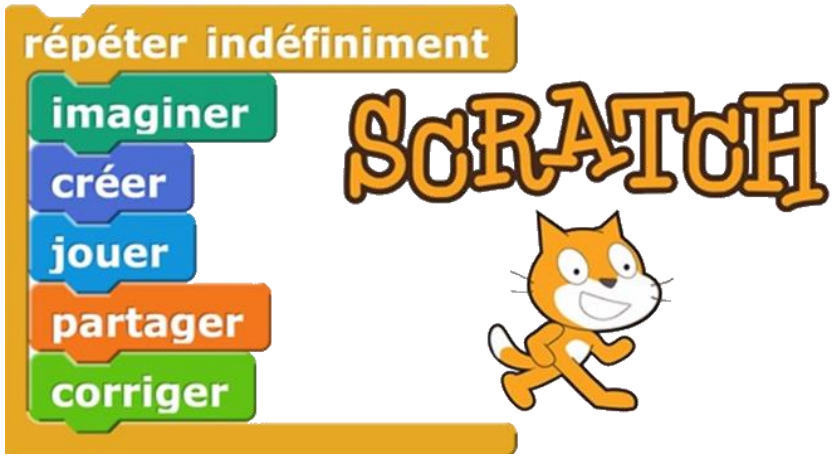
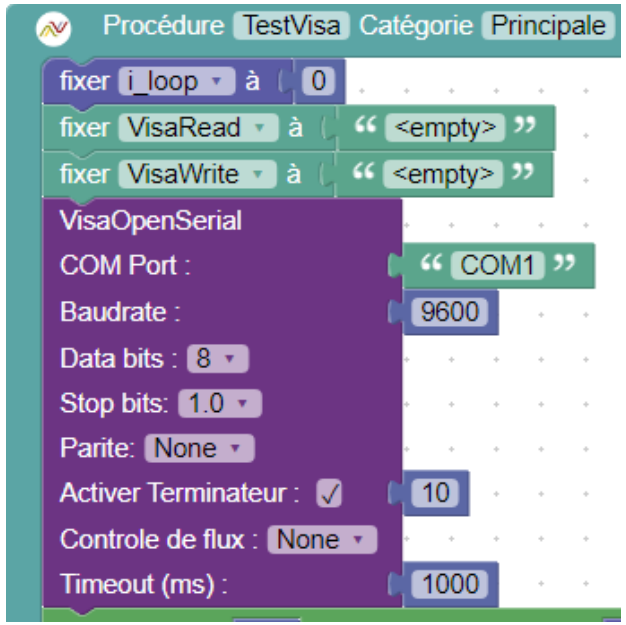
- Blockly -

Automatiser des actions sous LabVIEW avec l'interface de Scratch





Le présentateur

Nerys Mesulog
Nerys Group companies

- Amaury LAURENT
- Développeur LabVIEW depuis 2014
- Certifications : CLED et CLA
- Responsable projets systèmes embarqués
- Aéromodéliste

NERYS en bref

Nerys Mesulog
Nerys Group companies

- Fondée en 2007 à Gardanne (Bouches du Rhône), une équipe de 15 personnes ;
- Concepteur de bancs de tests, de caractérisation et de systèmes de mesure :
  - Collaborateurs spécialisés en électrotechnique, électronique et mécanique ;
  - Pôle développement logiciels, avec une expertise LabVIEW.
- NERYS propose une solution personnalisable pilotage de moyens d'essais appelée "Vasco" pouvant être déployée sur cible temps-réel (NI RT).

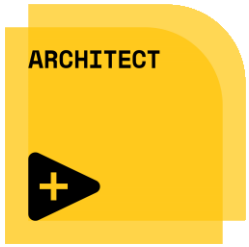
vasco by Nerys

NERYS GROUP en bref

Nerys Mesulog
Nerys Group companies

- Depuis Mai 2022 : NERYS + MESULOG → NERYS GROUP
- Compétences logicielles NI :
  - LabVIEW (Windows, RT, DSC, FPGA)
  - TestStand
  - VeriStand
- Partenaire National Instruments
- Développeurs certifiés LabVIEW et TestStand

