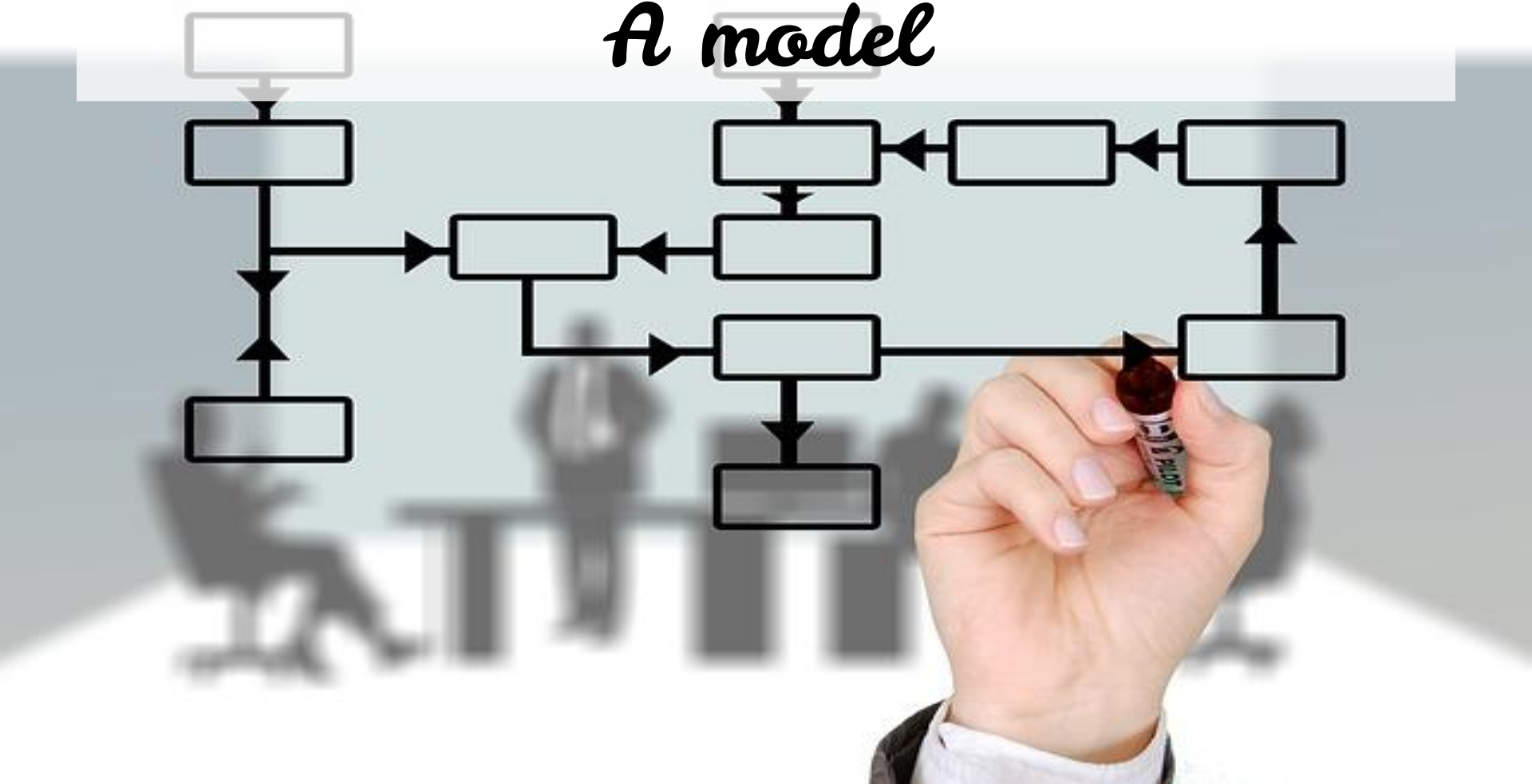
Arduino Designer

The making of!

