



SASE 2016

Viernes 12 de agosto de 2016 FI-UBA - CABA.



Programando la CIAA en LADDER y agregado de bloques de programa personalizados en lenguaje C

Módulo 2 - Herramientas de desarrollo de IDE4PLC

Ing. Eric Nicolás Pernia (UNQ)

ericpernia@gmail.com

Dr. Lic. Carlos Lombardi (UNQ)

carlombardi@gmail.com



Universidad
Nacional
de Quilmes



Computadora Industrial
Abierta Argentina
Desarrollo colectivo



Temario

Módulo 2 - Herramientas de desarrollo de IDE4PLC

- IDE4PLC para desarrolladores.
- Herramientas utilizadas para el desarrollo de IDE4PLC.
- Pharo y Smalltalk.
- Ejemplo "Simulink".
- Inspector y System Browser.

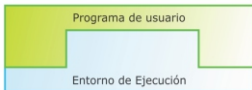
IDE4PLC para desarrolladores



Entorno de
Programación
de PLC

genera el

Software de PLC
(firmware)



Archivos .C y .H
que implementan el
Programa de usuario.
Son generados por el
Editor de Programas
de PLC
Archivos .C y .H
fijos de funcionalidad
básica que implementan
el Entorno de Ejecución

se compila
y descarga al



Equipo
electrónico
PLC

Hardware programable desde IDE4PLC



PLeriC

PLC con microcontrolador LPC1769.

8DI 24VDC - 4DO 24VDC - 4DO a Relé.

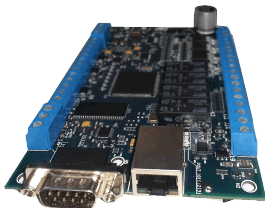


EDU-CIAA-NXP

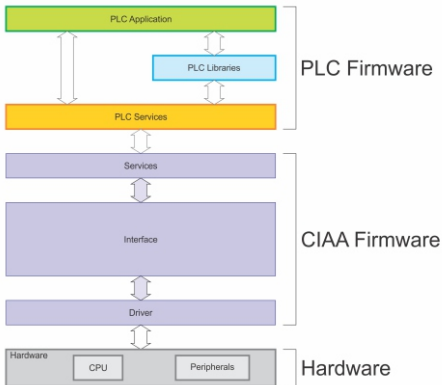
Con microcontrolador LPC4337
doble núcleo (Cortex M0 y M4).

CIAA-NXP

Sin soporte oficial por el momento pero
funciona con ciertas consideraciones.
Con microcontrolador LPC4337.



Firmware en lenguaje C

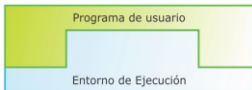




Entorno de
Programación
de PLC

genera el

Software de PLC
(firmware)



Archivos .C y .H
que implementan el
Programa de usuario.
Son generados por el
Editor de Programas
de PLC
Archivos .C y .H
fijos de funcionalidad
básica que implementan
el Entorno de Ejecución

se compila
y descarga al



Equipo
electrónico
PLC

Herramientas utilizadas para el desarrollo de IDE4PLC


El entorno de programación está desarrollado sobre Pharo-Smalltalk.

El firmware del PLC, está escrito íntegramente en lenguaje C, utilizando el IDE Eclipse y se monta de manera natural sobre el Firmware de la CIAA.



Editor de lenguajes PLC en Pharo

Pharo (D:\Proyecto\Pharo2.0\Pharo2.0.image)



PLC_Boolean >> #bitSize

- PLerC-ConnectableBlocks
- PLerC-DataTypes
- PLerC-DataTypes-Element
- PLerC-DataTypes-Generic
- PLerC-Declarations
- PLerC-Declarations-POU
- PLerC-EjemplosMorphic
- PLerC-Elements-GraphicLa
- PLerC-Elements-IL
- PLerC-GUI
- PLerC-GraphicElementMor
- PLerC-GraphicElementMor

PLC_Boolean >> #bitSize

- PLC_Boolean
- PLC_Byte
- PLC_DoubleWord
- PLC_LongWord
- PLC_Word
- PLC_DateAndTime
- PLC_Date
- PLC_DateAndTimeOfDay
- PLC_TimeOfDay
- PLC_Reals
- PLC_LongReal
- PLC_Real
- PLC_SignedIntegers

acceptValue
acceptWire
bitSize
description
initialValue
keyword
range

Groups Hierarchy Class side Comments PLC_Boolean >> #bitSize


bitSize

"Devuelve el tamaño del tipo de dato en cantidad de bits."

