WEEZ-U WELDING

YOUR WELDING TOOLS

Package documentaire pour Nimbl’Bot

21 mars 2025

101 rue de Coulmiers | 44 000 | NANTES

Rappel de l’architecture générale

Architecture générale pour Nimbl’Bot

Poste à souder

PC Nimbl-Bot

Interface relais

Manette

Modbus

PC Weez-U

Ethernet

Caméra USB

WEEZ-U WELDING

YOUR WELDING TOOLS

Composants hardware pour Nimbl'Bot

Manette

WeezTouch

SUBD 15

HDMI

0 / 24 V

Signal de soudage vers le générateur

Carte d'interface

GPIO

COM

24 V

PC industriel

AAEON Boxer 6639

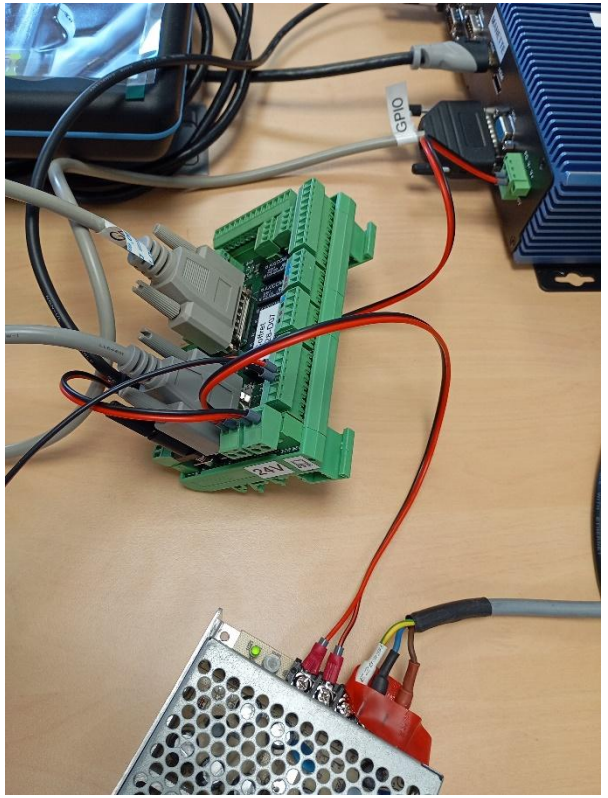
Ethernet robot