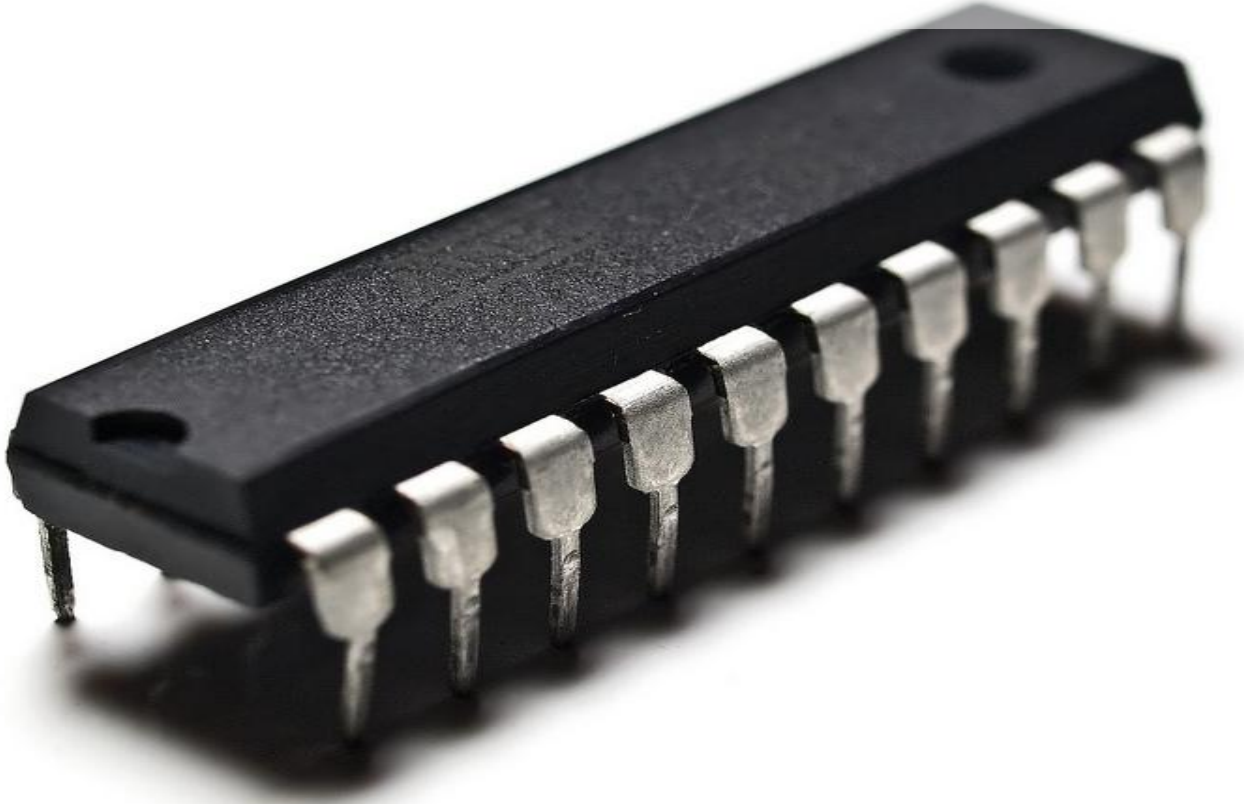
Mélanie Bats
melanie.bats@obeo.fr
@melaniebats