1.

Workspace

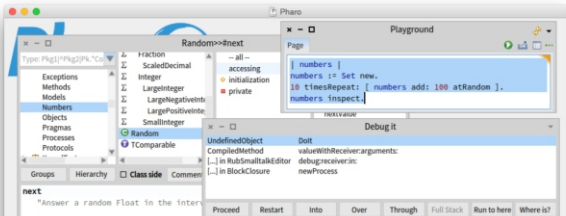
```
1),  
c3 := net addContact: PLC_Element NOContact  
onConnection: [pr connectedAtOutput: 2].  
rpr2 := c3 blockConnectedToOutput: 1.  
~v1 := "net closeBranchFrom: rpr2 to: NL.  
net changeActualArgumentFor:  
  (c1 element  
   connectableBlockActualArgumentFor: c1)  
  by: 'ENT0'.  
net changeActualArgumentFor:  
  (c2 element  
   connectableBlockActualArgumentFor: c2)  
  by: 'ENT1'.  
net changeActualArgumentFor:  
  (c3 element  
   connectableBlockActualArgumentFor: c3)  
  by: 'ENT2'.  
net changeActualArgumentFor:  
  (b1 element  
   connectableBlockActualArgumentFor: b1)  
  by: 'SAL1'.  
"controlador := PLC_LadderController newWithModel:  
net."
```





The immersive programming experience

Pharo is a pure object-oriented programming language *and* a powerful environment, focused on simplicity and immediate feedback (think IDE and OS rolled into one).



Pharo

- Entorno integrado de desarrollo.
- De código abierto.
- Proyecto activo. Pharo 5.0 lanzado en mayo de 2016.
- Desarrollado en **Francia**... con bastante mano **Argentina** de gente que conocemos.
- Entorno gráfico propio, muy particular.
 - Permite máxima portabilidad.

Smalltalk

- Todo se hace mediante objetos.
- Manejo automático de memoria.
 - Sin malloc() ni free(), confiamos en Smalltalk.
- No existen tipos de datos.
- Sintaxis particular.
- Bloques.

Pharo Smalltalk

The complete syntax

exampleWithNumber: x

“A method that illustrates every part of Smalltalk method syntax”

| y |

true & false not & (nil isNil) ifFalse: [self halt].

y := self size + super size.

#\$a #a 'a' 1 1.0)

do: [:each | Transcript

show: (each class name);

show: (each printString);

show: ' '].

^ x < y

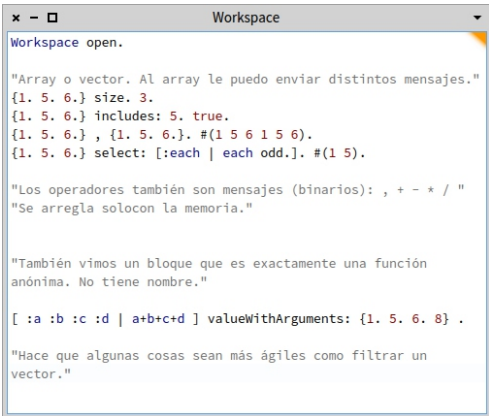
Objeto

Podemos pensarlos como “una Estructura de datos con **mucha** vitamina”.

Idea inicial de la programación con objetos

Cualquier cosa que quiera hacer,
se la tengo que pedir a un **objeto**
enviándole un **mensaje**
que puede tomar **parámetros**

Ejemplo Array.



```
x - □ Workspace
Workspace open.

"Array o vector. Al array le puedo enviar distintos mensajes."
{1. 5. 6.} size. 3.
{1. 5. 6.} includes: 5. true.
{1. 5. 6.} , {1. 5. 6.}. #(1 5 6 1 5 6).
{1. 5. 6.} select: [:each | each odd.]. #(1 5).

"Los operadores también son mensajes (binarios): , + - * / "
"Se arregla solo con la memoria."

"También vimos un bloque que es exactamente una función
anónima. No tiene nombre."

[ :a :b :c :d | a+b+c+d ] valueWithArguments: {1. 5. 6. 8} .

"Hace que algunas cosas sean más ágiles como filtrar un
vector."
```

Mensaje

Reemplaza a:

- Operando.
- Procedimiento.
- Función.