Silver Partner



Plus d’infos : nous contacter

Nerys Mesulog
Nerys Group companies

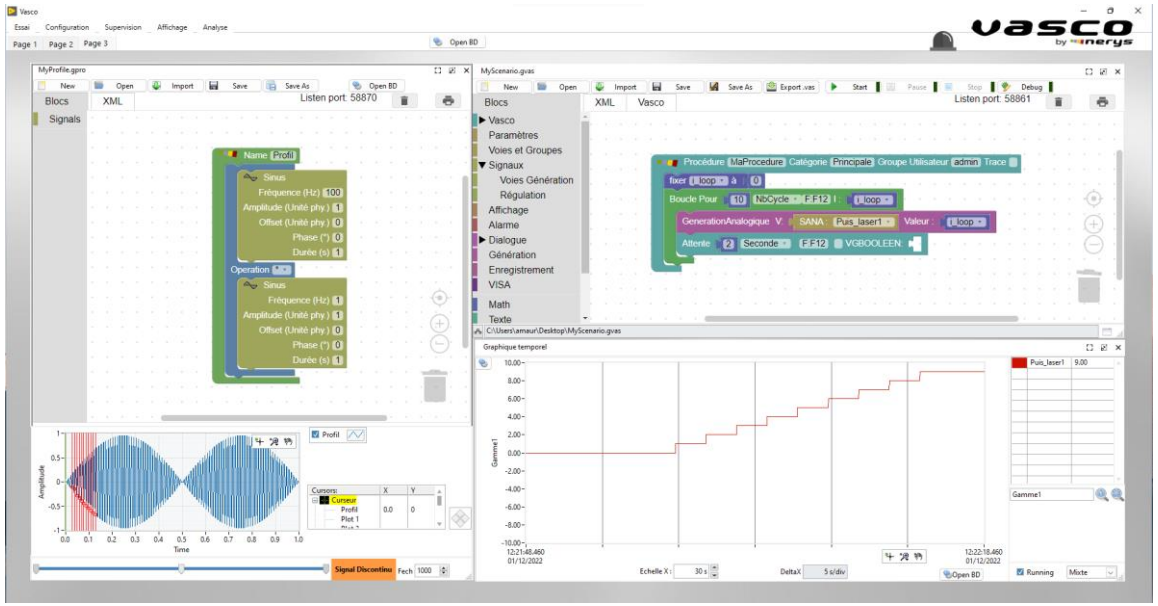
- Nos sites :
  - Gardanne (13)
  - Moirans (38)
  - La Rochette (73)
- Sur le web :
  - https://nerysgroup.com/fr/
  - https://www.mesulog.fr/
- Nous contacter :
  - contact@nerysgroup.com
  - contact@mesulog.fr

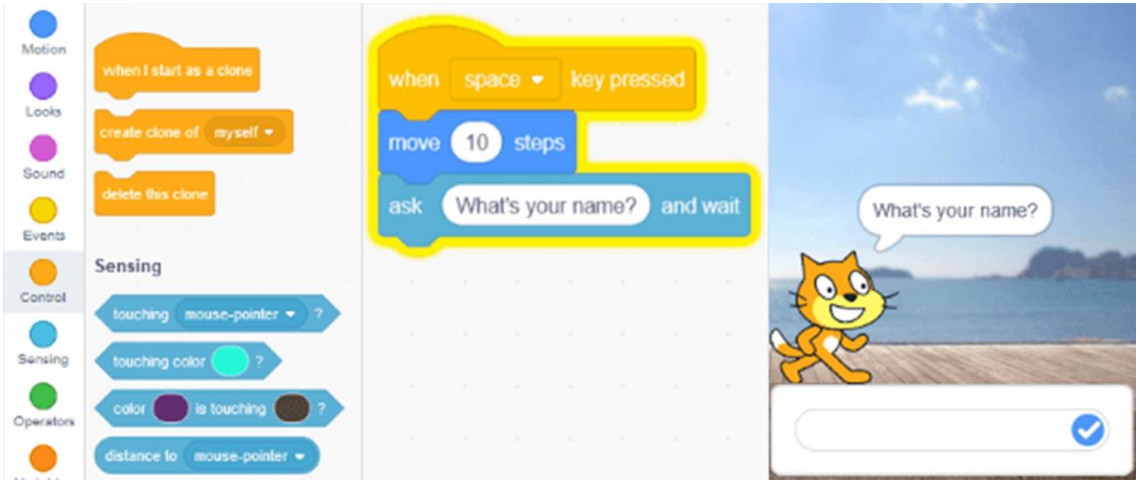
Google Blockly

Nerys Mesulog
Nerys Group companies

- Projet présenté en 2012
- Open Source
- Bibliothèques JavaScript
- But : faire des interfaces de programmation graphique
- Entièrement personnalisable

- A notamment servi pour:
  - AppInventor (Android)
  - Scratch (MIT)
  - Vasco (NERYS)