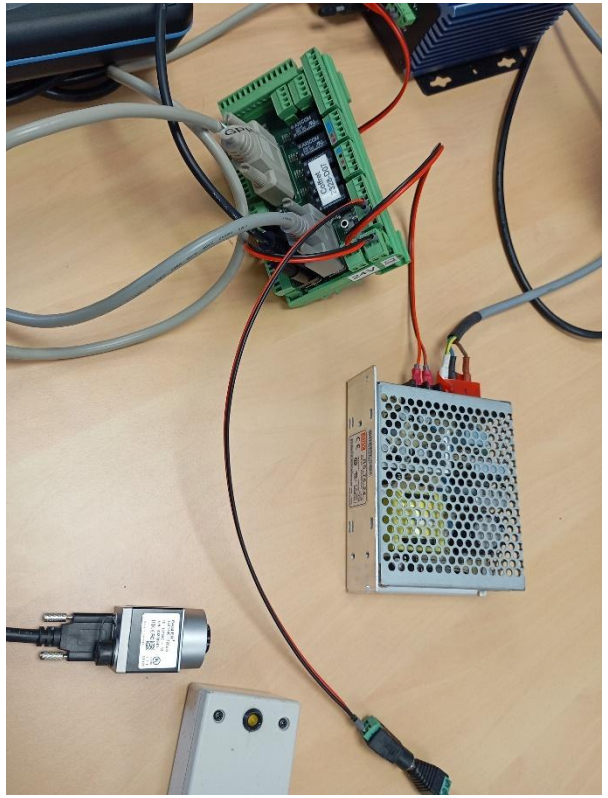
USB

Caméra

WEEZ-U WELDING
YOUR WELDING TOOLS

Composants hardware pour Nimbl’Bot





WEEZ-U WELDING
YOUR WELDING TOOLS

Carte coffret

Schéma de la carte coffret

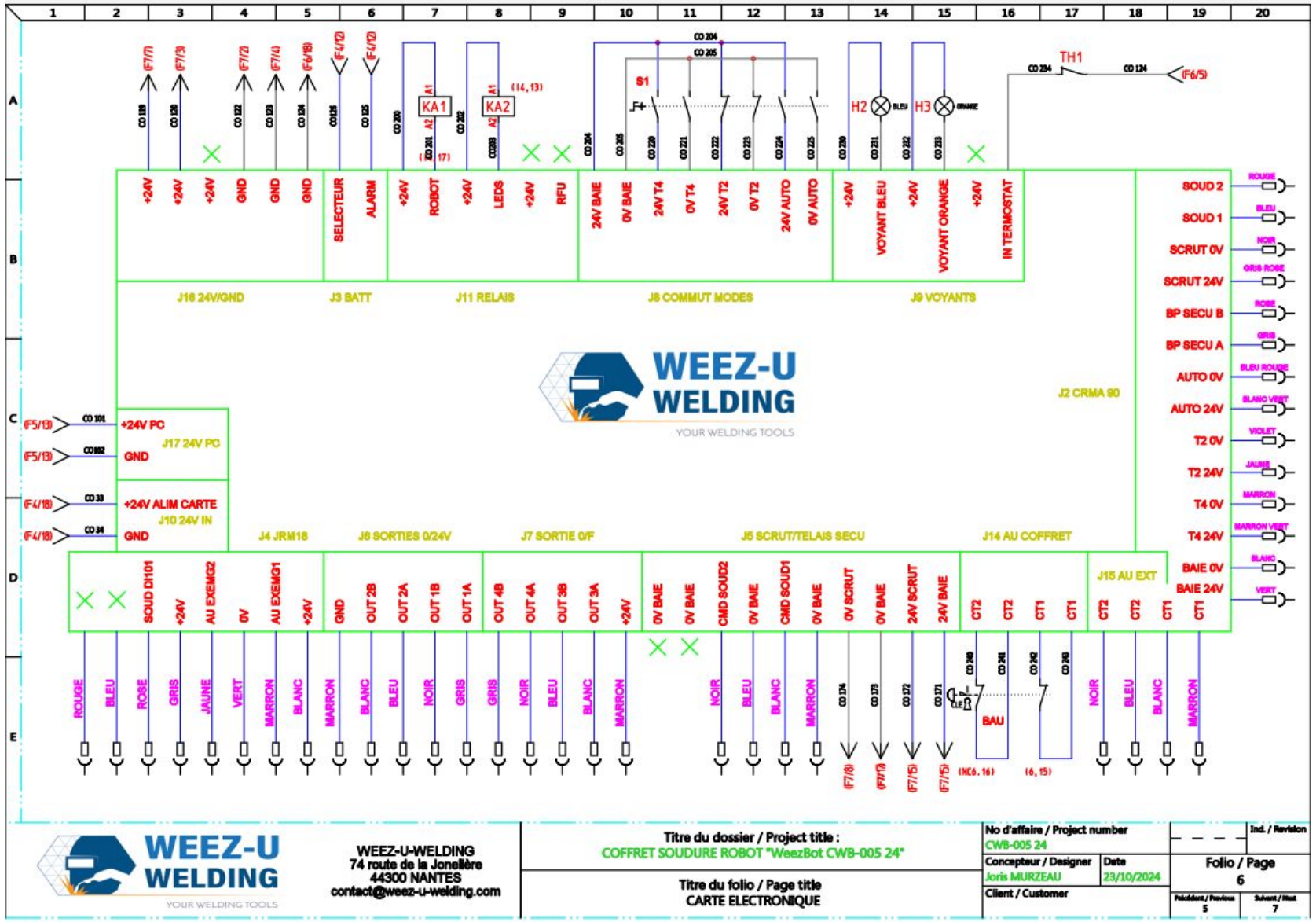
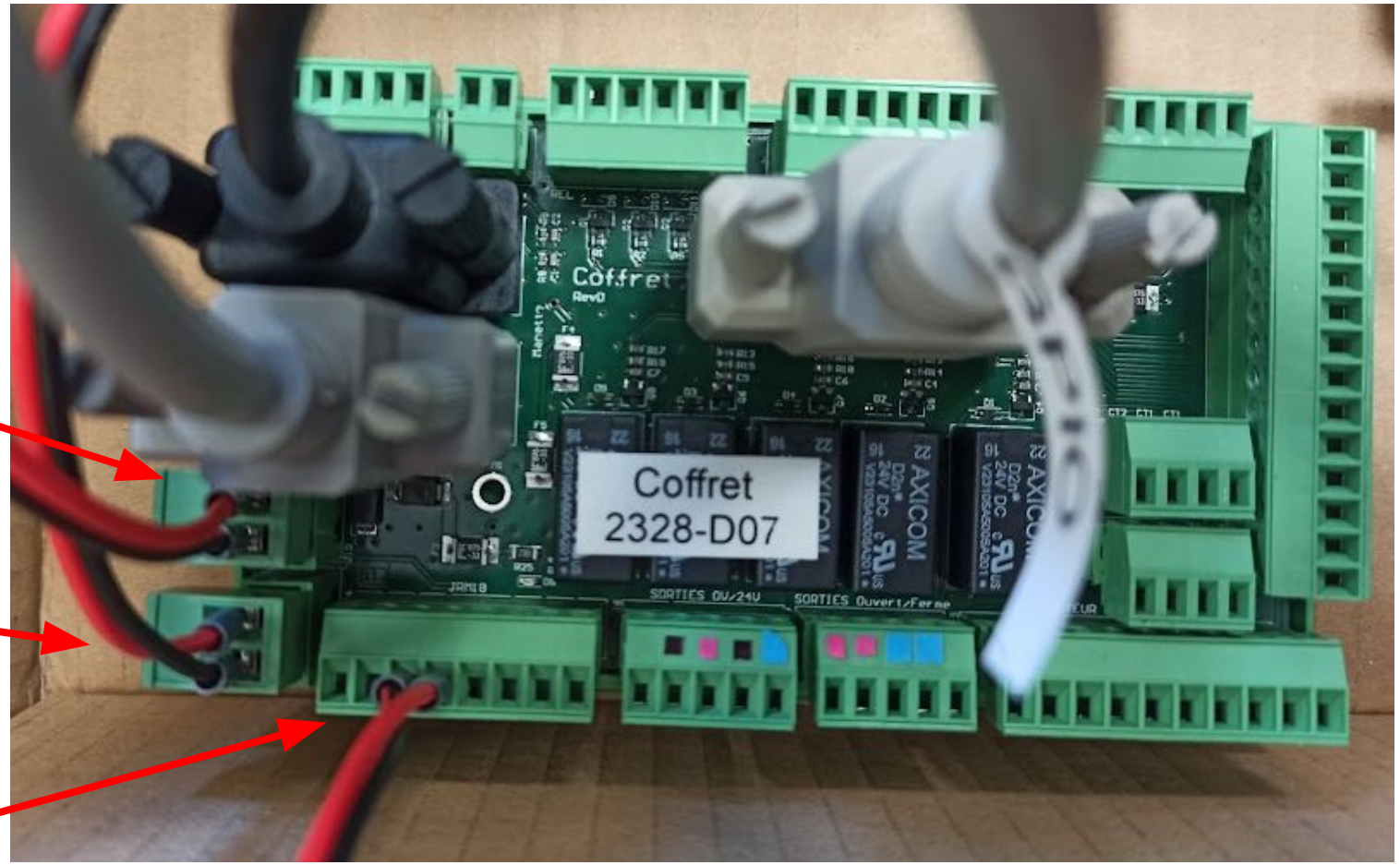