Clase

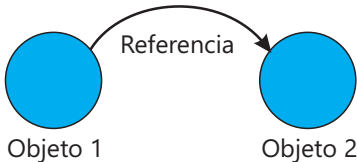
Si los objetos son son "estructuras de datos"...

entonces las Clases son los "Typedef"
... también con "mucha vitamina".

Definen estructura y **mensajes** de sus **instancias**

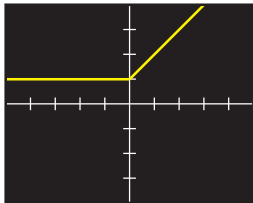
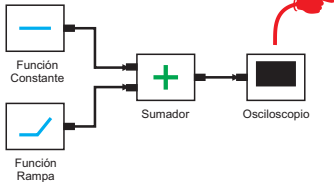
Referencias

Se manejan todas como punteros
que controla automáticamente
smalltalk.



Ejemplo "Simulink"

El software Matlab provee un entorno gráfico que permite simular señales en el tiempo y aplicarle funciones con un aspecto similar al siguiente:



¿Cómo lo implementaríamos en lenguaje C?

Funciones

Tipos de datos

Variables

Estructuras

Punteros

En lenguaje Smalltalk

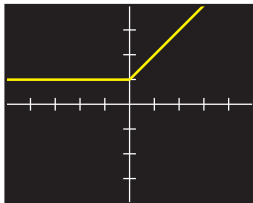
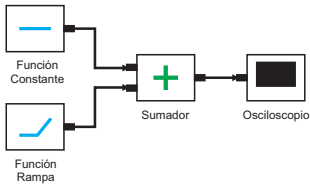
Sólo Objetos...

¿Qué hace?

¿Cómo lo programo?

En lenguaje Smalltalk

¿Cuántos objetos podríamos definir?



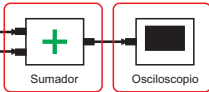
En lenguaje Smalltalk

¿Cuántos objetos podríamos definir?

1 - Bloque Constante



3 - Bloque Sumador

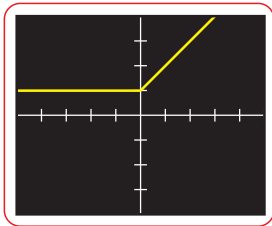


Bloque Osciloscopio - 4

Función Rampa

2 - Bloque Función rampa

5 - Gráfico



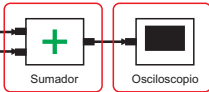
En lenguaje Smalltalk

¿Alguno más?

1 - Bloque Constante



3 - Bloque Sumador

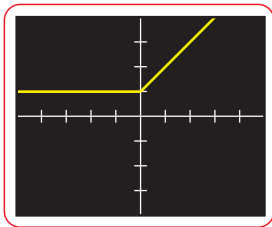


Bloque Osciloscopio - 4



2 - Bloque Función rampa

5 - Gráfico



En lenguaje Smalltalk

¿Alguno más?

6 - Modelo

1 - Bloque Constante



Función
Constante

3 - Bloque Sumador



Sumador



Osciloscopio

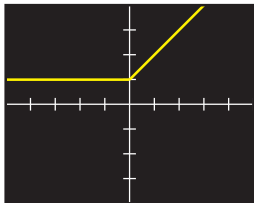
Bloque Osciloscopio - 4



Función
Rampa

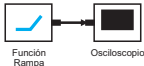
2 - Bloque Función rampa

5 - Gráfico



En lenguaje Smalltalk

Ejemplo 1. Textual.



```
Workspace

"Ejemplo 1 simular una función rampa
-----"

"Creo el modelo"
modelo1 := Modelo new.

"Establezco los parámetros de simulación"
modelo1 ti: -10. "Tiempo inicial"
modelo1 tf: 10. "Tiempo final"
modelo1 dt: 0.1. "Paso de simulación"

"Creo el bloque función rampa"
bloque1 := BloqueFuncionRampaT newConNombre: 'rampa' yModelo: modelo1.

"Creo el bloque osciloscopio"
bloqueS1 := BloqueSalida newConNombre: 'salida' yModelo: modelo1.

"Conecto la salida del bloque función rampa con la entrada del bloque
osciloscopio"
bloque1 salida: 1 conectateCon: bloqueS1 entrada: 1.

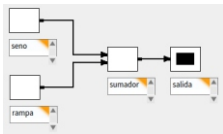
"Simulo el modelo"
modelo1 simulate.

"Muestro un array con el resultado de la simulación"
bloqueS1 salidasEnElTiempo.

"Muestro el resultado de la simulación en un gráfico"
modelo1 muestraOsciloscopioConNombre: 'salida'.
```

