Documentation Projet Pont IP-Serie pour boucle d'asservissement.

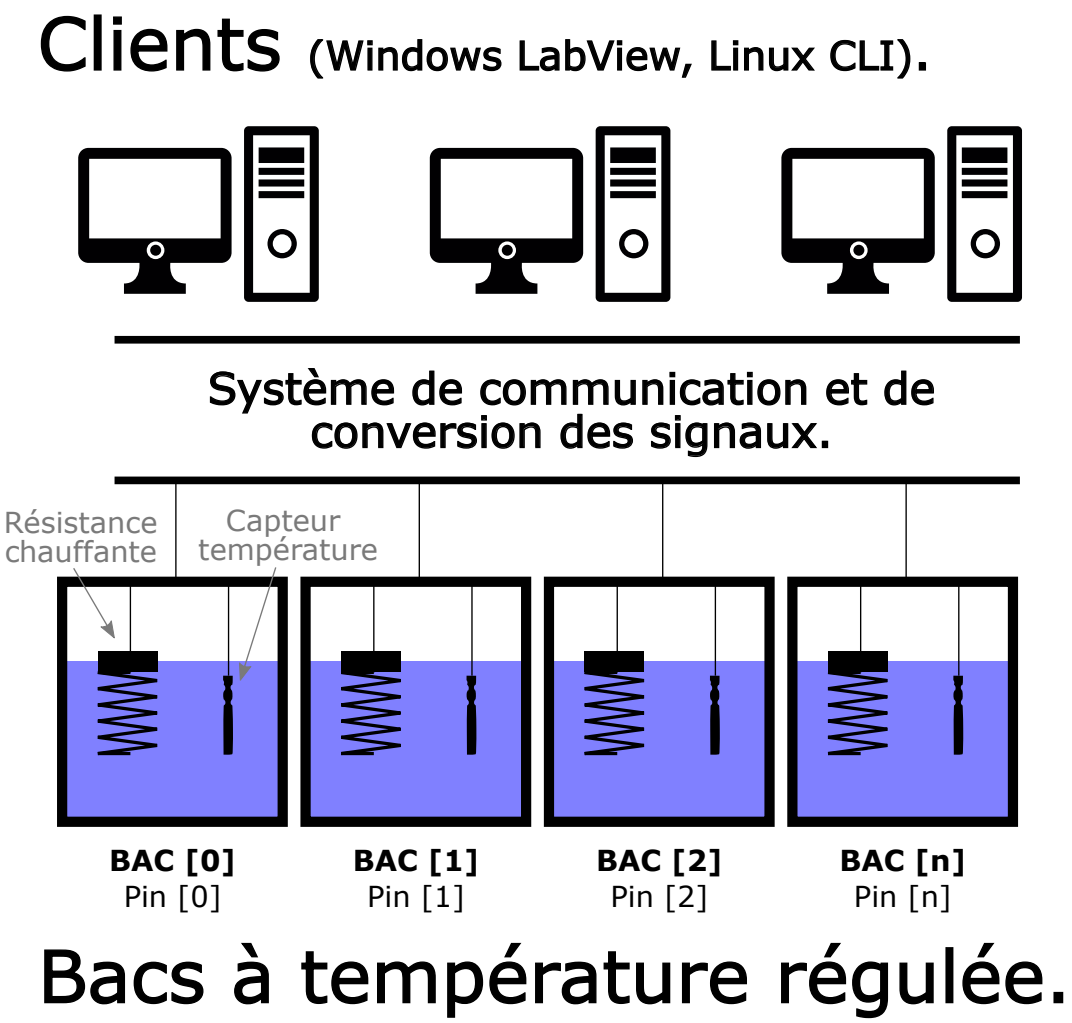
COUTAUD Ulysse

<2022-04-08 Fri>

Contents

1 Intro

- Le système présenté dans ce document est l'interface de communication pour l'asservissement en température d'un parc de bac d'eau:



- Le système permet au *Client* :
  - D'envoyer des requêtes pour déclencher une lecture de la valeur d'un capteur, et de recevoir la valeur lue.
  - D'envoyer des requêtes pour inscrire une valeur de commande sur un actionneur.