Principe d'utilisation de Blockly

Nerys Mesulog
Nerys Group companies

Editeur graphique

HTML 5

Multiplateformes

Ecriture du code

Descripteurs block

Javascript

Définitions des blocks

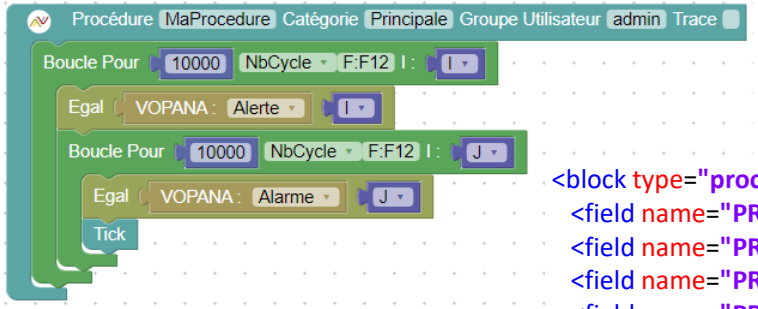
Générateurs code

Javascript

Convertisseur block -> code

Code source

```
Debut ( MaProcedure , Principale , admin , )
  Pour ( 10000 , NbCycle , F:F12 , , null , , I )
    Egal ( O:Alerte , N:I )
    Pour ( 10000 , NbCycle , F:F12 , , null , , J )
      Egal ( O:Alarme , N:J )
      Tick ( )
    FinPour
  FinPour
Fin
```



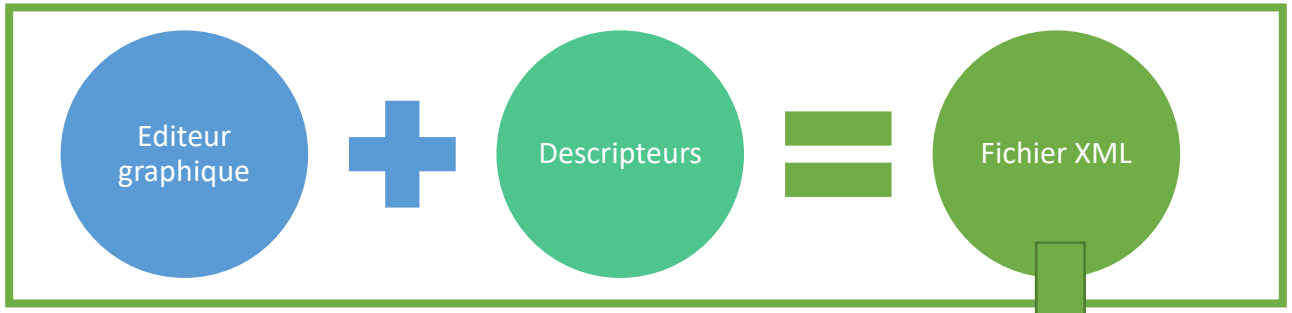

```
<block type="procedure>
  <field name="PROC_NAME">MaProcedure</field>
  <field name="PROC_CAT">Principale</field>
  <field name="PROC_GROUP">admin</field>
  <field name="PROC_TRACE">FALSE</field>
  <statement name="PROC">
    <block type="bky_for_loop" id="Et:[fwd6Ato!|9c8x:%M">
      <field name="UNIT">NbCycle</field>
```

Exemple de descripteur XML

Intégration avec LabVIEW

Nerys Mesulog
Nerys Group companies

Principe : Utilisation du fichier XML comme source interprétée par LabVIEW



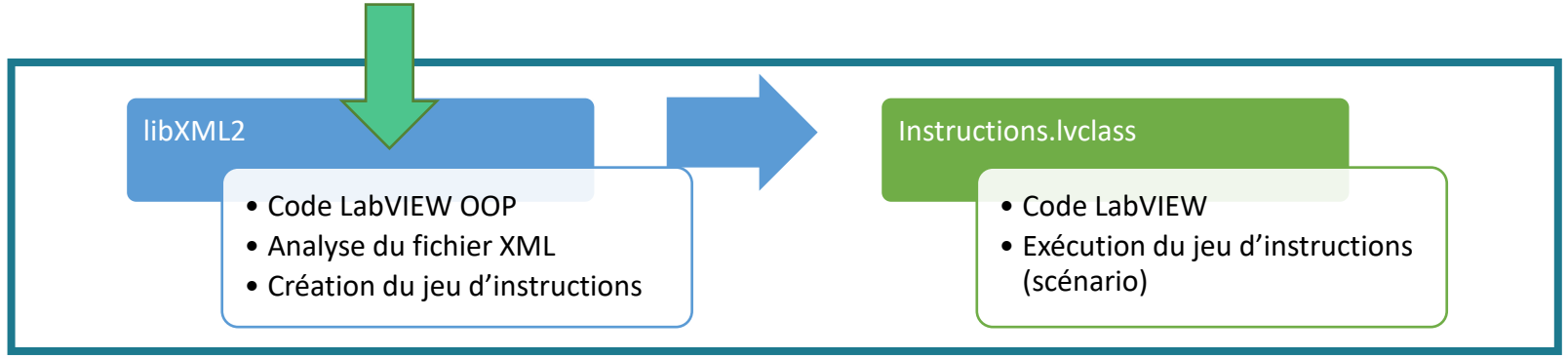
Navigateur Web

Structure XML :