Ejemplo 2

Gráfico.



```
Workspace

"Ejemplo 2 simular la suma de una función rampa y una función seno"
-----"

"Creo el modelo"
modelo2 := Modelo new.

"Establezco los parámetros de simulación"
modelo2 ti: -10. "Tiempo inicial"
modelo2 tf: 10. "Tiempo final"
modelo2 dt: 0.1. "Paso de simulación"

"Creo los bloques"
bloque2 := BloqueFuncionRampaT newConNombre: 'rampa' yModelo: modelo2.
bloque3 := BloqueFuncionSenoT newConNombre: 'seno' yModelo: modelo2.
bloque4 := BloqueSumador newConNombre: 'sumador' yModelo: modelo2.
bloqueS2 := BloqueSalida newConNombre: 'salida' yModelo: modelo2.

"Creo los bloques gráficos"
bloqueMorph2 := BloqueMorph newConBloque: bloque2.
bloqueMorph3 := BloqueMorph newConBloque: bloque3.
bloqueMorph4 := BloqueMorph newConBloque: bloque4.
bloqueMorphS2 := BloqueOsciloscopio newConBloque: bloqueS2.

"Muestro el resultado de la simulación en un gráfico"
modelo2 muestraOsciloscopioConNombre: 'salida'.

"Inspect"
bloqueMorphS2 inspect.

"Elimino el modelo, eliminando los gráficos asociados"
modelo2 borrate.
```

Inspector

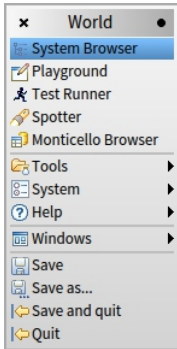
Inspector on a BloqueOsciloscopio(683409408)

a BloqueOsciloscopio(683409408)

Raw Extension Submorphs Morph Keys Meta

Variable	Value
self	a BloqueOsciloscopio(683409408)
bloque	a BloqueSalida
conexionesDeEntrada	an OrderedCollection [1 item] (an Array(a BloqueSumador 1))
array	an Array [10 items] (an Array(a BloqueSumador 1) nil nil ni...
1	an Array [2 items] (a BloqueSumador 1)
1	a BloqueSumador
conexionesDeEntrada	an OrderedCollection [2 items] (an Array(a BloqueFuncionSenoT 1) an Ar...
array	an Array [10 items] (an Array(a BloqueFuncionSenoT 1) an Ar...
1	an Array [2 items] (a BloqueFuncionSenoT 1)
1	a BloqueFuncionSenoT
2	1
2	an Array [2 items] (a BloqueFuncionRampaT 1)
1	a BloqueFuncionRampaT
2	1
3	nil
4	nil
5	nil
6	nil

System Browser



The screenshot shows the IDE4PLC interface with the following components and annotations:

- Package Explorer (Left):** Lists packages like 'Last Modified Classes', 'Most Viewed Classes', 'Work', and 'IDE4PLC-ProyectoComputadores2'. A red box highlights this area with the text: **Packages "Contenedores de clases"**.
- Class Explorer (Middle-Left):** Shows classes under the 'Bloque' package, including 'BloqueDerivador', 'BloqueFuncionConstante', 'BloqueFuncionRampaT', 'BloqueFuncionSenoT', 'BloqueFuncionSinT', 'BloqueSumador', and 'BloqueMorph'. A red box highlights this area with the text: **Clases dentro del Package seleccionado**.
- Protocol Explorer (Middle-Right):** Shows protocols for the selected class, including 'all', 'accessing', 'acciones', and 'initialize-release'. A red box highlights this area with the text: **Protocols "Contenedores de métodos"**.
- Method Explorer (Right):** Shows methods for the selected protocol, including 'leerEntradaEnTiempo', 'modelo', 'nombre', 'removeEntrada', and 'salidas'. A red box highlights this area with the text: **Método dentro del Protocol seleccionado**.
- Source Editor (Bottom):** Displays the source code for the selected method. A red box highlights this area with the text: **Código fuente del método seleccionado**.

