Interfaçage de LabVIEW avec une carte Arduino R3 pilotant un système de balayage mécanique

Nikolay SMAGIN[1],

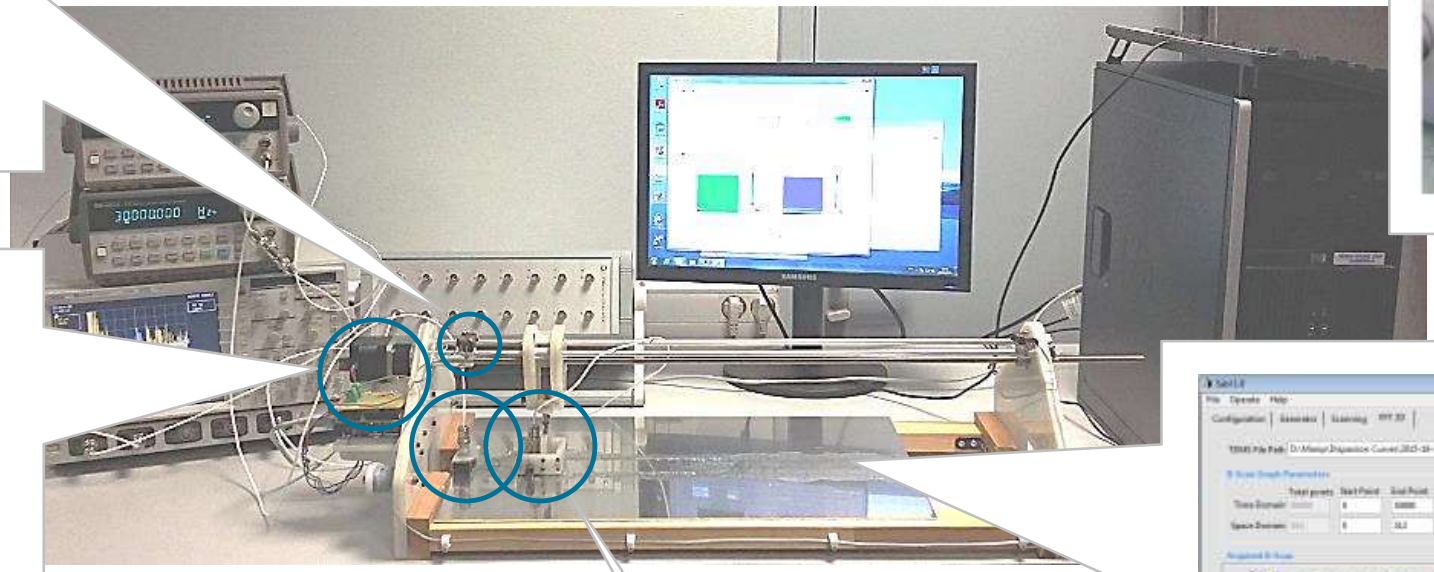
[1]**Univ. Polytechnique Hauts-de-France, CNRS,** Univ. Lille, ISEN, Centrale Lille, **UMR 8520 - IEMN** - Institut d'Électronique de Microélectronique et de Nanotechnologie, DOAE - Département d'Opto-Acousto-Électronique, **F-59313 Valenciennes, France**

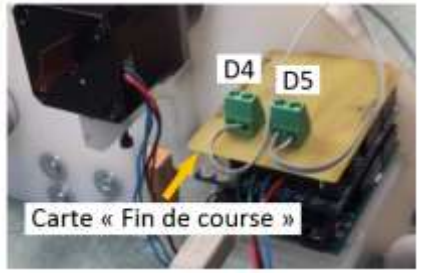
Schéma général du dispositif expérimental

Le dispositif permet d'obtenir les mesures de courbes de dispersion des ondes ultrasonores dans des plaques, des barres, des rails, etc. Il est utilisé pour la validation des méthodes numériques de CND. Le matériel simple et bon marché (carte Arduino, fabrication de chassis dans l'atelier mécanique, etc.) est utilisé.