Obeo
Model Driven Company

A model

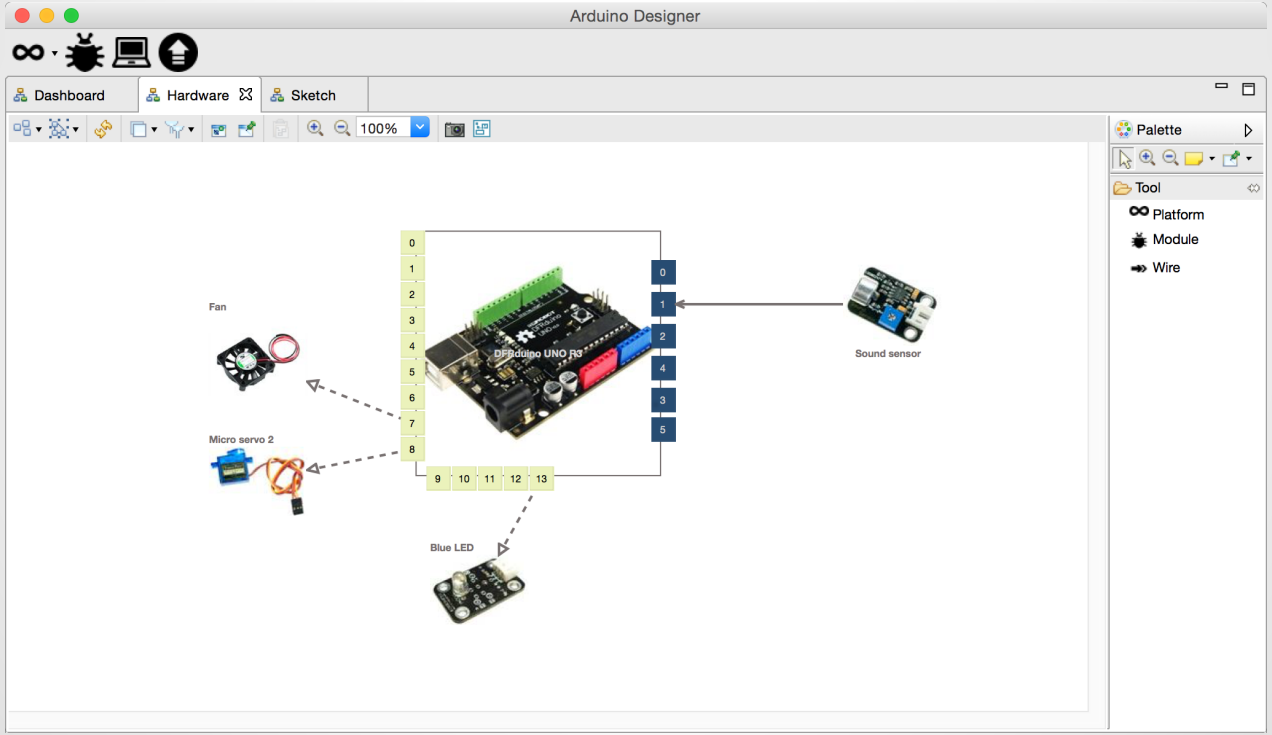


A micro controller



And a cat ?