Photo de la carte coffret

Entrée alim 24 V
(non fournie)

Sortie alim vers
PC (câble fourni)

Sortie signal de
soudage (contact sec)



WEEZ-U WELDING
YOUR WELDING TOOLS

Communication

Protocole de communication

- Connexion en **réseau Ethernet** du PC Aaeon et du PC Nimbl'bot
- Protocole de communication **RPC** entre Telesoud (PC Aaeon) et Application Nimbl'bot
- **Livrables** permettant d'établir une communication entre Telesoud et un noeud ROS2 fourni par Weez-U-Welding.
    - **Dockerfile** explicitant les dépendances nécessaires au fonctionnement de la communication (nouvelles dépendances à ajouter dans l'environnement dans lequel tourne l'application Nimbl'bot)
    - **Noeud ROS2 minimal fonctionnel** permettant de communiquer avec Telesoud.

Configuration de la liaison Telesoud <--> Robot

- **Ouvrir une session dans le PC Telesoud fourni par Weez-U-Welding**
  - Connecter un **clavier et souris** sur les ports USB du PC Telesoud
  - Faire **Ctrl + Alt + F2**
  - Se logger avec le user **weez-u-welding** mot de passe **0000**
  - Ouvrir un terminal
  - Se connecter sur le user **runtime** avec la commande **sudo su runtime** (pas de mot de passe requis)
  - Ouvrir le fichier **Telesoud/_runtime/user-config/welding-config.yml** avec un editeur
  - Vérifiez que la ligne **robot** désigne bien **Weezlite** comme robot cible
    - **robot:**
      - **type: Weezlite**
  - Renseignez l'adresse IP de votre robot une fois la connexion Ethernet établie avec le PC de votre robot
    - **weezliteConfig:**
      - loopPeriod: 25
      - tcpConnectionConfig:
      - **ipAddress: xxx.xx.xx.xx**

WEEZ-U WELDING
YOUR WELDING TOOLS

Software – Données descendant vers le robot

Message RPC **TELESOUD —-> ROBOT**

- **InstructionCode:**
  - Type**: uint16_t**
  - **Valeurs possibles (objectif POC) :**
    - **0** ==========> STOP
    - **1** ==========> GET ROBOT DATA
    - **7** ==========> SET DYNAMIC CARTESIAN SPEED
    - **8** ==========> START DYNAMIC CARTESIAN MOVEMENT
    - **15** ==========> PLAY CARTESIAN TRAJECTORY
    - **16** ==========> PLAY JOINT TRAJECTORY
    - **21** ==========> ABORT ALL ERROR
- **pose1:**
  - Type: **tableau de 6 doubles**
  - Information: Données nécessaires à l'exécution d'un mouvement. Uniquement lors des instruction codes
    - PLAY CARTESIAN TRAJECTORY et PLAY JOINT TRAJECTORY
      - "pose1" communique les coordonnées cible de l'effecteur en coordonnées **XYZWPR**
        - XYZ en mètres
        - WPR en degrés
- **pose2, pose3:**
  - Type: **2 x tableau de 6 doubles**
  - **Non utilisé**

Structure complète visible dans code fourni:

InstructionToRobotCode.hpp

```
  STOP                            =0, // stop all movements
  GET_ROBOT_DATA                  =1, // get fresh robot data (used to monitor position)
  SET_DYNAMIC_CARTESIAN_SPEED     =7, // TCP (end-effector or twist) speed vector
  START_DYNAMIC_CARTESIAN_MOVEMENT=8, // start the cartisian movement of the end-effector
  PLAY_CARTESIAN_TRAJECTORY       =15,// start a cartesian trajectory
  PLAY_JOINT_TRAJECTORY           =16,// start a joint trajectory
  SET_TCP_COORDINATES             =19,
  ABORT_ALL_ERRORS                =21,// abort errors order
  ACTIVATE_CLEAR_SIGNAL           =22
};
```