Boutons « Fin de course »

S.A.B.L. – Système Automatique de Balayage Linéaire

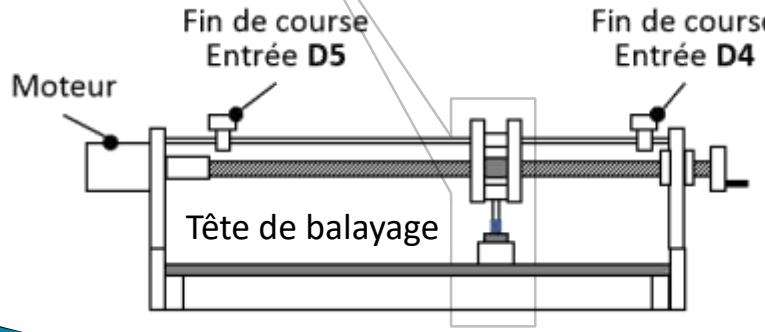


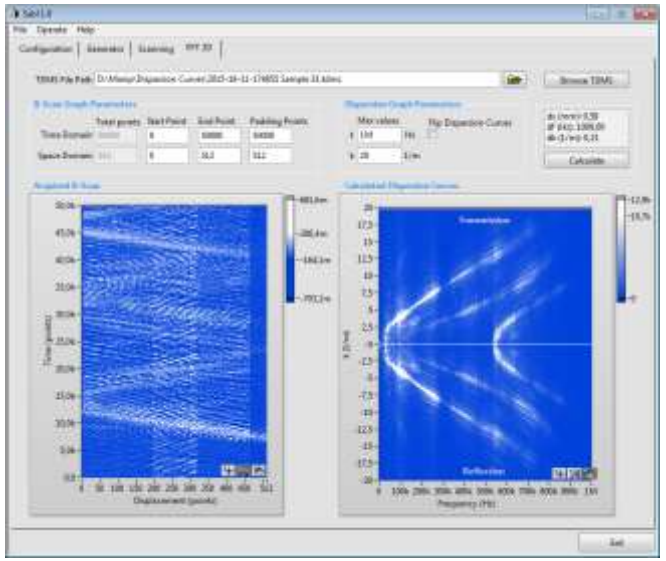


Shield de puissance
Velleman VMA03

Moteur pas-à-pas

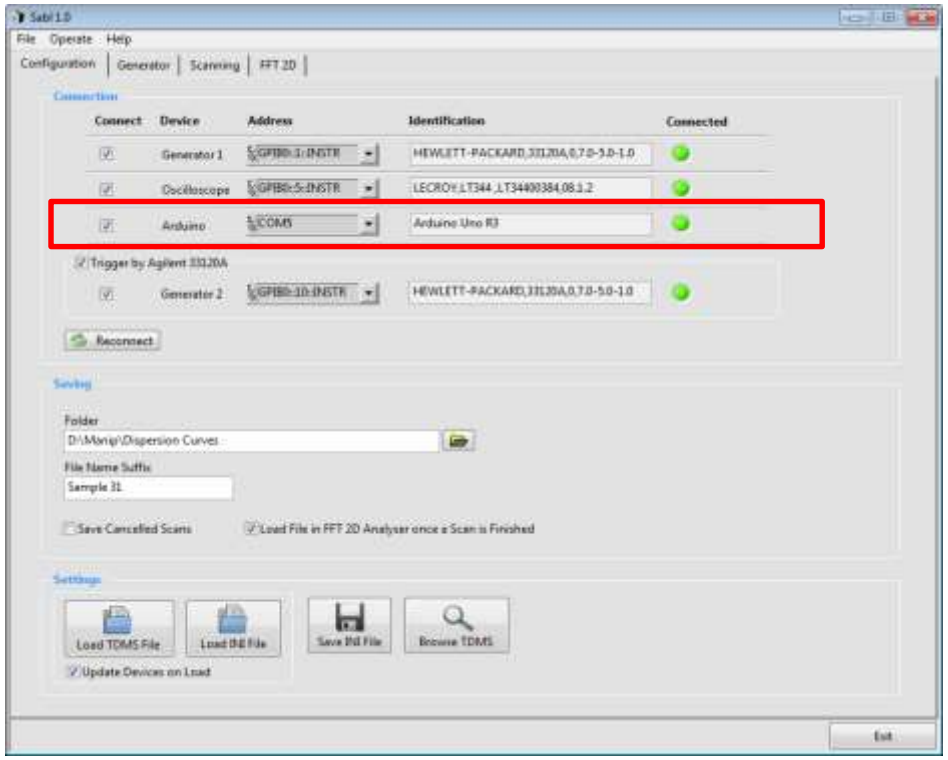
Carte Arduino Uno R3



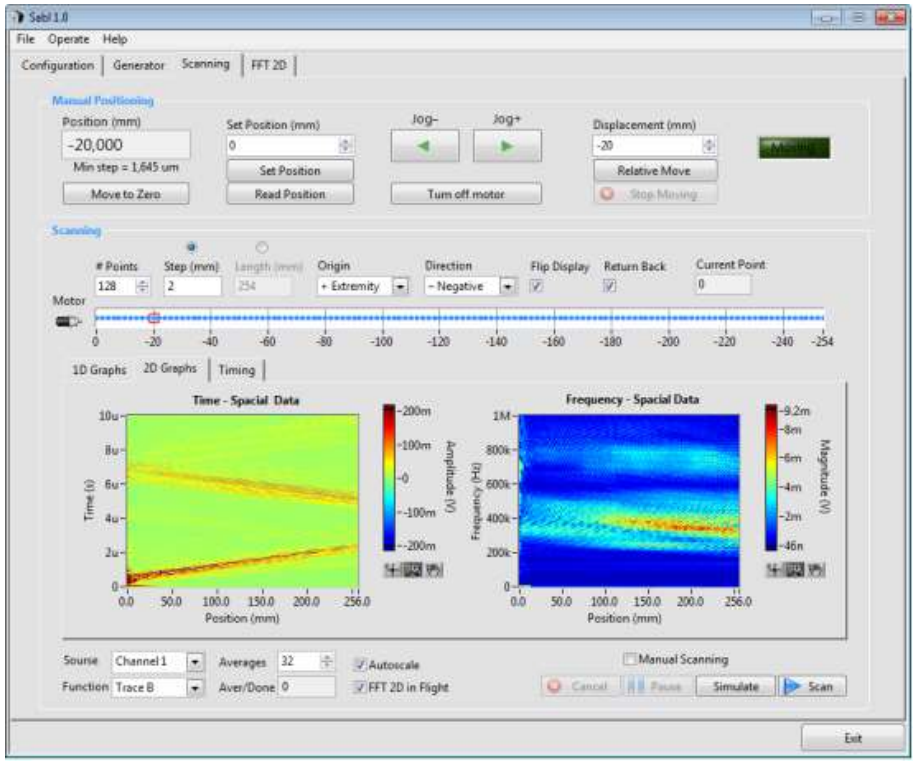


IHM du programme LabVIEW

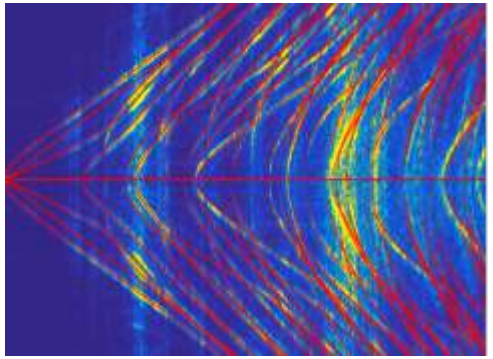
Communication avec Arduino

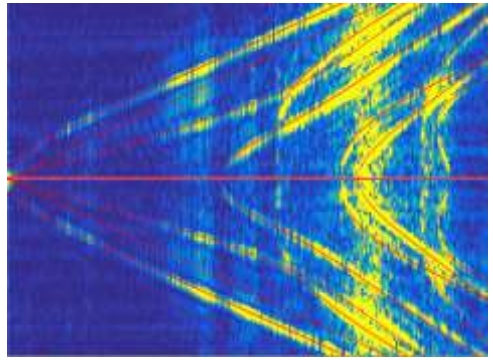


Courbes de dispersion des ondes de Lamb



Superposition des courbes sous Matlab



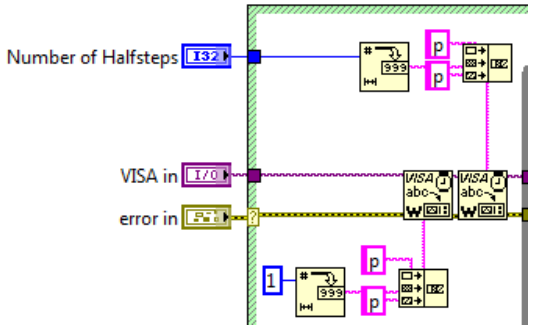


Principe de réalisation de l'interfaçage

LabVIEW

Gestion haut-niveau

- Calcul des déplacements
- Conversion « pas – coordonnées »
- Récupération de la position actuelle



Système de commandes communes

« 1 »

« -1200 »

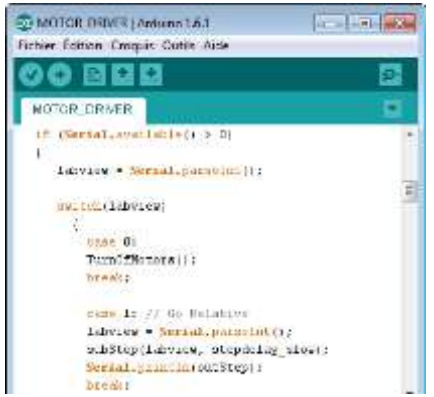
Se déplacer de -1200 pas…

```
0 – Couper l'alimentation
1 – Se déplacer en relative
2 – Se déplacer en continu (+)
3 – Se déplacer en continu (-)
4 – Arrêt d'urgence
5 – Lire la position actuelle
7 – Message d'identité
8 – Attribuer la position
buttonState4 - fins de course (+)
buttonState5 - fins de course (-)
Input long labview
Output long outStep
```