- Adaptée à une représentation objet
- Récursive
- Simple à lire

```
<block type="procedure" id="pz*o73e6+Am^zYpJvpZ2" x="38" y="38">
  <field name="PROC_NAME">MaProcedure</field>
  <field name="PROC_CAT">Principale</field>
  <field name="PROC_GROUP">admin</field>
  <field name="PROC_TRACE">FALSE</field>
  <statement name="PROC">
    <block type="bky_for_loop" id="Et:[fwd6Ato!|9c8x:%M">
      <field name="UNIT">NbCycle</field>
```

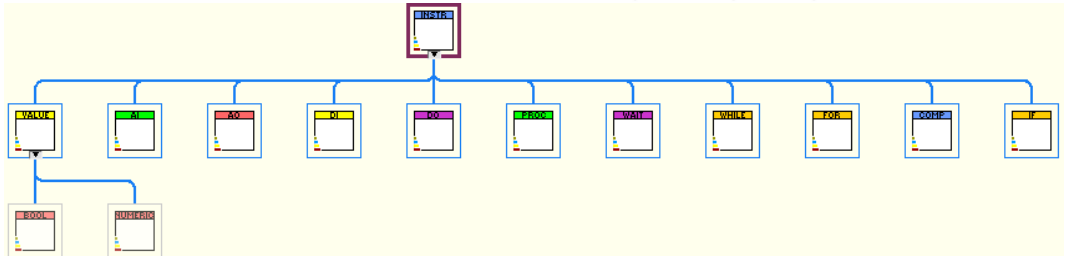
LabVIEW



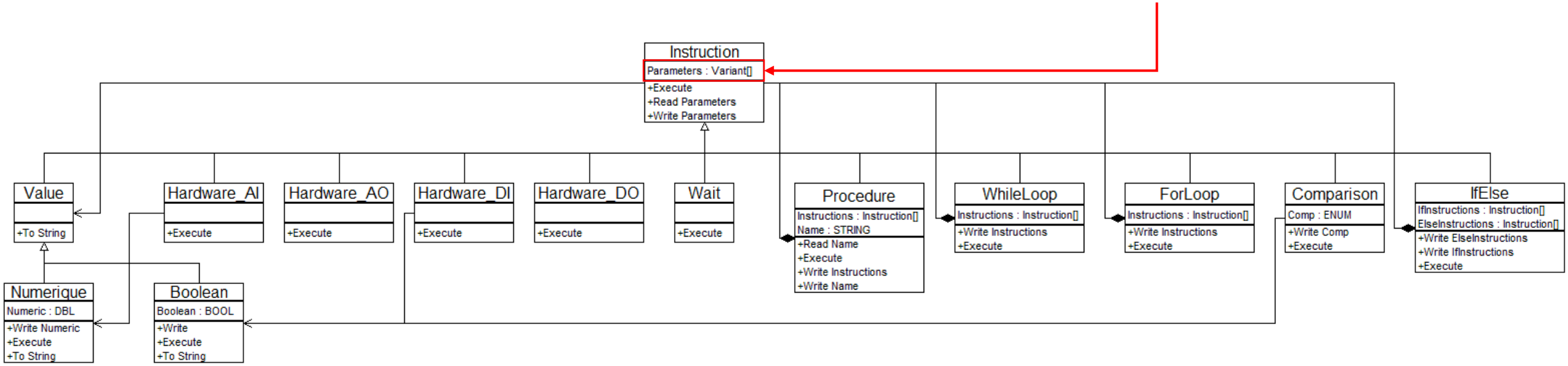
LabVIEW : Gestion des instructions

Nerys Mesulog
Nerys Group companies

- Architecture Objets
  - Dispatch dynamique
- Exécution récursive
  - Une seule primitive : Execute.vi



NB : Parameters : Instruction[]
Astuce pour permettre à un objet de contenir des instances de lui-même : le variant