Source Code:

```
salida: unNumeroDeSalida conectateCon: unBloque entrada: unNumeroDeEntrada.  
"Conecta la salida del bloque al que le mando este mensaje con la entrada del otro bloque.  
Para lograrlo, pone en la OrderedCollection 'conexionesDeEntrada' de 'unBloque' en la posición  
'unNumeroDeEntrada' que tiene un array de 2 elementos cuyo primer valor 'self' (el bloque al  
que le mandan el mensaje) y con 'unNumeroDeSalida' como segundo valor del array."  
  
(unBloque conexionesDeEntrada at: unNumeroDeEntrada)  
  at: 1 put: self;  
  at: 2 put: unNumeroDeSalida.
```

IDE Interface showing the class hierarchy for **Bloque**.

Left Panel (Project Explorer):

- IDE4PLC-ProyectoComputadores2

Center Panel (Class Hierarchy):

- Bloque
 - BloqueDerivador
 - BloqueFuncionConstante
 - BloqueFuncionRampaT
 - BloqueFuncionSenoT
 - BloqueFuncionSincT
 - BloqueFuncionTAlCuadrado
 - BloqueGanancia
 - BloqueRetardo
 - BloqueSalida
 - BloqueSumador
 - BloqueMorph

Right Panel (Class Details):

- all --
- accessing
- acciones
- initialize-release
- SosBloqueDeSalida
- SosBloqueDerivador
- agregarEntrada
- agregarSalida
- cantidadDeEntradas
- cantidadDeSalidas
- condicionInicial:ParaLaSalida:
- conexionesDeEntrada
- conexionesDeEntrada:
- deconectarEntrada:
- entradas
- entradas:
- guardarValorDeEntrada:

Bottom Panel (Object Inspector):

```
Object subclass: #Bloque
instanceVariableNames: 'conexionesDeEntrada entradas salidas nombre modelo'
classVariableNames: ''
category: 'IDE4PLC-ProyectoComputadores2'
```

Annotation: A red arrow points to the instance variable names in the Object Inspector, labeled "Componentes del objeto".

ide

Last Modified Classes
Most Viewed Classes
Work
IDE4PLC-ProyectoComputadores2

Bloque
BloqueDerivador
BloqueFuncionConstante
BloqueFuncionRampaT
BloqueFuncionSenoT
BloqueFuncionSincT
BloqueFuncionT
BloqueFuncionTAlCuadrado
BloqueGanancia
BloqueRetardo
BloqueSalida
BloqueSumador
BloqueMorph

-- all --
accessing
acciones
initialize-release

leerEntradaSenTiempos:
modelo
modelo:
nombre
nombre:
removeEntrada:
removeSalida:
resetearEntradas
salida:
salida:conectateCon:entrada:
salida:enTiempo:
salidas
salidas:

Groups

Hierarchy

☐ Class side

Comments

History Navigator

salida: unNumeroDeSalida conectateCon: unBloque entrada: unNumeroDeEntrada.

"Conecta la salida del bloque al que le mando este mensaje con la entrada del otro bloque. Para lograrlo, pone en la OrderedCollection 'conexionesDeEntrada' de 'unBloque' en la posicion 'unNumeroDeEntrada' que tiene un array de 2 elementos cuyo primer valor 'self' (el bloque al que le mandan el mensaje) y con 'unNumeroDeSalida' como segundo valor del array."

```
(unBloque conexionesDeEntrada at: unNumeroDeEntrada)
```

```
at: 1 put: self;
```

```
at: 2 put: unNumeroDeSalida.
```

ide

Last Modified Classes

Most Viewed Classes

Work

IDE4PLC-ProyectoComputadores2

Bloque

BloqueDerivador

BloqueFuncionConstante

BloqueFuncionRampaT

BloqueFuncionSenoT

BloqueFuncionSincT

BloqueFuncionT

BloqueFuncionTAICuadrado

BloqueGanancia

BloqueRetardo

BloqueSalida

BloqueSumador

BloqueMorph

-- all --

acciones

initialize-release

funcionEnTiempo:

initialize

Groups

Hierarchy

☐ Class side

Comments

History Navigator

functionEnTiempo: t

"Calcula lo que hace el bloque en el tiempo t..
Y guarda el resultado en la salida correspondiente.
Este BloqueFuncionRampaT devuelve una funcion rampa ($f(t)=0$ si $t<0$, $f(t) = t$ si $t\geq 0$)."

salidas at: 1 put: ((t < 0) ifTrue: [0] ifFalse: [t]).

51

¿Preguntas?

¡Gracias por participar!

Dudas, comentarios o sugerencias
por favor comuníquese a
ide4plc@gmail.com
ericpernia@gmail.com