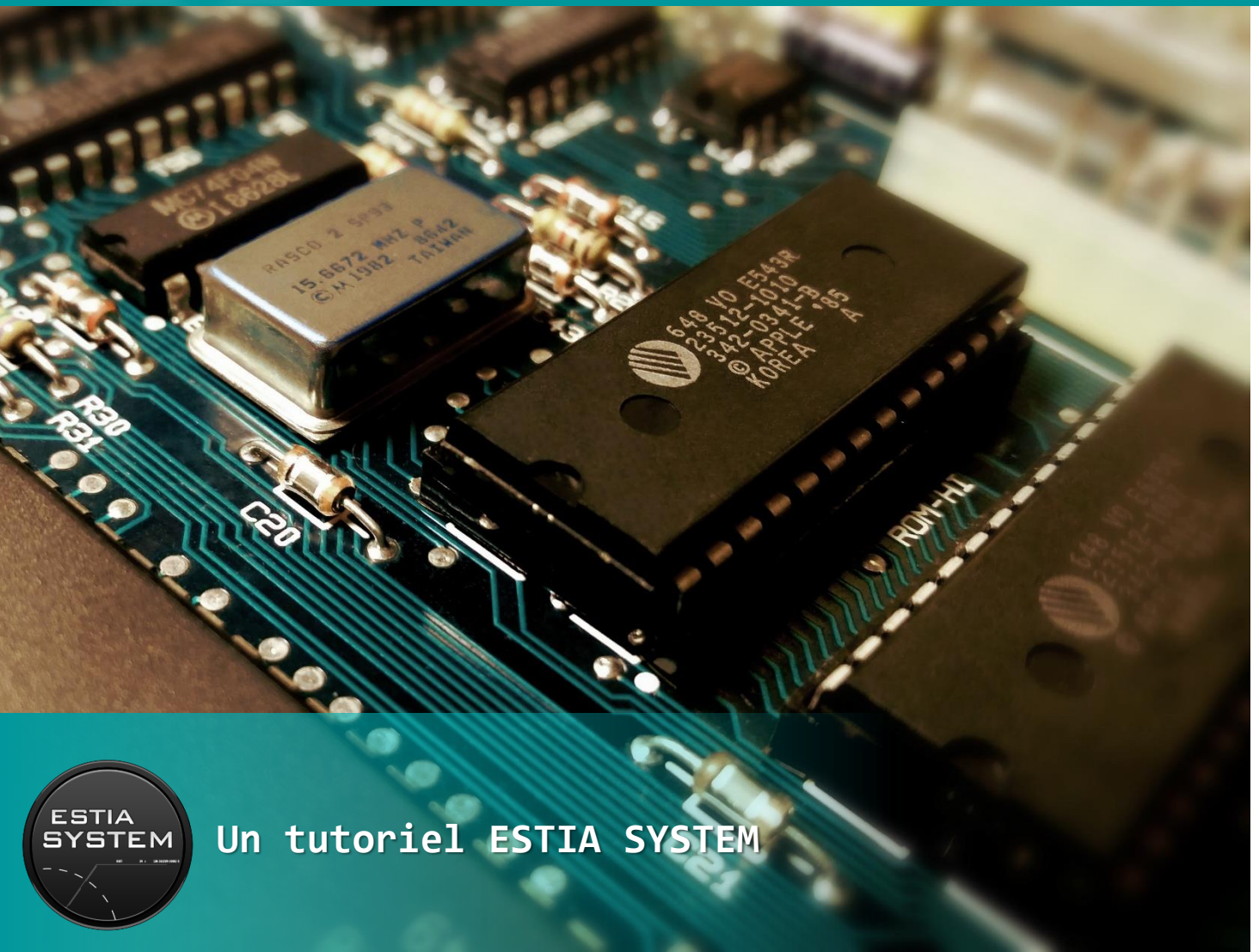
SERVOMOTEUR DYNAMIXEL 2XL430 ET ARDUINO

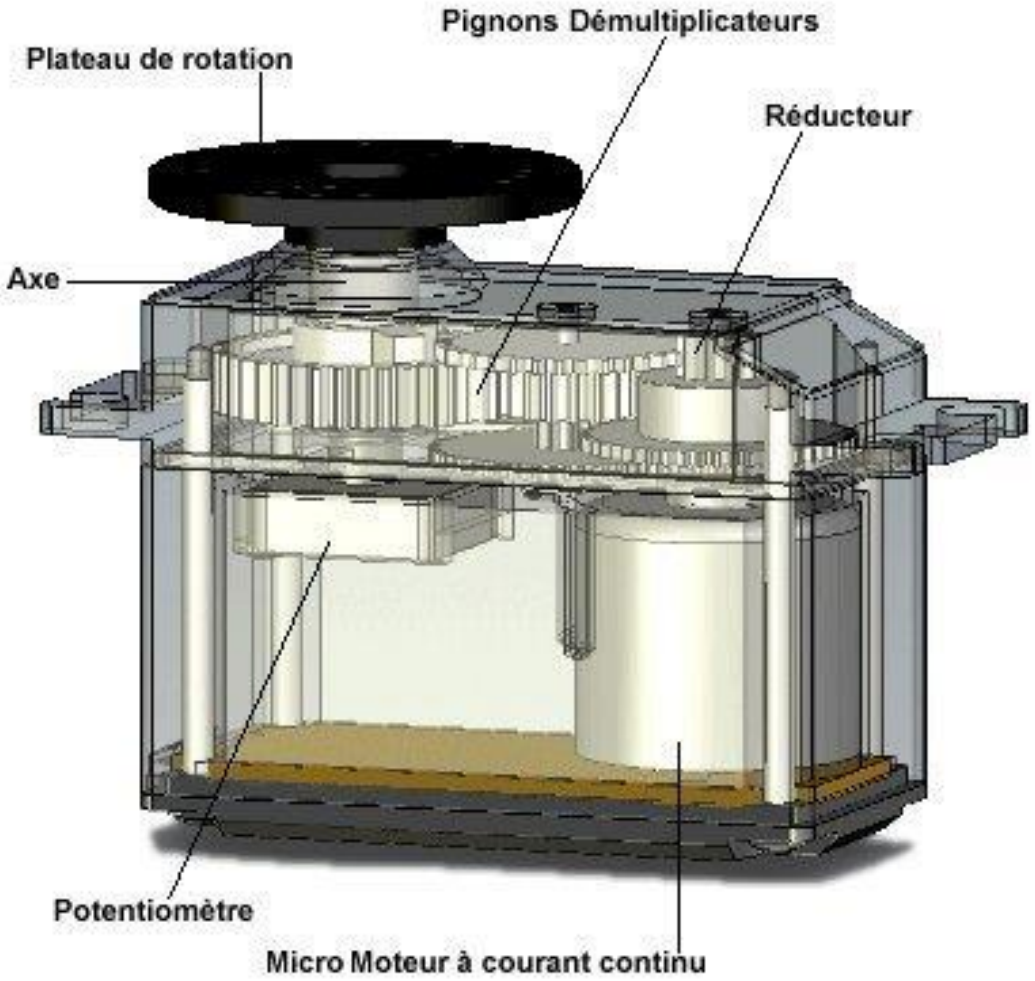


ESTIA SYSTEM

Un tutoriel ESTIA SYSTEM

INTRODUCTION

Définition
Vous avez dit… Servomoteur?



C’est un moteur ‘‘intelligent’’ composé de :

- Un moteur à courant continu avec son réducteur
- Un capteur de position angulaire (potentiomètre)
- Une carte électronique de contrôle et pilotage du moteur

Il a l’avantage d’être asservi en position angulaire (il respectera la consigne envoyée même s’il y a un obstacle).