2 Installation

2.1 Installation de l'environnement

2.1.1 Télécharger le projet

1. En ligne de commande:
   - Télécharger le dépot git (permet d'avoir les mise à jour via simple "git pull":
     - Soit : git clone *https*://github.com/coutaudu/CommandLabviewRpiArduino.git
     - Soit : git clone ssh://git@github.com/coutaudu/CommandLabviewRpiArduino.git
   - Télécharger juste les sources:
     - wget `https://github.com/coutaudu/CommandLabviewRpiArduino/archive/refs/heads/main.zip`
   - Si besoin:
     - sudo apt-get install git
     - sudo apt-get install wget
2. Via un navigateur web:
   - Page `https://github.com/coutaudu/CommandLabviewRpiArduino`

2.1.2 Installer le projet

1. Les contrôleurs Le projet est développé et tester avec des micro-contrôleur *Arduino Uno*. Il est possible d'adapter pour d'autres types de contrôleur, auquel cas il faut développer le code spécifique et implémenter les fonctions décritent en section 4.

   (a) Flasher un UID sur chacun des contrôleurs du système
   - Il est IMPERATIF d'inscrire un UID dans chacun des contrôleurs utilisés dans le système (nécessaire pour la détection puis la redirection des requêtes vers le bon contrôleur).
   - Pour cela il faut utiliser la fonction *int setBoardUID(unsigned char uid);*:
     - Décommenter la ligne dans la fonction *setup* en inscrivant l'UID voulu.
     - Compiler puis flasher le contrôleur. L'UID a été gravé dans l'EEPROM.
     - Recommenter la ligne, compiler et flasher pour ne pas re-graver l'UID par la suite.

   (b) Flasher le firmware sur chacun des contrôleurs du système
   - Le firmware doit être flashé sur chacun des contrôleurs.
     - Testé avec Arduino IDE 1.8.19.
     - Carte Arduino AVR Board > Arduino UNO.
     - Utilise librairie "EEPROM.h" (built-in).
   - Les contrôleurs sont branchés en USB.
     - Une fois branché, le Serveur doit les détecter: */dev/ttyACM0* et */dev/ttyACM0*.
     - Les paramêtres de communication série sont:
       * 115200 bauds

    * 8 bits de données
    * 0 bits de parité
    * 1 bit de stop
    * binaire nu (toutes les valeurs de contrôles soft, telles que XonXoff sont désactivées).

2. Le serveur

    - Installer l'environnement: rpi@raspberry:~/CommandLabViewRpiArduino $ *sudo ./install-required-software.sh*
    - Compiler le code du serveur: rpi@raspberry:~/CommandLabViewRpiArduino/Serveur $ make
    - Le Serveur peut être lancé soit comme un executable "classique" (./SerialServeur), soit être installer et exécuter comme un service.
    - Le fichier de configuration SerialServeur.conf contient les paramêtres du service. A adapter si besoin de modifier des paramètres, notamment le nom d'utilisateur par défaut (rpi).
    - Installer le service: via les commandes *make*:
        - *make* $install_{service}$ pour installer le service.
        - *make* $run_{service}$ pour lancer manuellement le service:
        - $stop_{service}$ pour stopper le service.
        - *make* $auto_{runservice}$ pour lancer automatiquement le service à la mise sous tension.
        - *make* $disable_{service}$ pour désactiver le démarrage automatique.