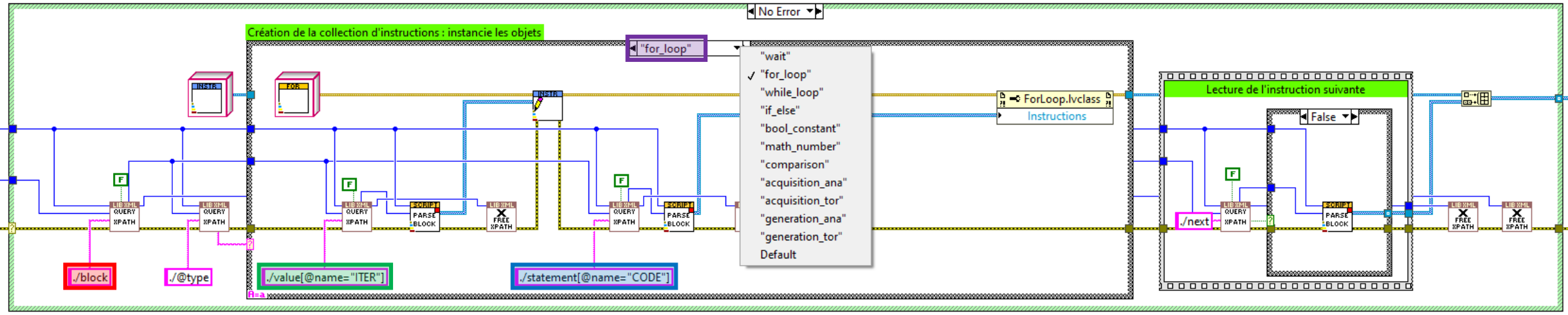
LabVIEW : Analyse du XML

Nerys Group companies

LibXML2 sous LabVIEW

- Utilisation des XPath
- Lecture récursive

```
<block type="for_loop" id=":|8-T_rWn@-JjY573Tqk">
    <value name="ITER">
        <block type="math_number" id="/`AzaFG2xxN-L|1py^%6">
            <field name="NUM">10</field>
        </block>
    </value>
    <statement name="CODE">
        <block type="wait" id="I?+4ik@.Ms$GVpXnblq9">
            <value name="DURATION">
                <block type="math_number" id="@#kFEYc))7%jYcOvRwQ7">
                    <field name="NUM">5</field>
```



Lien vers le toolkit LabXML (SourceForge)

Exécution récursive : Execute.vi

Nerys Mesulog
Nerys Group companies

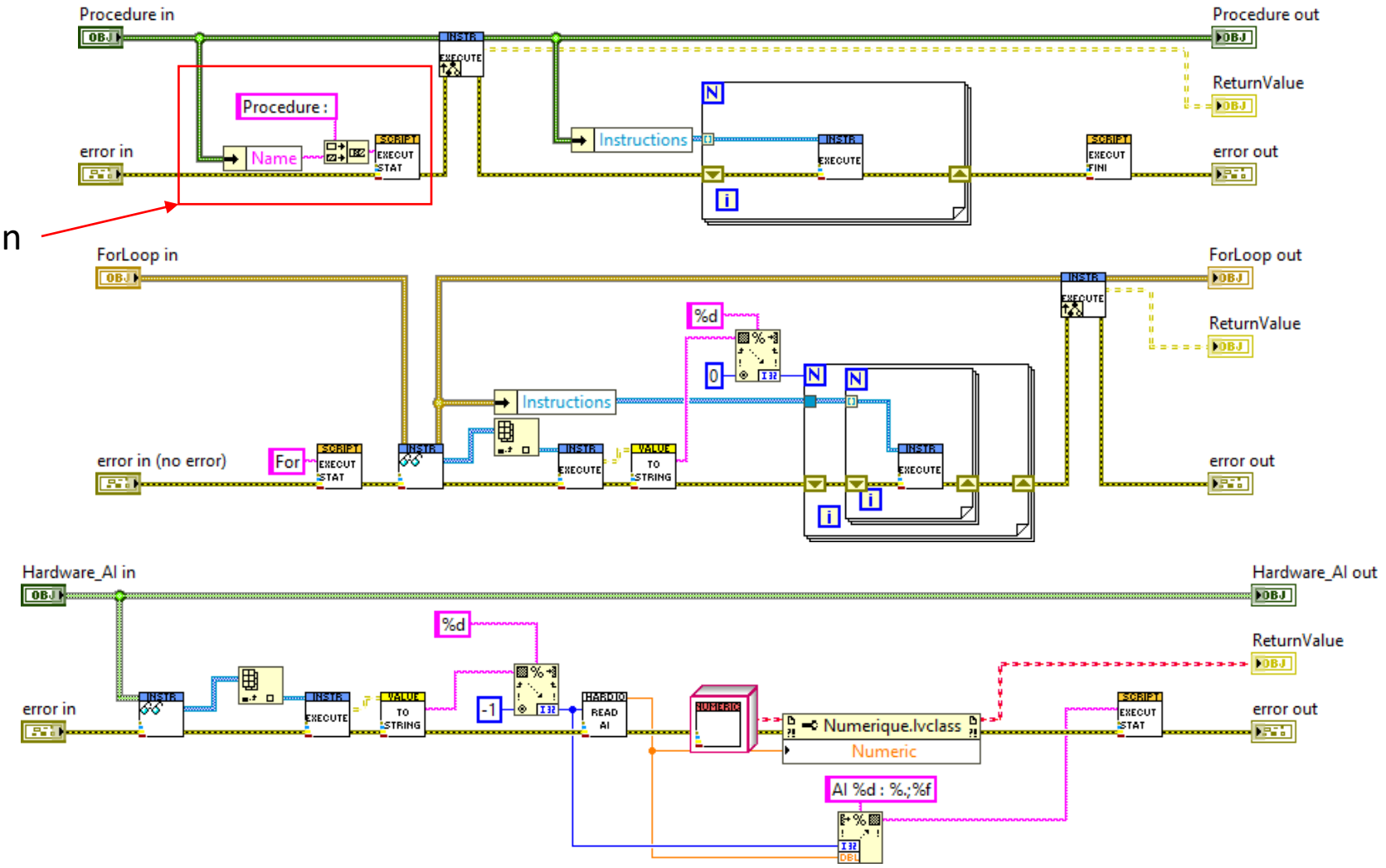
Dispatch dynamic :