ARDUINO

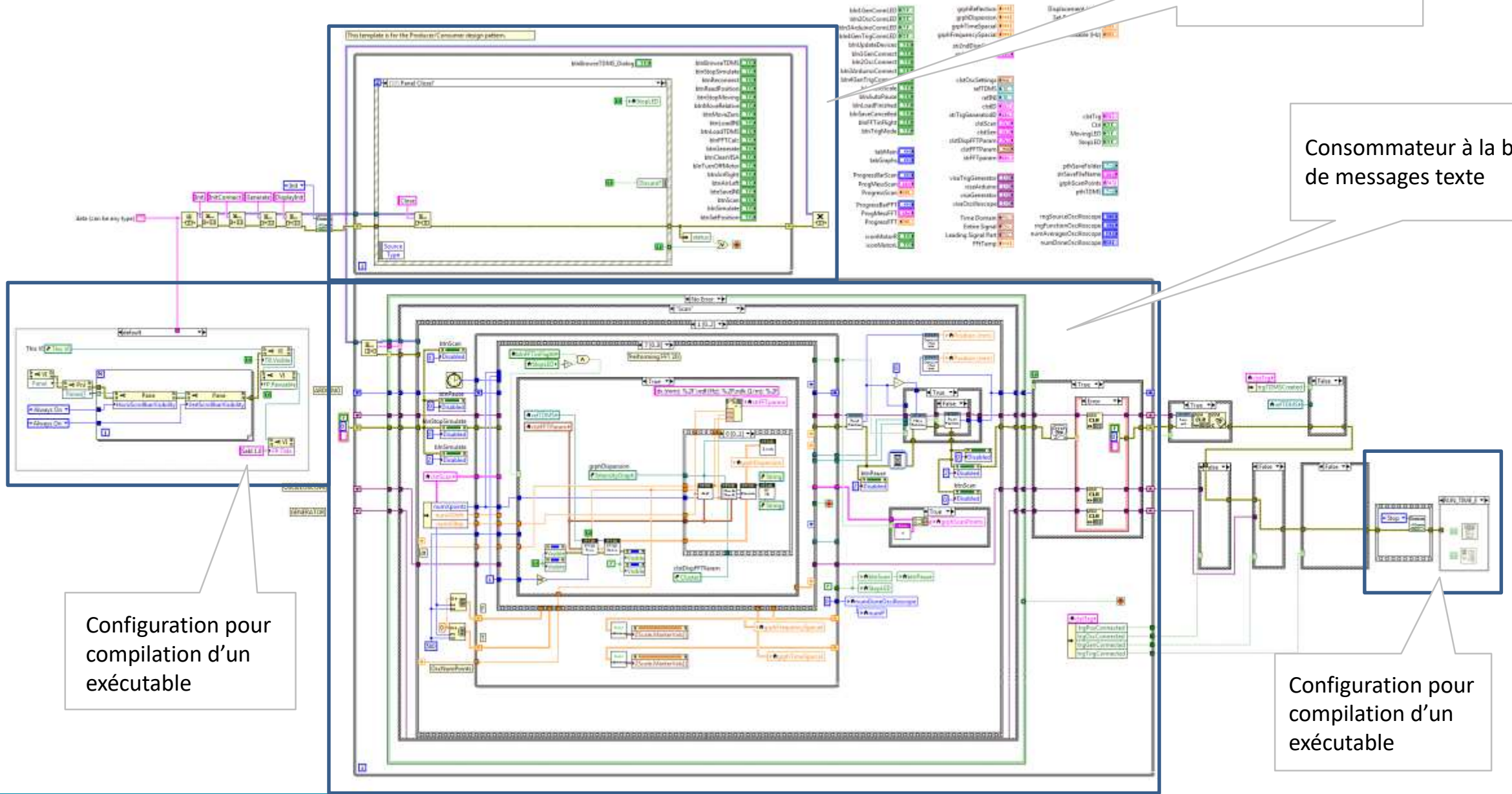
Gestion bas-niveau

- Exécution des pas
- Traitement des boutons « fin de course »
- Calcul de pas effectués



Architecture du programme LabVIEW