Grâce à la librairie Dynamixel2Arduino dans Arduino, nous pouvons lui donner directement différents ordres et notamment un ordre de position angulaire.

Eric G

Définition
Vous avez dit… Servomoteur?

Quelques constructeurs connus:

- Robotis: La gamme des Dynamixels
- Tower Pro: La gamme des servomoteurs 9g
- Hitec
- Etc…

Les différents types de servomoteurs Dynamixel

Contient 2 servomoteurs XL430 !

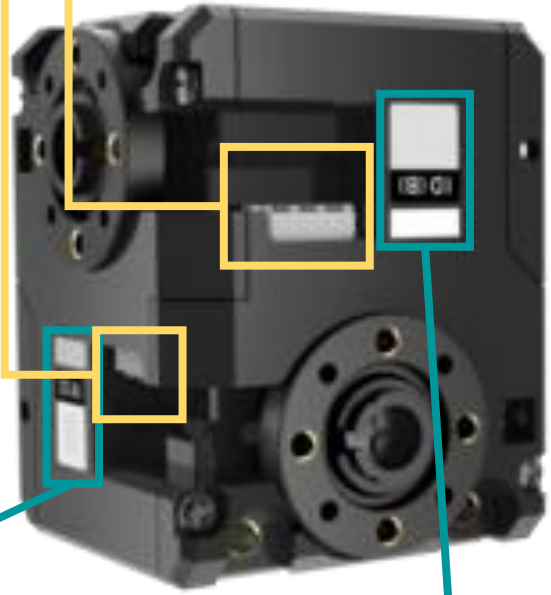
AX RX MX … 2XL …

Ouais mais on le fait fonctionner comment concrètement?