3. Les clients

    (a) Client CLI

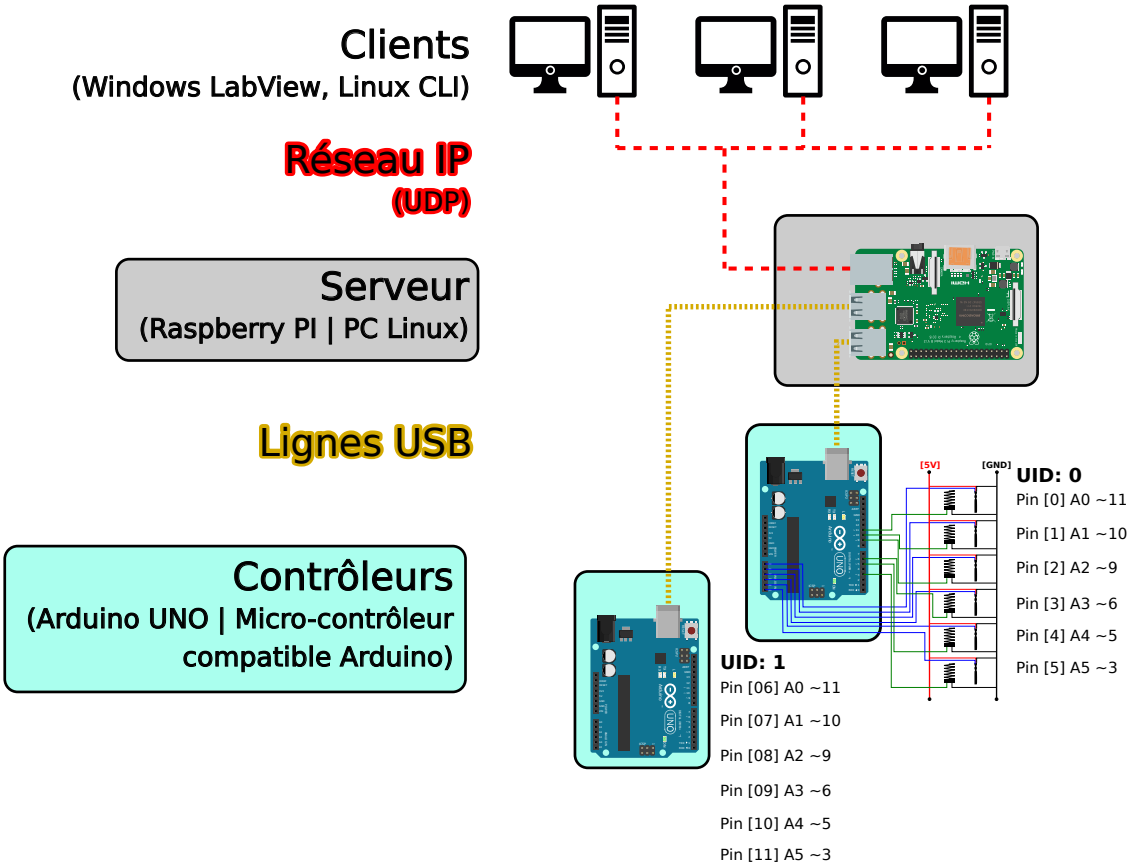
        - Configurer IP et port destination (du serveur donc) dans le code source.
        - Compiler via la commande *make*.
        - Executer *./Client*

    (b) Clients Labview

        - Installer Labview 21.
        - Ouvrir le projet.
        - Choisir le programme d'exemple souhaité.
        - Executer.

3 Architecture

3.1 Matérielle



3.2 Architecture réseau

- Les clients communiquent avec le serveur en UDP/IP.
- Le serveur communique avec les contrôleurs en USB.
- Le protocole applicatif [TODO NOM DE PROTOCOLE] est un développement spécifique au projet. Le protocole est décrit en section 4.
- Le serveur fait :
  - Pont UDP<->USB.
  - Routage N°Pin<->contrôleur (UID).

3.3 Architecture logicielle

- Le code du projet est découpé en 3 modules:
  - **Arduino** qui contient le firmware des contrôleurs.
  - **Serveur** qui contient le code pour la communication réseau USB/UDP (Raspberry PI/ Linux).

  - **Client** qui contient des exemples d'application client utilisant le système:
    * CLI: Programme minimal codé en C avec interface en ligne de commande pour envoyer et recevoir des requetes.
    * GUI: Panel de fonction LabView faisant office de bibliothèque d'utilisation du système de communication avec des exemples d'applications graphiques.

- Le fichier d'entête Arduino/shared.h contient les définitions des types du protocole de communication. Il est donc partagé entre le code du firmware des contrôleurs et le code du serveur.