Arduino Designer



A dedicated tooling

Graphical Programming

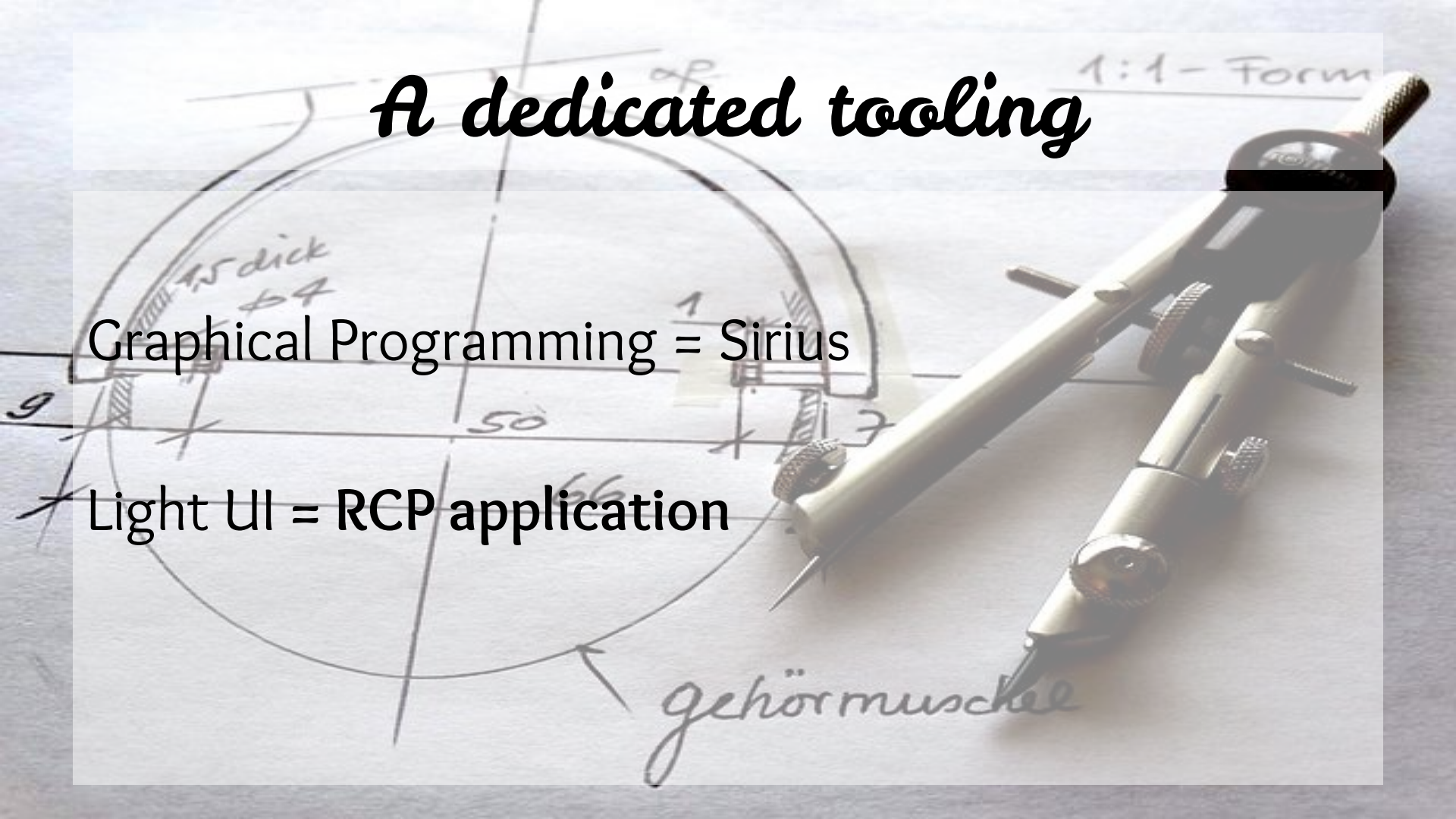
Light UI