Architecture d’un servomoteur Dynamixel 2XL430

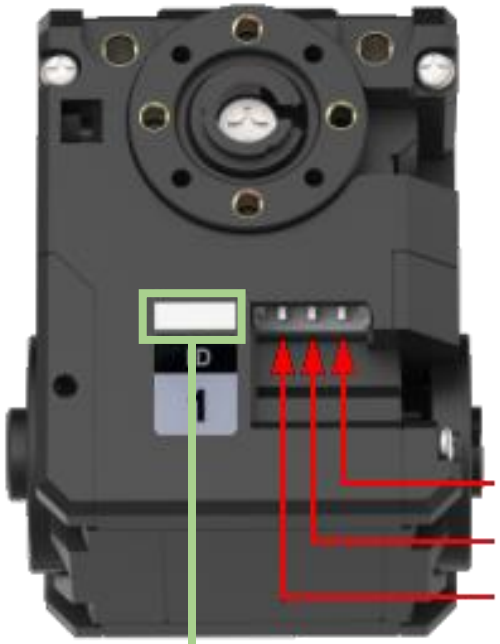
Pins de branchement (2 triplettes pour brancher les servos en série)

On peut s’adresser à tous les servos en même temps avec l’Id 254



Identifiant du servomoteur A (vaut 1 par défaut)

Identifiant du servomoteur B (vaut 2 par défaut)



Alimentation (7-12V DC) 11,1 V recommandé

PIN1 : GND

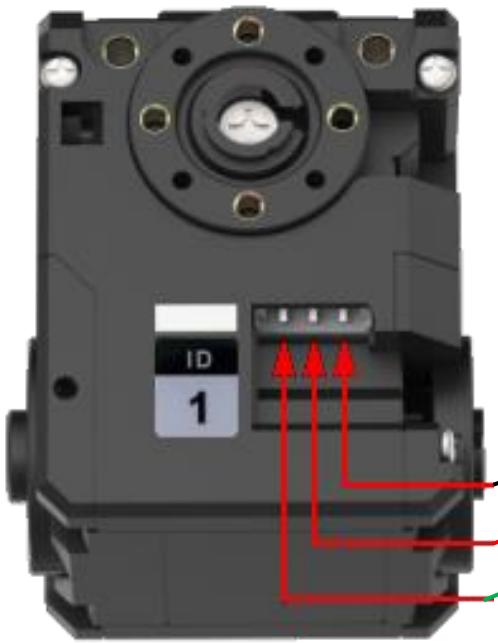
PIN2 : VDD

PIN3 : Data

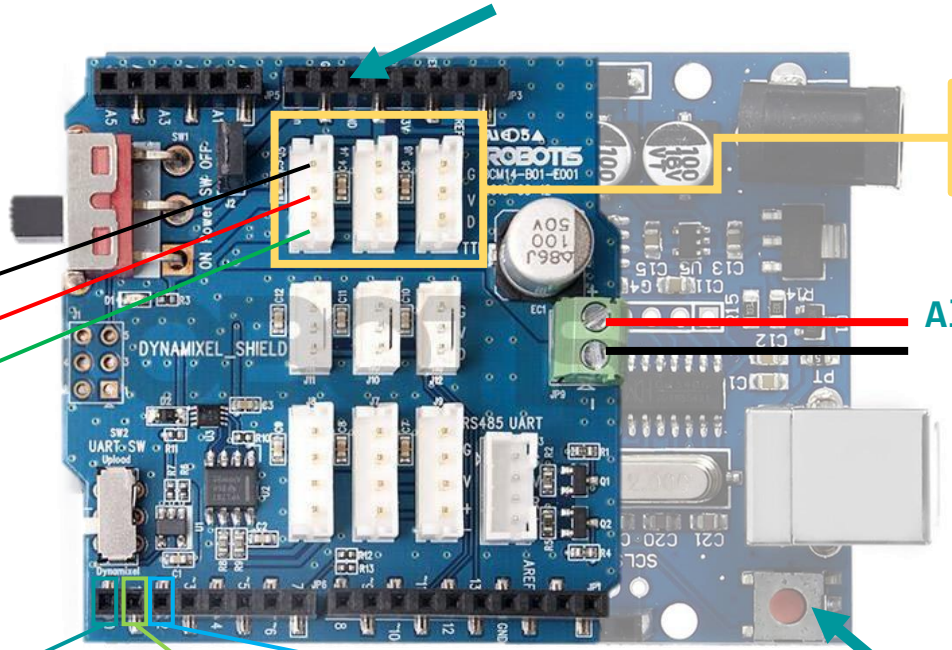
Vers l’Arduino

LED rouge qui peut s’allumer pour des tests basiques

Exemple de câblage d'un 2XL430



Shield Dynamixel pour Arduino avec lequel nous pouvons contrôler plusieurs types de Dynamixels



Broches de communication TTL

Alimentation (11,1V DC)

Arduino Uno

DXL_RX(0) — Réception d'ordres

DXL_TX(1) — Émission d'ordres

DXL_DIR(2) — Switch émission/ réception

→ Car les data du servomoteur passent par 1 seul câble

Détails sur le 2XL430-W250-T

- Fonctionnement à un certain BaudRate (équivalent à une fréquence en bit/s): **57 600** par défaut. Les 2 servomoteurs partage **le même BaudRate** !

- Utilisation de la librairie **Dynamixel2Arduino** pour le faire fonctionner correctement.