- Le code doit être sujet à une passe de nettoyage et de structuration. Je ne met donc pas de détails sur la structure du code dans sont implémentation actuelle qui doit être considérée comme un *Proof Of Concept*.

4 Protocole

- Le protocole est basé sur le principe de *requête/réponse*.
  - 1 *requête* implique 1 *réponse* de même code fonction.
  - Les clients ou le serveur pilotent la communication: ils envoient les requêtes.
  - Les contrôleurs sont esclaves de la communication: ils reçoivent les requêtes et y répondent pas des réponses.
- Les requêtes et les réponses ont le même format.

Numéro de version	Code fonction	Argument [0]	Argument [1]
1 octet	1 octet	1 octet	1 octet

- Le protocole est un protocole de niveau applicatif dans le modèle en couche OSI.
- Le protocole fonctionne par paquet de 4 octets qui doivent pouvoir être envoyés entre les clients et le serveur et entre le serveur et les contrôleurs.
- Le protocole est neutre vis à vis des couches réseaux inférieures utilisées pour transmettre ses paquets. L'implémentation actuelle utilise UDP/IP entre les clients et le serveur, et USB entre le serveur et les contrôleurs.
- Le protocole ne fournit pas de garantie de qualité de service (QoS), ni en termes de latence, ni d'intégrité des données, ni d'ordonnancement. En cas de besoin de besoin QoS, il faut utiliser des protocoles réseaux fournissant ces garanties dans les couches réseaux inférieures.

4.1 Version du protocole

- Version 0 = erreur.
- Version 1 = version actuelle.
- Version 2-255 = RFU (Reserved for Future Use). Le numéro de version est vérifié lors des communications entre contrôleurs et serveur doit être cohérents. Le numéro de version doit être incrémenter en cas d'ajout ou de modification des fonctions implémentées.

4.2 Les fonctions

- 0 = Erreur.
- 1 = Lecture analogue.
- 2 = Ecriture digitale PWM.
- 3 = Lecture UID contrôleur.
- 4-255 = RFU.

4.2.1 0 - Erreur

- Le code fonction 0 indique une erreur.
  - Par exemple: lecture sur un numéro de pin inconnu, réponse à un code fonction inconnu.
  - Une requête de code fonction *erreur* doit recevoir une réponse de code fonciton erreur.

4.2.2 1 - Lecture analogue

- Le code fonction 1 indique une lecture analogue.
- Requête:
  - Argument[0]: numéro de pin cible (voir schémas en section 3.1 et 1).
  - Argument[1]: non utilisé.
- Réponse:
  - Argument contient la valeur analogue lut par le contrôleur (valeur entre 0 et 1024 au format unsigned int sur 2 octets dans le cas *Arduino Uno*).

4.2.3 2 - Ecriture digitale PWM

- Le code fonction 2 indique une écriture digitale sur une pin en PWM.
- Requête:
  - Argument[0]: numéro de pin cible (voir schémas en section 3.1 et 1).
  - Argument[1]: valeur à affecter entre 0 et 255 (unsigned char).
- Réponse:
  - Argument[0]: numéro de pin cible (voir schémas en section 3.1 et 1).
  - Argument[1]: valeur affectée entre 0 et 255 (unsigned char).

4.2.4 3 - Lecture UID contrôleur

- Le code fonction 3 indique la lecture de l'UID du contrôleur.
- Requête:
  - Argument[0]: non utilisé.
  - Argument[1]: non utilisé.
- Requête:
  - Argument[0]: UID du contrôleur (valeur en 0 et 255 unsigned char).
  - Argument[1]: non utilisé.

4.2.5 4-255 - RFU

Les codes fonctions 4 à 255 sont libres pour l'ajout de nouvelles fonctions.

5 Guide d'utilisation

TODO

6 Améliorations

- ☐ Restructuration et nettoyage du code.
- ☐ Log rotatifs.
- ☐ Passage des arguments du serveur en ligne de commande.
- ☐ Passage des arguments du serveur en fichier etc/xml/json/csv.
- ☐ Installer serveur smb sur serveur pour accès au fichier de configuration
- ☐ Mesures des temps d'execution.
- ☐ Documentation.
- ☐ WebViewer