Software – Données descendant vers le robot

- **speedVector:**
  - Type: **tableau de 6 doubles**
  - Information: Vitesse cible de l'effecteur en coordonnées **XYZWPR**. Uniquement dans le cas du DYNAMIC CARTESIAN MOVEMENT.
- **speed:**
  - Type: **double**
  - Information: Vitesse cible de l'effecteur en coordonnées **XYZWPR** lors d'un mouvement de type PLAY CARTESIAN TRAJECTORY ou PLAY JOINT TRAJECTORY
- **bouton free drive:**
  - Type: **booléen**
  - **Non utilisé**

```
constexpr const uint8_t NUMBER_OF_AXIS = 6;

struct InstructionFromTelesoudToRobot {
  uint16_t instructionCode = static_cast<uint16_t>(InstructionCode::STOP);
  std::array<double, NUMBER_OF_AXIS> pose1{0.0, 0.0, 0.0, 0.0, 0.0, 0.0}; // xyzwpr or joints value
  std::array<double, NUMBER_OF_AXIS> pose2{0.0, 0.0, 0.0, 0.0, 0.0, 0.0}; // xyzwpr or joints value
  std::array<double, NUMBER_OF_AXIS> pose3{0.0, 0.0, 0.0, 0.0, 0.0, 0.0};    // xyzwpr or joints value
  std::array<double, DIMENSION_OF_SPEED_VECTOR> speedVector{0.0, 0.0, 0.0, 0.0, 0.0, 0.0}; // end effector speed vector
  double speed = 0.0; // speed of goToPosition instruction
  bool freeDriveBtnStatus = false;
  MSGPACK_DEFINE_ARRAY(instructionCode, pose1, pose2, pose3, speedVector, speed, freeDriveBtnStatus)
};
```

Software – Données descendant vers le robot

USE CASES:

- **STOP :**
  - En dehors des moments d'exécution des commandes de mouvement (SET DYNAMIC CARTESIAN SPEED, START DYNAMIC CARTESIAN MOVEMENT, PLAY CARTESIAN TRAJECTORY, PLAY JOINT TRAJECTORY).
  - La commande STOP est envoyée en boucle au robot.

- **MOUVEMENT CARTÉSIEN DYNAMIQUE:** dit "réglage fin" ou "soudage"
  - Deux commandes sont envoyés via l'API Telesoud vers le Robot:
    - D'abord **START_DYNAMIC_CARTESIAN_MOVEMENT**
    - Puis **SET_DYNAMIC_CARTESIAN_SPEED** pour les éventuels corrections du vecteur vitesse

- Déplacement **PREMIER (direct) et DERNIER (direct):** Commande API **PLAY JOINT TRAJECTORY**

- Déplacements **PREMIER** et **DERNIER**:
  - Enchaînement de 3 mouvements:
    - Dégagement de la torche: Commande API **PLAY JOINT TRAJECTORY**
    - Trajectoire jusqu'au point désiré en trajectoire dégagée: Commande API **PLAY CARTESIAN TRAJECTORY**
    - Approche vers le point cible: Commande API **PLAY JOINT TRAJECTORY**

Software – Données ascendant depuis le robot

Message RPC **ROBOT —-> TELESOUD**

- **robotInFaultStatus:**
  - Type: **booléen**
  - Information: Robot en erreur.
- **robotInSlaveModeStatus:**
  - Type: **booléen**
  - Information: Robot en mode esclave.
- **robotPose:**
  - Type: **tableau de 6 doubles**
  - Information: communique la position courante de l'effecteur en coordonnées **XYZWPR**
- **robotJoints:**
  - Type: **tableau de 6 doubles**
  - **Information:** communique la position courante de axes en **degrés**.
- **errorsAsString:**
  - Type: **string**
  - Information: Message d'erreur associé à la valeur "true" dans **robotInFaultStatus**
- **collisionStatus**
  - Type: **booléen**
  - Information: Etat "collision" du robot.
- **emergencyStop:**
  - Type: **booléen**
  - Information: Etat de l'arrêt d'urgence

Software – Données ascendant depuis le robot

- **weldingTrigger_PlcSignal:**
  - Type: **booléen**
  - Information:
- **operatinMode**
  - Type: **intéger**
  - Information: Mode opératoire soudage. Valeurs autorisées
    - 0 = Auto
    - 1 = 4T
    - 2 = 2T

Merci de votre attention

WEEZ-U WELDING

YOUR WELDING TOOLS

101 rue de Coulmiers | 44 000 | NANTES