- Ne pas mettre le même Id aux 2 servomoteurs !

- Valeurs de position angulaire en degrés ou uint.

- Valeurs de vitesse angulaire en RPM (tour/mn) ou pourcentage ou uint.

- On peut s’adresser à un groupe de servomoteurs en donnant un second Id (caché) à chaque servomoteur qui sera identique pour tous les servomoteurs du groupe.

- Les deux servomoteurs fonctionnent **indépendamment**, excepté pour le BaudRate et le reset (Un reset réinitialise les deux aux paramètres d’usine !!)

Fonctions de la Librairie Dynamixel2Arduino

- **Dynamixel2Arduino dxl(DXL_SERIAL, DXL_DIR_PIN)** : Créé un objet nommé dxl de type Dynamixel2Arduino.
  - DXL_SERIAL : Canal utilisé pour communiquer avec le servomoteur (**Serial**, Serial1, Serial2 etc).
  - DXL_DIR_PIN : Pin qui permet de basculer la communication entre émission et réception de données (**pin 2** sur le schéma de câblage).

- **dxl.begin(BaudRate)** : Etablie la communication entre l'Arduino et le servomoteur à un certain BaudRate
  - BaudRate : entier spécifique (voir datasheet en fin de diapo pour les valeurs précises) (souvent **57600**).

- **dxl.setPortProtocolVersion(DXL_PROTOCOL_VERSION)** : Etablie la version du protocole de communication.
  - DXL_PROTOCOL_VERSION : float qui vaut soit 1.0 ou 2.0 (les 2XL430 fonctionnent en **2.0**).

- **dxl.ping(DXL_ID)** : Récupère les informations du servomoteur pour pouvoir identifier son type et adapter la librairie en fonction.
  - DXL_ID : entier (de 0 à 253). Vaut **1** ou **2** suivant le servomoteur par défaut. **254** correspond au BroadCast.

- **dxl.ledOn(DXL_ID)** : Allume la LED correspondant au servomoteur spécifié par son Id. (Inverse pour ledOff)
  - DXL_ID : comme précédemment.

Fonctions de la Librairie Dynamixel2Arduino

- **dxl.setOperatingMode(DXL_ID, mode)** : Sélectionne le mode de fonctionnement du servomoteur.
  - DXL_ID : Comme précédemment.
  - mode : Mode de fonctionnement :
    - **OP_POSITION** : Contrôle de la position (mode par défaut)
    - **OP_EXTENDED_POSITION** : Contrôle de la position en multi-tours (de -256 à 256 tours)
    - **OP_CURRENT_BASED_POSITION** : Contrôle de la position en courant
    - **OP_VELOCITY** : Contrôle de la vitesse de rotation (rotation continue)
    - **OP_PWM** : Contrôle en PWM
    - **OP_CURRENT** : Contrôle du courant

- **dxl.setGoalPosition(DXL_ID, angle, unit)** : Envoi un ordre de position angulaire au servomoteur.
  - DXL_ID : …
  - angle : ordre de position angulaire (un entier de **0 à 4095** ou un float représentant un **angle en degré**).
  - unit : unité de l'ordre :
    - **UNIT_DEGREE** : angle de 0 à 360°
    - **UNIT_RAW** : angle de 0 à 4095

Exemple d'architecture d'un code pour contrôler un servomoteur 2XL430

1. Inclusion des librairies, définition des variables et création d'un objet de type Dynamixel2Arduino

```
#include <Dynamixel2Arduino.h>

#if defined(ARDUINO_AVR_UNO) || defined(ARDUINO_AVR_MEGA2560)
  #include <SoftwareSerial.h>
  SoftwareSerial soft_serial(7, 8); // DYNAMIXELShield UART RX/TX
  #define DXL_SERIAL   Serial
  #define DEBUG_SERIAL soft_serial
  const uint8_t DXL_DIR_PIN = 2; // DYNAMIXEL Shield DIR PIN
#else
  #define DXL_SERIAL   Serial1
  #define DEBUG_SERIAL Serial
  const uint8_t DXL_DIR_PIN = 2; // DYNAMIXEL Shield DIR PIN
#endif

const uint8_t DXL_ID = 1;
const float DXL_PROTOCOL_VERSION = 2.0;

Dynamixel2Arduino dxl(DXL_SERIAL, DXL_DIR_PIN);
```

Si on utilise une Arduino Uno ou Mega, on créer un nouveau canal série pour du debug (afficher des infos).

Création d'un objet **Dynamixel2Arduino** nommé dxl, connecté sur le **Serial** et dont la pin de switch est la n°**2**.