A dedicated tooling

Graphical Programming = **Sirius**

Light UI

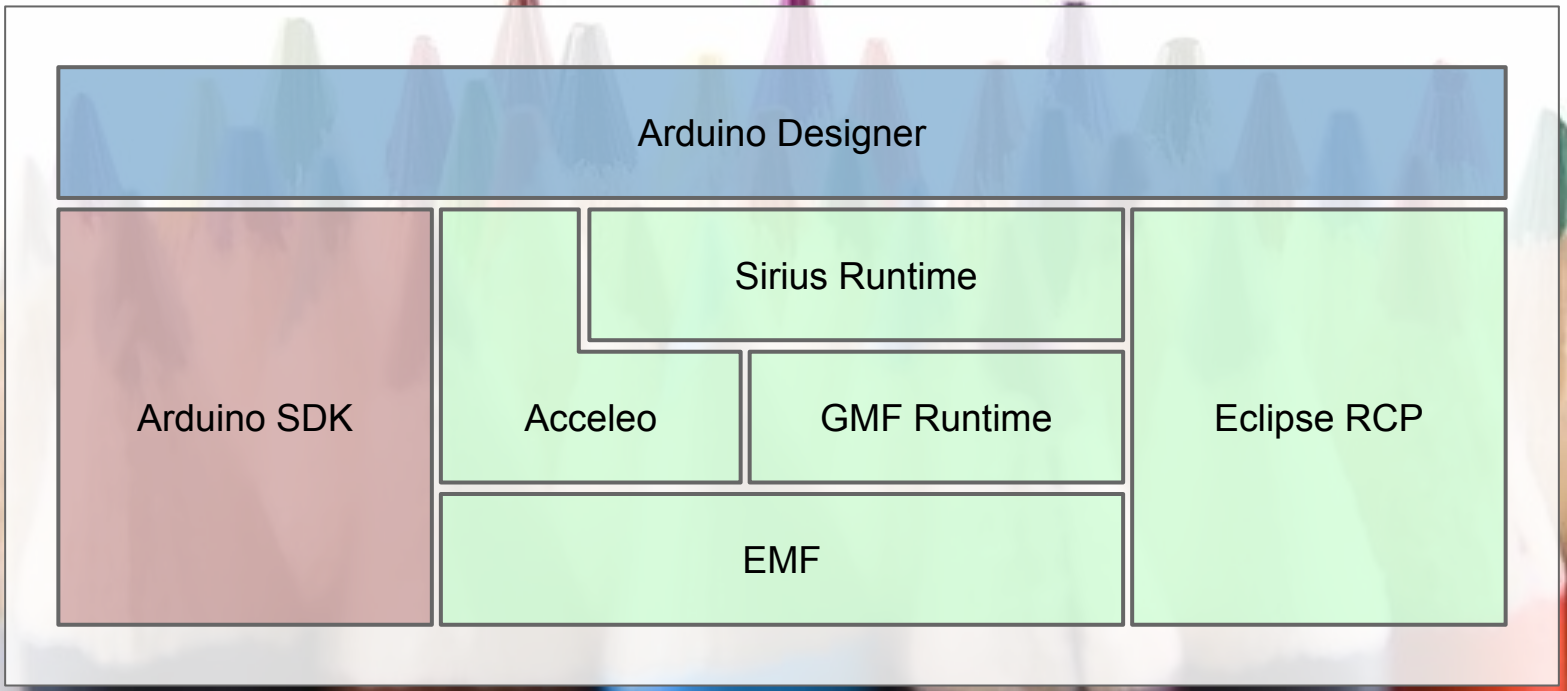