Producer/consumer design pattern with events



Producteur avec des évènements

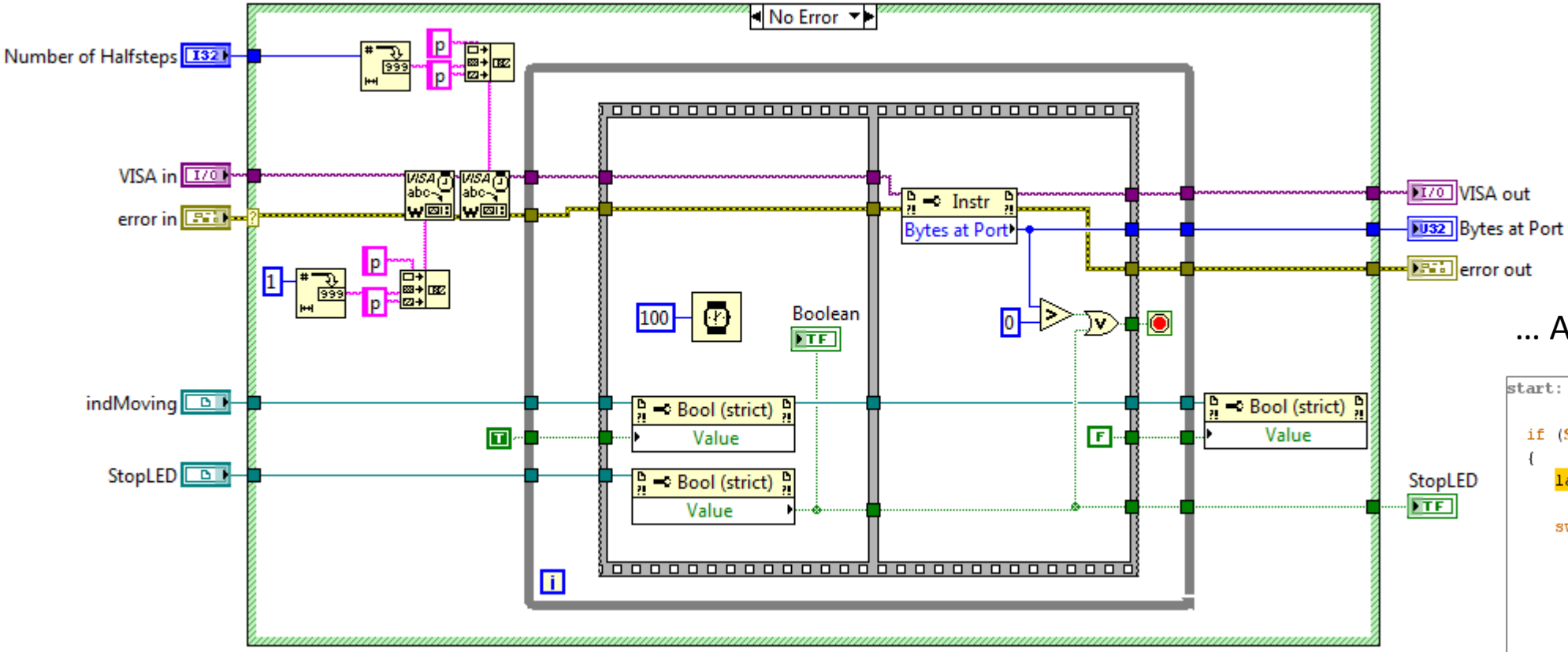
Consommateur à la base de messages texte