Exemple d'architecture d'un code pour contrôler un servomoteur 2XL430

2. Fonction setup : initialisation, démarrage des communications, sélection du mode de fonctionnement

```
void setup() {
  // put your setup code here, to run once:

  // Use UART port of DYNAMIXEL Shield to debug.
  DEBUG_SERIAL.begin(115200);

  // Set Port baudrate to 57600bps. This has to match
  dxl.begin(57600);
  // Set Port Protocol Version. This has to match with
  dxl.setPortProtocolVersion(DXL_PROTOCOL_VERSION);
  // Get DYNAMIXEL information
  dxl.ping(DXL_ID);

  // Turn off torque when configuring items in EEPROM
  dxl.torqueOff(DXL_ID);
  dxl.setOperatingMode(DXL_ID, OP_POSITION);
  dxl.torqueOn(DXL_ID);
}
```

Sélection du mode de fonctionnement (Penser à désactiver puis réactiver le couple avec ''torqueOff/On'').

Exemple d'architecture d'un code pour contrôler un servomoteur 2XL430

3. Fonction loop : actions que l'on veut effectuer

```
void loop() {
  // put your main code here, to run repeatedly:

  // Please refer to e-Manual(http://emanual.robotis.com/docs/en/parts,
  // Set Goal Position in RAW value
  dxl.setGoalPosition(DXL_ID, 512);
  delay(1000);
  // Print present position in raw value
  DEBUG_SERIAL.print("Present Position(raw) : ");
  DEBUG_SERIAL.println(dxl.getPresentPosition(DXL_ID));
  delay(1000);

  // Set Goal Position in DEGREE value
  dxl.setGoalPosition(DXL_ID, 5.7, UNIT_DEGREE);
  delay(1000);
  // Print present position in degree value
  DEBUG_SERIAL.print("Present Position(degree) : ");
  DEBUG_SERIAL.println(dxl.getPresentPosition(DXL_ID, UNIT_DEGREE));
  delay(1000);
}
```

→ Ordre de position de 0 à 4095

→ Ordre de position en degré

Problèmes possibles et solutions

Parce qu’il y a toujours des problèmes …

Les problèmes et solutions

- Si un servo ne fonctionne pas, toujours revenir à un programme simple (type allumer la LED) pour voir s’il reçoit les ordres.

- Si on ne connaît pas le Baudrate, il faut tester tous ceux qui sont dans la datasheet. On peut ensuite le remettre à 57600 avec un ‘‘setBaudrate’’.

- Si on ne connaît pas l'id du servo, on peut s'adresser à tous en utilisant l'id 254 puis remettre un id avec ‘‘setID’’. Attention, il faut le faire si on a qu'un seul servo !!

Les problèmes et solutions

- Si on veut programmer une carte arduino UNO avec deux types de dynamixels différents, il y aura un problème de librairie, car chaque type de dynamixel a sa propre librairie.

- Pour cela, il faut suivre plusieurs étapes, connaître la version du protocole de chaques dynamixels, son ID mais aussi son baudrate.

- Voici les étapes à suivre :

Les problèmes et solutions

- Si on veut programmer une carte arduino UNO avec deux types de dynamixels différents, il y aura un problème de librairie, car chaque type de dynamixel a sa propre librairie.

- Pour cela, il faut suivre plusieurs étapes, connaître la version du protocole de chaques dynamixels, son ID mais aussi son baudrate. Ensuite il faut modifier le baudrate pour que tous les dynamixels aient le même