Procedure.lvclass:Execute.vi

Trace d'exécution

ForLoop.lvclass:Execute.vi

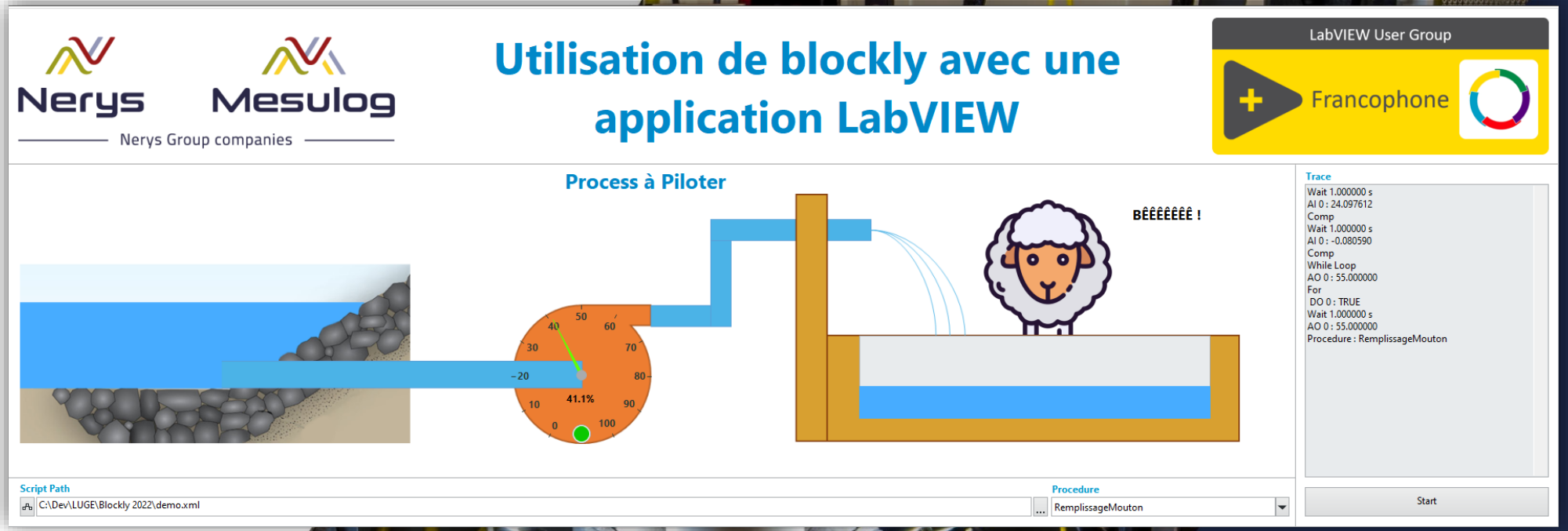
Hardware_AI.lvclass:Execute.vi



Pour aller plus loin

Nerys Mesulog
Nerys Group companies