Configuration pour compilation d'un exécutable

Configuration pour compilation d'un exécutable

VI « Se déplacer en relatif »

LabVIEW parle…



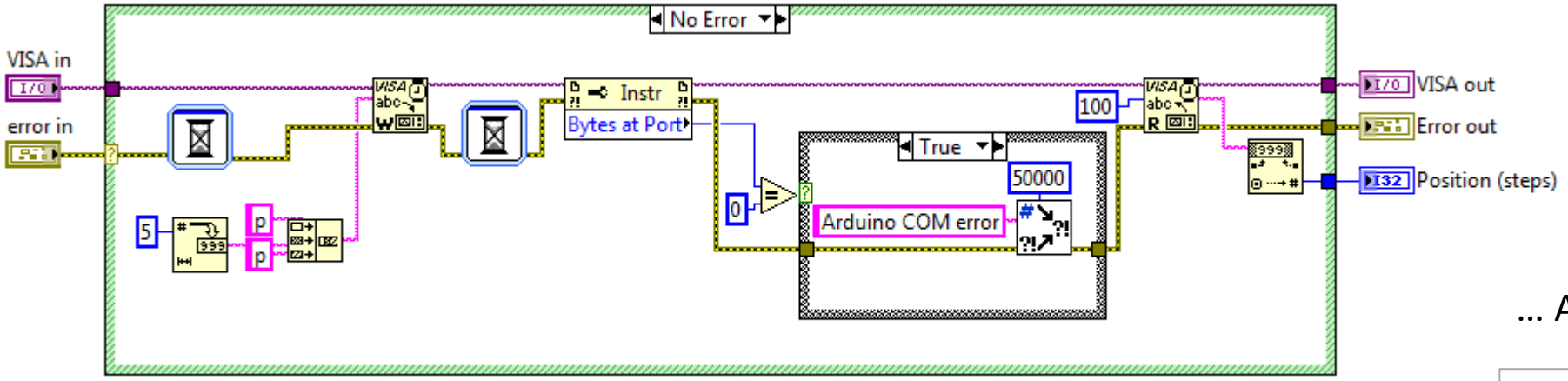
… Arduino écoute

```
start:

 if (Serial.available() > 0)
 {
    labview = Serial.parseInt();

    switch(labview)
      {
        case 0:
        TurnOfMotors();
        break;

        case 1: // Go Relative
        labview = Serial.parseInt();
        subStep(labview, stepdelay_slow);
        Serial.println(outStep);
        break;
```