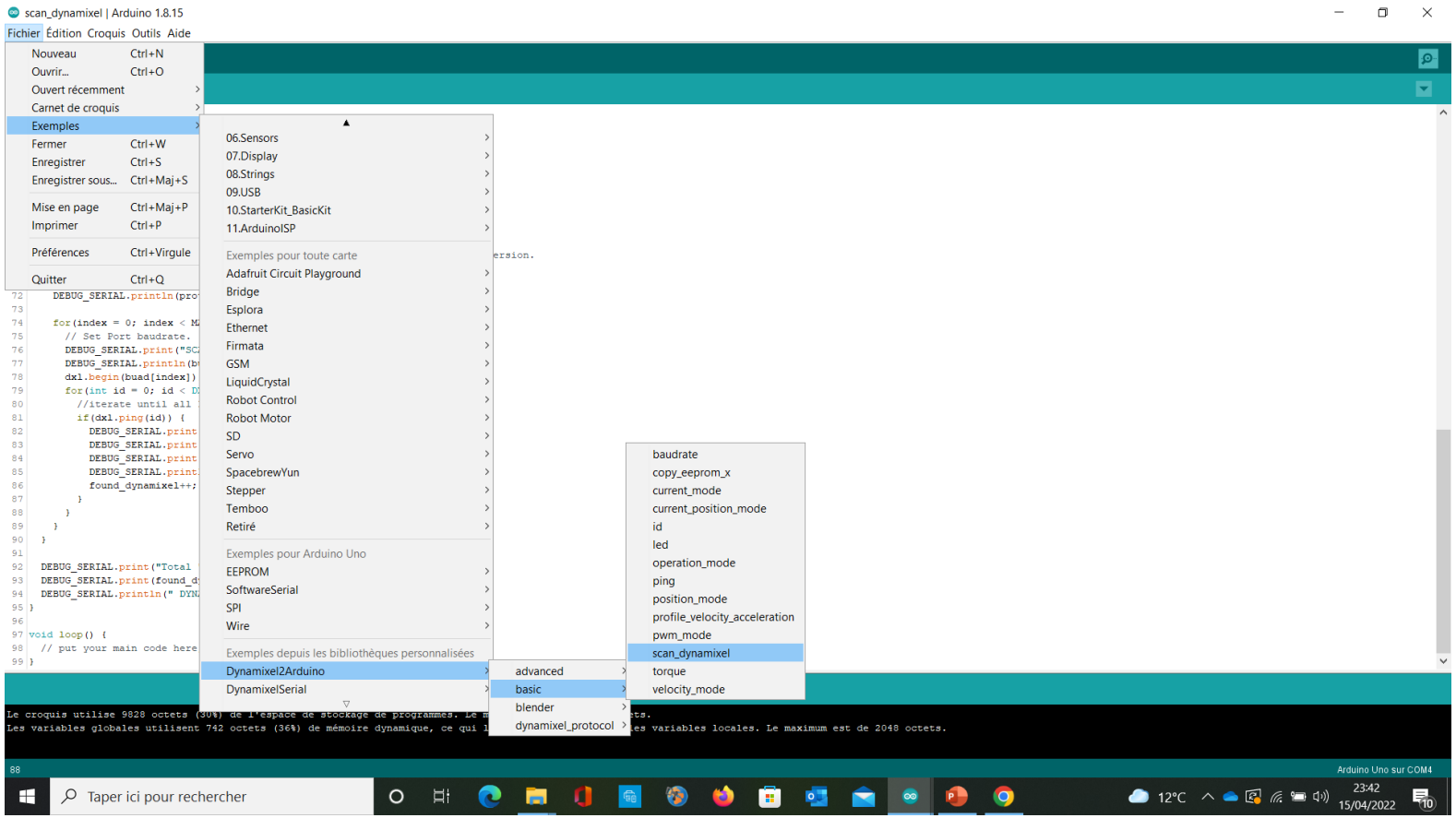
- Voici les étapes à suivre :

Connaître les caractéristiques du dynamixel

- Avoir 2 cartes Arduino UNO, une où est le shield avec le dynamixel branché (Carte 1) et une autre pour recevoir les informations du dynamixel (Carte 2)
- Les deux cartes sont connectées comme suit :
- Les pins 7 et 8 de la carte 1 sont branchés respectivement sur les pins 0 et 1 de la carte 2.
- Compiler le programme « scan_dynamixel » sur la carte 1 (voir diapo 20)
- Ensuite Brancher la carte 2 et lancer le moniteur de série (la loupe en haut à droite), (voir diapo 21)
- Faite un reset sur la carte 1 afin de relancer le programme (bouton rose à côté du port USB)
- Récolter ensuite les informations récoltées sur le moniteur de série
- Refaire le même procédé pour l’autre type de dynamixel

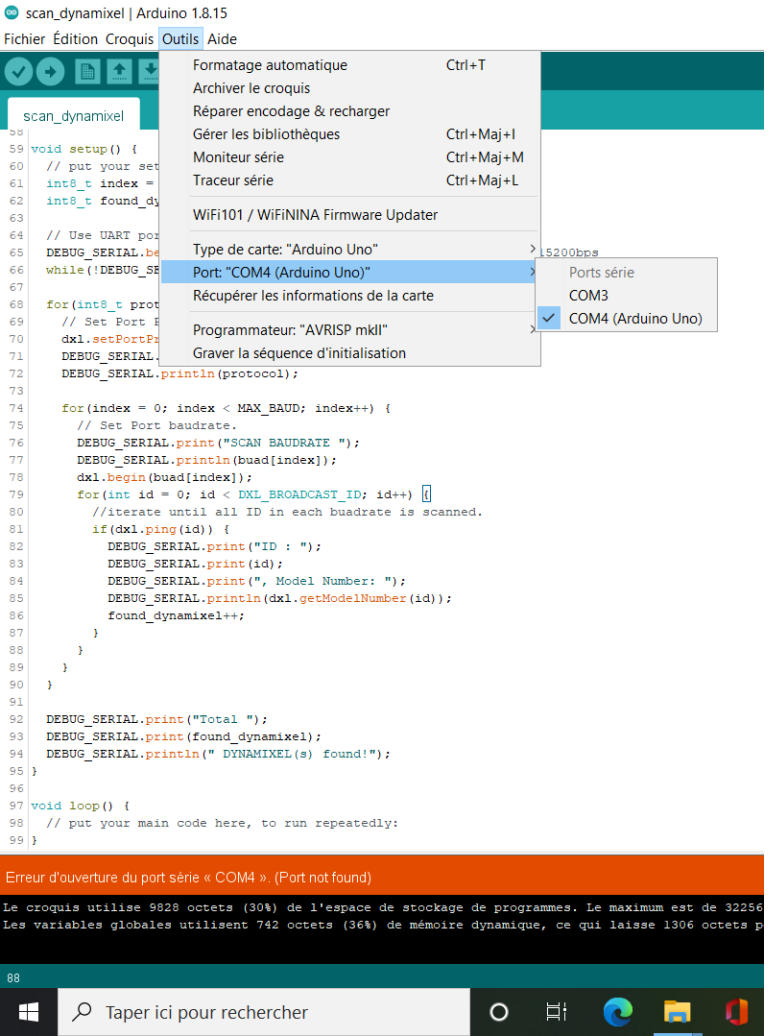
Connaître les caractéristiques du dynamixel