A dedicated tooling



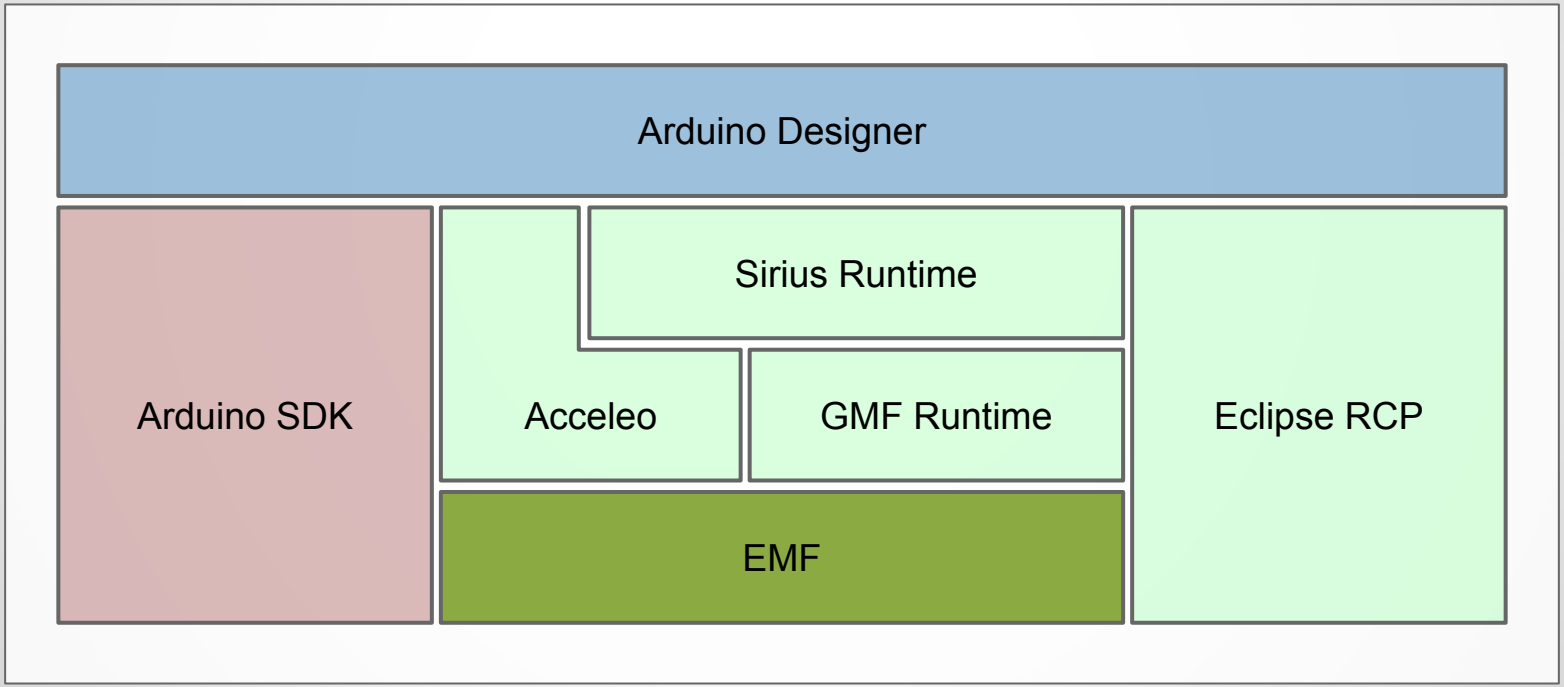
Graphical Programming = Sirius

Light UI = **RCP application**