`Serial.parseInt()`

VI « Donner sa position actuelle »

LabVIEW demande…



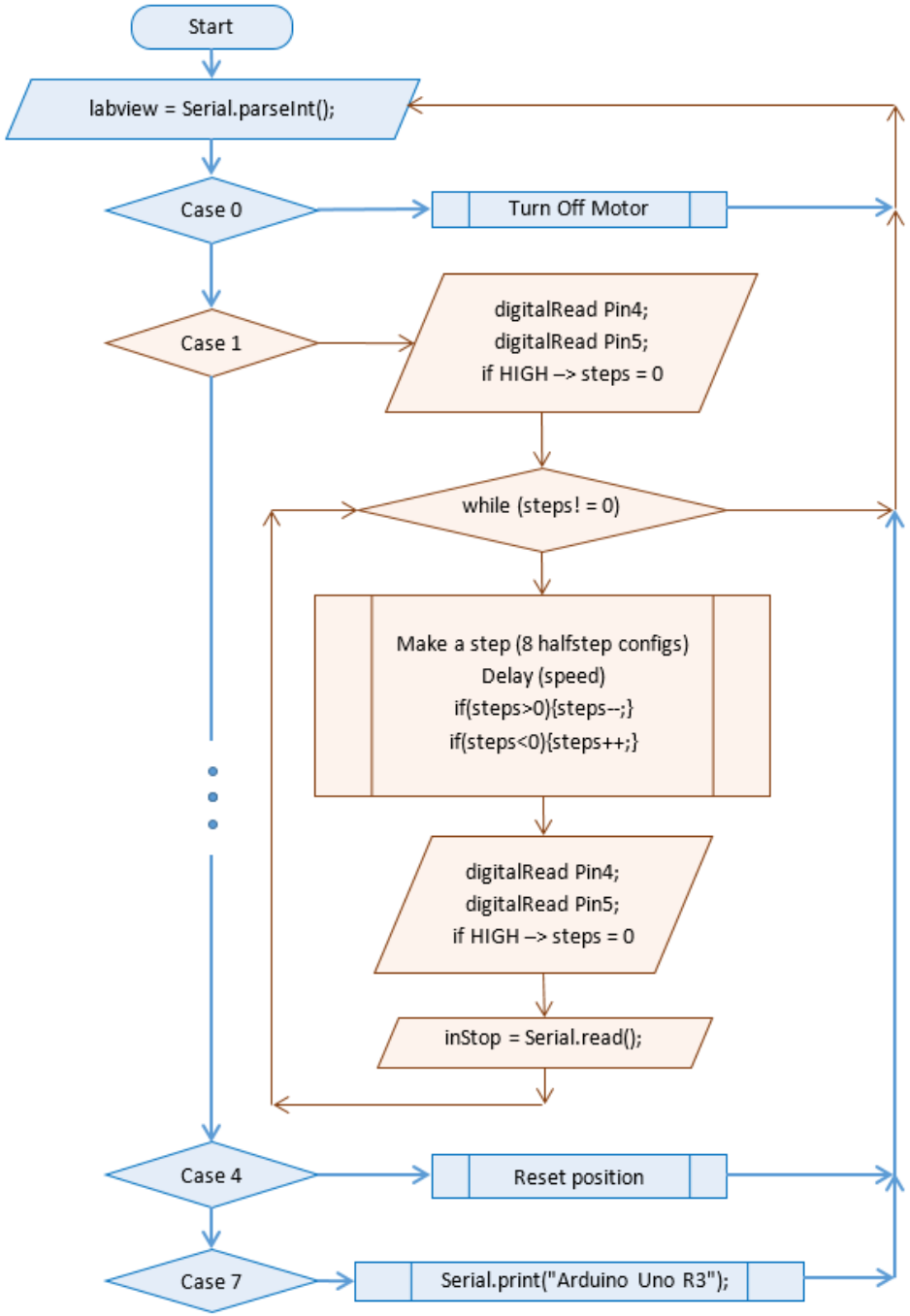
… Arduino répond

```
  case 4: // Stop
  outStep = 0;
  Serial.println(outStep);
  break;

  case 5: // Read Actual Position
  Serial.println(outStep);
  break;

  case 7: // Identification
  Serial.print("Arduino Uno R3");
  break;

  case 8: // Set Position
  outStep = Serial.parseInt();
  break;
}
```