- Compiler le programme « scan_dynamixel » sur la carte 1



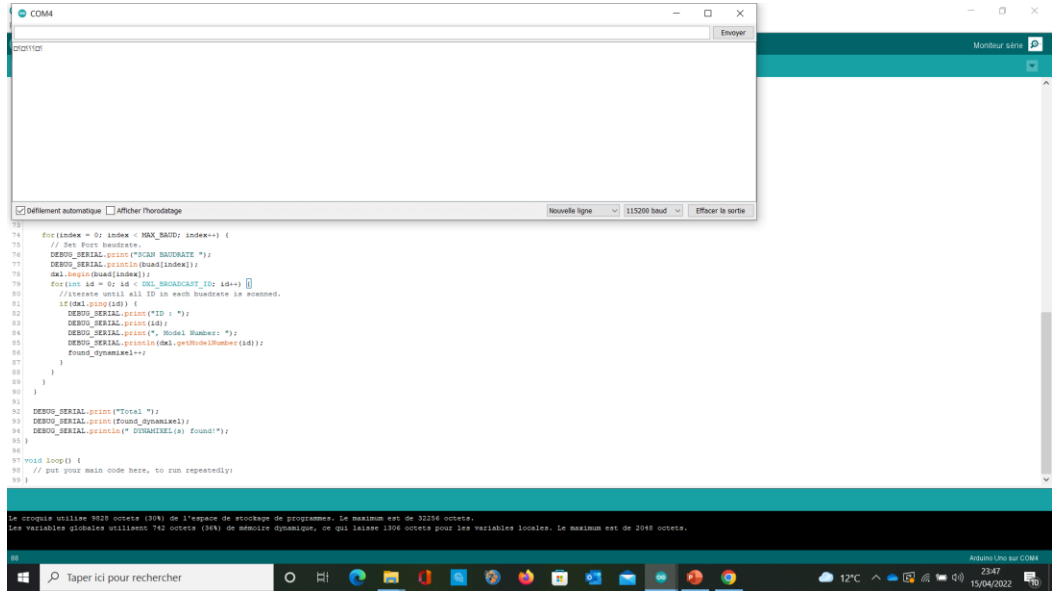
Connaître les caractéristiques du dynamixel

- lancer le moniteur de série (la loupe en haut à droite)



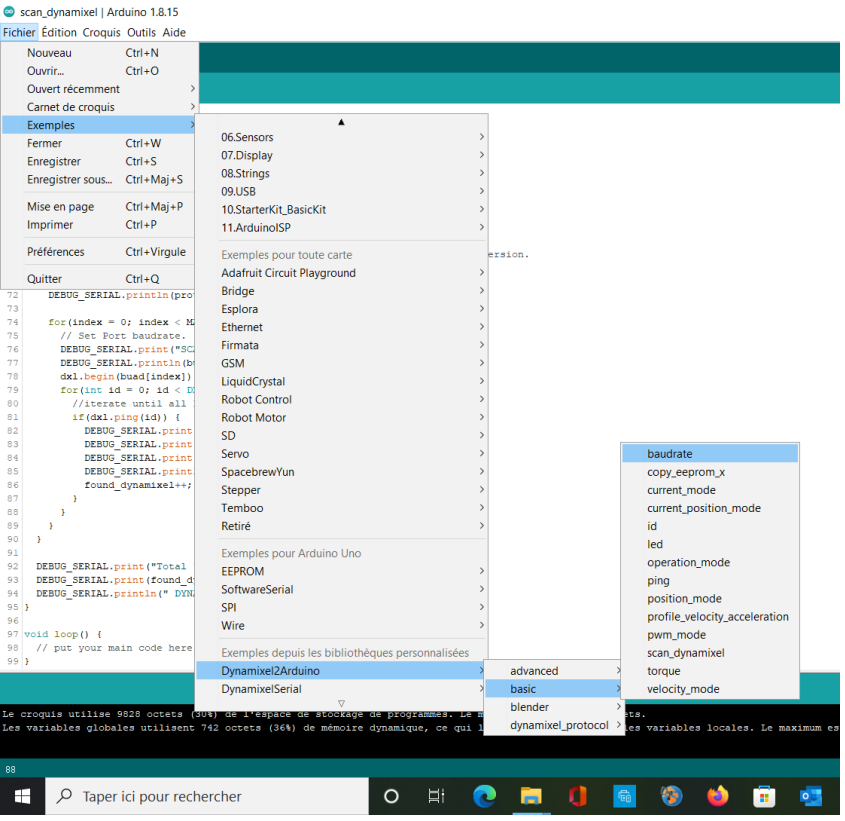
Pensez à bien paramétrer le port de la carte