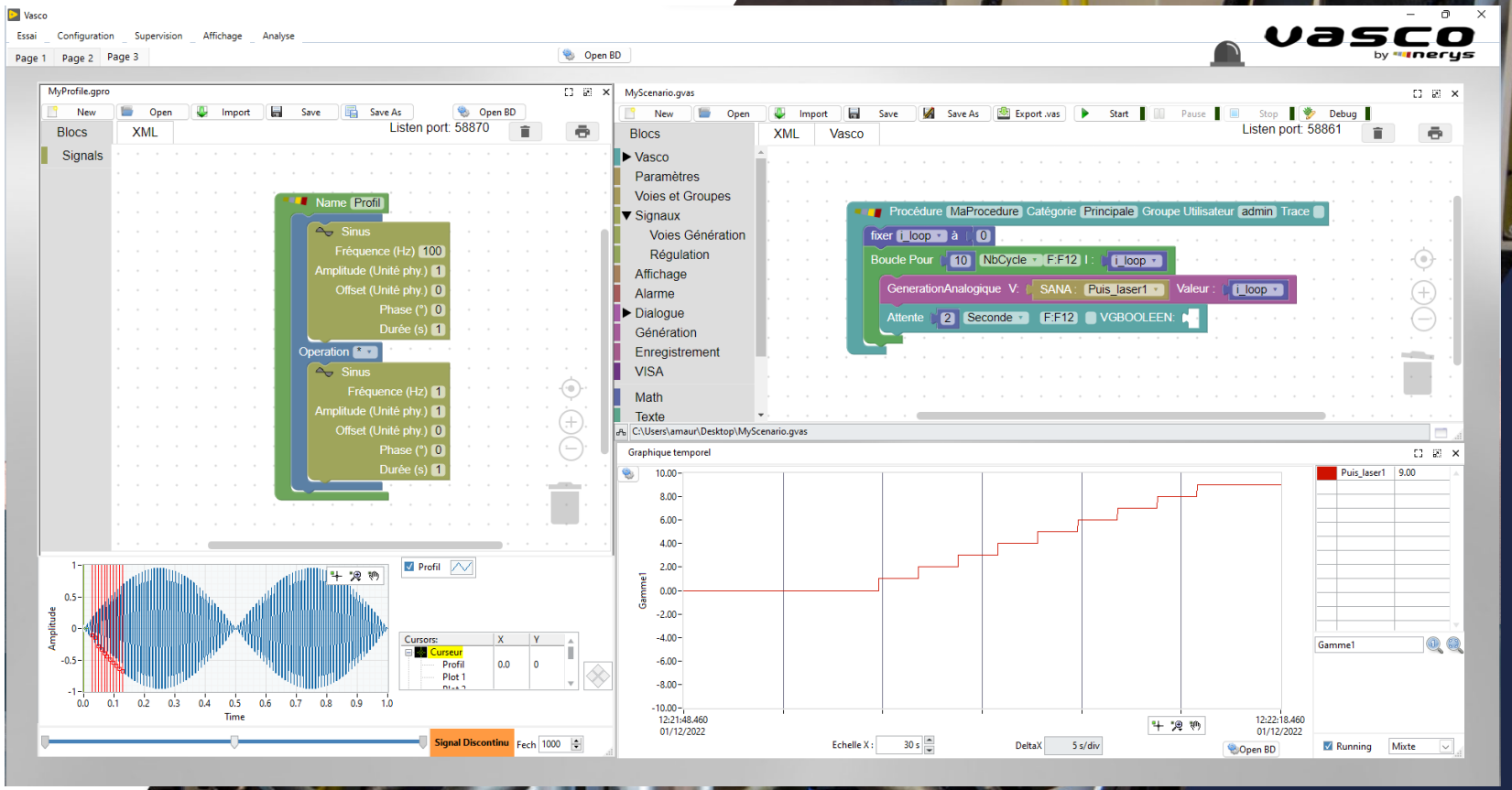
- Communication entre l'éditeur Blockly et LabVIEW
  - Animation d'exécution
  - Chargement automatique du fichier XML dans blockly
- Gestion d'exécution :
  - Debugger
  - Breakpoints
  - Variables