Diagramme de bloques du programme Arduino

Protocole de communication

```
0 – Couper l'alimentation de moteur
1 – Se déplacer en relative
2 – Se déplacer en continu (+)
3 – Se déplacer en continu (-)
4 – Arrêt
5 – Lire la position actuelle
7 – Message d'identité
8 – Attribuer la position

buttonState4 – fins de course (+)
buttonState5 – fins de course (-)
Input long labview
Output long outStep
```



Code Arduino

Fins de course

```
buttonState4 = digitalRead(buttonPin4);
buttonState5 = digitalRead(buttonPin5);

if (buttonState4 == HIGH)
  {
    if (steps < 0)
    {
    steps = 0;
    inStop = 1;
    digitalWrite(ledPin, HIGH);
    }
  }


if (buttonState5 == HIGH)
  {
    if (steps > 0)
    {
    steps = 0;
    inStop = 1;
    digitalWrite(ledPin, HIGH);
    }
  }

if (Serial.available()

{
  inStop = Serial.rea
```

Bobines

```
switch(sub)
 {
        case 0:
// Starting position (if repeated, ful step (4))
// EXPLANATION: in this case, both our power are high.
// Therefore both coils are activated, with their standard polarities for their magnetic fields.
digitalWrite(pwr_a,HIGH);
digitalWrite(pwr_b,HIGH);
digitalWrite(dir_a,HIGH);
digitalWrite(dir_b,HIGH);
        break;

        case 1:
//Half step (2)
// EXPLINATION: In this case, only out b-coil is active, still with it's stand polarity.
digitalWrite(pwr_a,HIGH);
digitalWrite(pwr_b,LOW);
digitalWrite(dir_a,HIGH);
digitalWrite(dir_b,LOW);
        break;
case 2:
        //Full step (1)
// EXPLANATION: In this case, the b-coil is activated as in previous cases.
// But the a-coil now has it's direction turned on. So now it's active, but with the reversered polarity.
// By continuing this pattern (for reference: http://www.8051projects.net/stepper-motor-interfacing/full-step.gif) , you'll
digitalWrite(pwr_a,HIGH);
digitalWrite(pwr_b,HIGH);
digitalWrite(dir_a,HIGH);
```

Système de commandes

Gestion de bobines de moteur

Merci pour votre attention !

iemn
Institut d'Electronique, de Microélectronique et de Nanotechnologie
UMR CNRS 8520
Département OAE

Comparaison de deux approches

	Communication Firmware LINX	Liaison série directe
Pros	• Firmware LINX gratuit • conçu par des ingénieurs de NI • Rapidité d'utilisation avec sa propre palette de fonction et un firmware discret • Pas besoin de connaitre le langage Arduino	• Possibilité d'intégrer des codes Adruino issus de la communauté • Large champs de possibilités pour des personnalisations • Gestion des évènements en cours d'exécution d'autres taches • Possibilité de fonctionner avec des autres contrôleurs logiciels (Matlab, Python, C++…) sans changer de code coté Arduino
Cons	• Pas une application temps réel • Bonne maitrise du langage Arduino pour les fonctions personnalisées	• Programmation de deux cotés • Elaboration de système de commandes/messages • Maitrise de langage Arduino (modérée) est requise • Gestion de temps de latence et des conflits d'écriture/lecture

Conclusion

Pros

Arduino – cible à temps réel (gestion du matériel bas niveau par des interruptions)

Simple et sans intermédiaires

Pilotage moteurs (séquences des pas adaptées au types de moteurs)

Possibilité d'introduire de rampes d'accélération

Gestion des évènements en cours d'exécution d'autres taches

Cons

Nécessité de programmation de deux cotés (LabVIEW - Arduino)

Nécessité de réfléchir aux système de messages

(un simple « long » ne suffira pour les applications plus avancées)