Une pop-up doit apparaître après avoir lancer le moniteur de série



Connaître les caractéristiques du dynamixel

- Si le baudrate n’est pas le même entre les deux types de dynamixel, changer le baudrate d’un des deux types :



Lancer le programme baudrate

Modifier l’ID en fonction du dynamixel (obtenue par le moniteur de série) et modifier le NEW_BAUDRATE par celui que l’on veut

```
const uint8_t DXL_ID = 1;
const float DXL_PROTOCOL_VERSION = 2.0;
uint32_t BAUDRATE = 57600;
uint32_t NEW_BAUDRATE = 1000000; //1Mbsp
```

Connaître les caractéristiques du dynamixel

- Si le baudrate n’est pas le même entre les deux types de dynamixel, changer le baudrate d’un des deux types :

```
void setup() {
  // put your setup code here, to run once:

  // Use UART port of DYNAMIXEL Shield to debug.
  DEBUG_SERIAL.begin(115200);

  // Set Port baudrate to 57600bps. This has to match with DYNAMIXEL baudrate.
  dxl.begin(BAUDRATE);
  // Set Port Protocol Version. This has to match with DYNAMIXEL protocol version.
  dxl.setPortProtocolVersion(DXL_PROTOCOL_VERSION);

  DEBUG_SERIAL.print("PROTOCOL ");
  DEBUG_SERIAL.print(DXL_PROTOCOL_VERSION, 1);
  DEBUG_SERIAL.print(", ID ");
  DEBUG_SERIAL.print(DXL_ID);
  DEBUG_SERIAL.print(": ");
  if(dxl.ping(DXL_ID) == true) {
    DEBUG_SERIAL.print("ping succeeded!");
    DEBUG_SERIAL.print(", Baudrate: ");
    DEBUG_SERIAL.println(BAUDRATE);

    // Turn off torque when configuring items in EEPROM area
    dxl.torqueOff(DXL_ID);

    // Set a new baudrate(1Mbps) for DYNAMIXEL
    dxl.setBaudrate(DXL_ID, NEW_BAUDRATE);
    DEBUG_SERIAL.println("Baudrate has been successfully changed to 1Mbps");

//    // Change to the new baudrate for communication.
//    dxl.begin(NEW_BAUDRATE);
//    // Change back to the initial baudrate
//    dxl.setBaudrate(DXL_ID, BAUDRATE);
//    DEBUG_SERIAL.println("Baudrate has been successfully changed back to initial baudrate");
  }
  else{
    DEBUG_SERIAL.println("ping failed!");
  }
}

void loop() {
  // put your main code here, to run repeatedly:
```

Mettre en commentaire la partie surligné du programme « baudrate ».

Ensuite compiler le programme dans la carte 1, où est connecté le dynamixel

Connaître les caractéristiques du dynamixel

- Modifier la version du protocole du dynamixel :

```
void loop() {
  // put your main code here, to run repeatedly:

 // Set Port Protocol Version. This has to match with DYNAMIXEL protocol version.
  dxl.setPortProtocolVersion(2.0);

  // Turn on the LED on DYNAMIXEL
  dxl.ledOn(DXL_ID);
  delay(500);
  // Turn off the LED on DYNAMIXEL
  dxl.ledOff(DXL_ID);
  delay(500);

   // Set Port Protocol Version. This has to match with DYNAMIXEL protocol version.
  dxl.setPortProtocolVersion(1.0);

  // Turn on the LED on DYNAMIXEL
  dxl.ledOn(DXL_ID2);
  delay(500);
  // Turn off the LED on DYNAMIXEL
  dxl.ledOff(DXL_ID2);
  delay(500);

}
```

Type 1

Type 2

Il faut spécifier la version du protocole que nous devons utiliser pour chaque action en fonction du type du dynamixel

Ici, le Type 1 utilise la version 2.0 alors que le Type 2 utilise la version 1.0

À vous de jouer !

Quoi? C’était pas dans le contrat ça!!!

Les TPs

2 Tps au choix:

- Contrôle d’un servomoteur 2XL430 avec un bouton.
- Contrôle d’un servomoteur 2XL430 avec un potentiomètre.

Du matériel vous est prêté… prenez en soins svp.

N’hésitez pas à poser des questions, à interpeller vos seniors, etc. ;)

Good Luck

Pour en savoir plus

Datasheet du 2XL430-W250-T ici : http://emanual.robotis.com/docs/en/dxl/x/2xl430-w250/

Datasheet du Shield Dynamixel : http://emanual.robotis.com/docs/en/parts/interface/dynamixel_shield/

Ou envoyer un mail à estiasystem@net.estia.fr