Démonstration

#DQMH #Blockly #LabVIEW #Mouton

www.nerysgroup.com



Démonstration de Blockly dans un logiciel Industriel

#Vasco #NERYS #PlacementDeProduit

www.nerysgroup.com

Nerys

Mesulog

Nerys Group companies

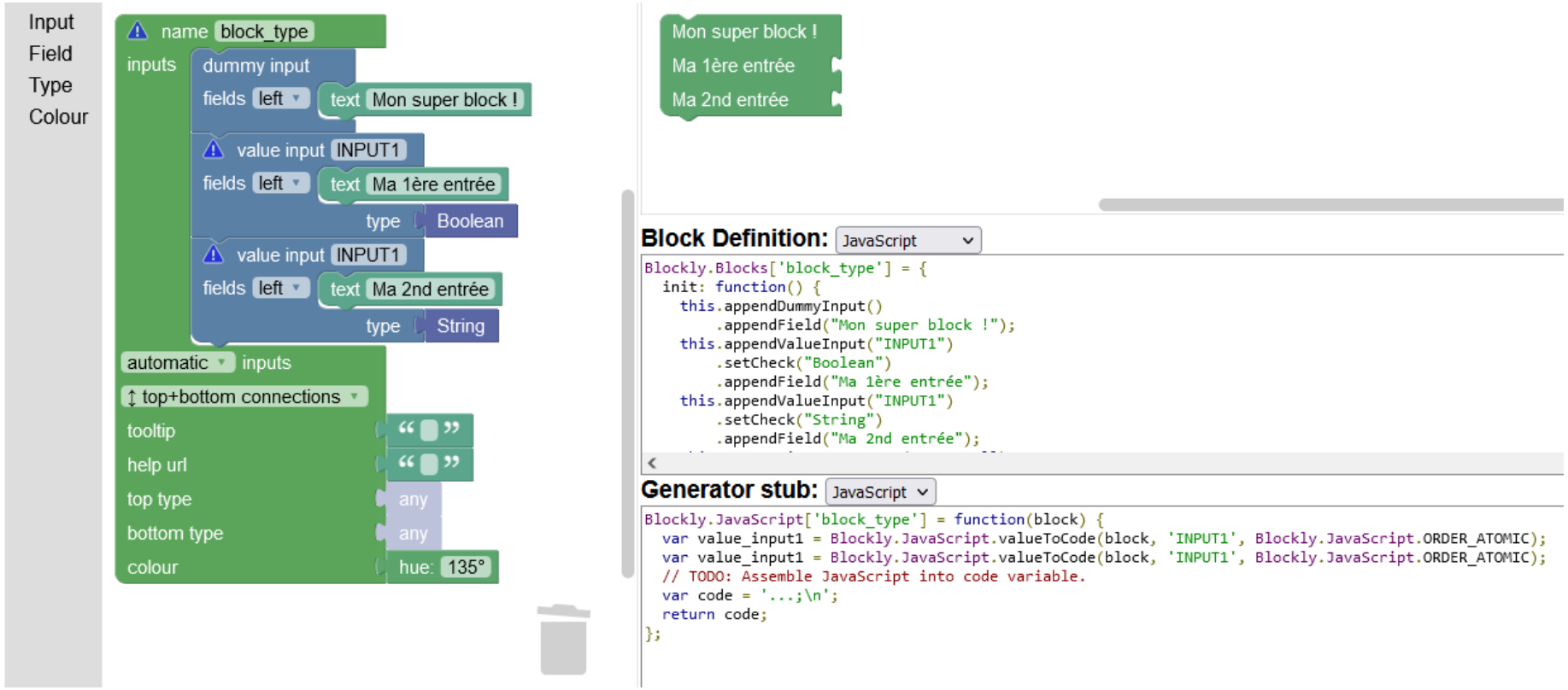
Merci à tous pour votre attention

www.nerysgroup.com

Annexe : Blockly Factory

Nerys Mesulog
Nerys Group companies

https://blockly-demo.appspot.com/static/demos/blockfactory/index.html



Annexe : descripteur de block

Nerys Group companies

.\Blockly\Specifique\addons-blockly\Blocks\LUGE.js

```
Blockly.Blocks['for_loop'] = {
 init: function() {
                this.loop_index = new Blockly.FieldTextInput("");
   this.appendDummyInput()
      .appendField("Pour");
   this.appendValueInput("ITER");
   this.appendDummyInput()
      .appendField("Iterations");
   this.appendStatementInput("CODE")
      .setCheck(null);
   this.setInputsInline(true);
   this.setPreviousStatement(true, null);
   this.setNextStatement(true, null);
   this.setColour("%{BKY_LOOPS_HUE}");
 this.setTooltip("");
 this.setHelpUrl("");
 this.loop_index.showEditor_=(()=>{

                });
  }
};
```

Annexe : générateur de code

Nerys Group companies

.\Blockly\Specifique\addons-blockly\Generators\Luge\LUGE.js

```
Blockly.Luge['for_loop'] = function(block) {

  var value_iter = Blockly.Luge.valueToCode(block, 'ITER', Blockly.Luge.ORDER_ATOMIC);
  var statements_code = Blockly.Luge.statementToCode(block, 'CODE');

  var code = 'Pour ( ' + value_iter + ' )\n'+
  statements_code +
  'FinPour\n';
  return code;
};

Blockly.Luge['generation_ana'] = function(block) {
  var value_channel = Blockly.Luge.valueToCode(block, 'CHANNEL',
Blockly.Luge.ORDER_ATOMIC);
  var value_value = Blockly.Luge.valueToCode(block, 'VALUE', Blockly.Luge.ORDER_ATOMIC);
  var code = 'GenerationAnalogique ( ' + value_channel + ' , ' + value_value + ' )\n';
  return code;
};
```


```
Pour ( 5 )
  GenerationAnalogique ( 1 , 2 )
FinPour
```