(20);
  QString B4 = resultat.split(" ").at(4);
  return B4.toInt();
}

void Robot::seDeconnecter()
{
  flux << "fin" << endl;
  service.close();
}

void Robot::reponse()
{
  resultat = service.readLine();
  reception = true;
}

void Robot::seConnecter(QString adresse)
{
  service.connectToHost(adresse, 5588);
}
```