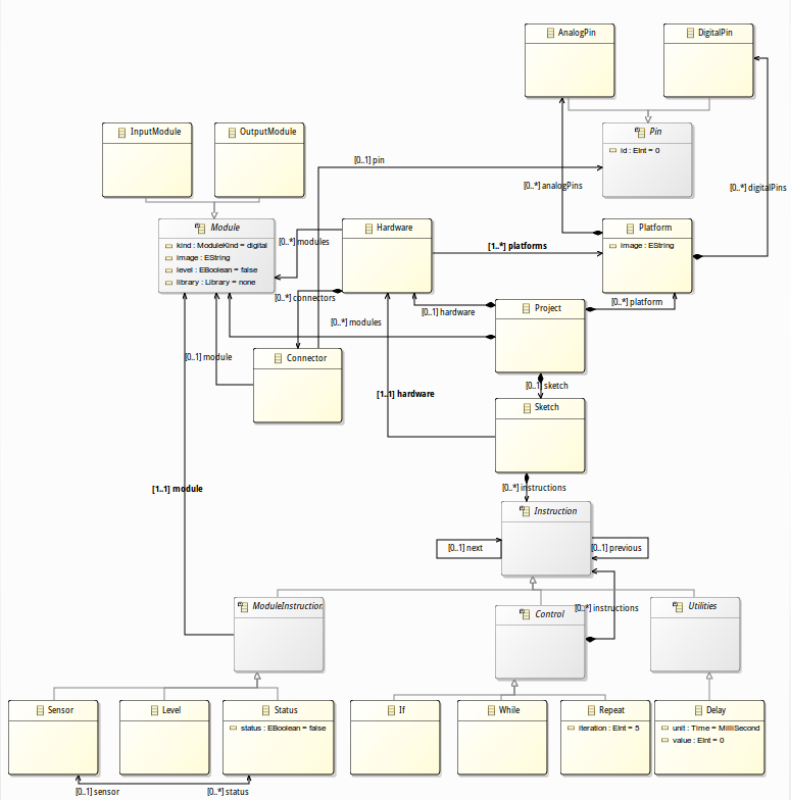
Create graphical editor

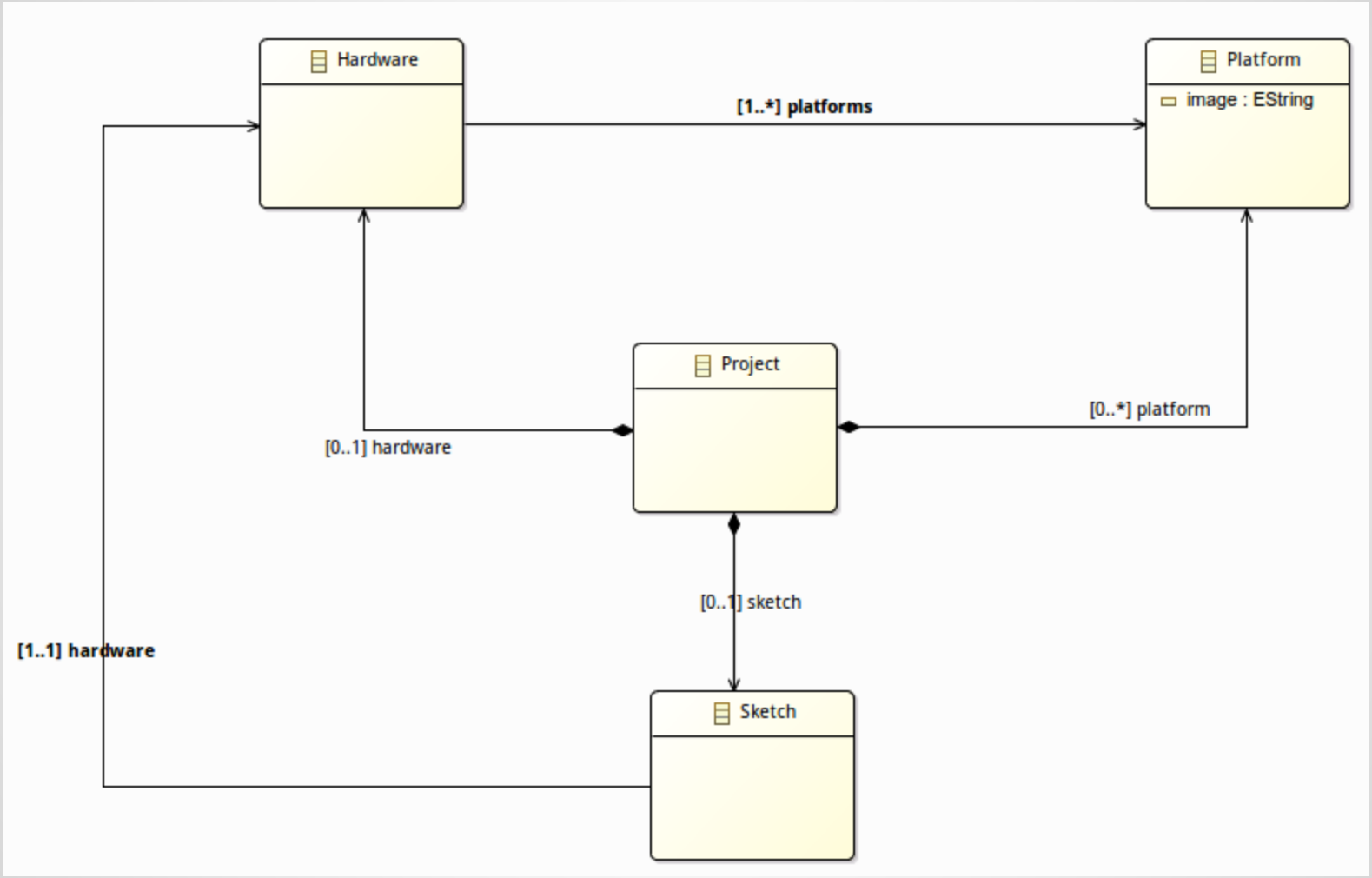


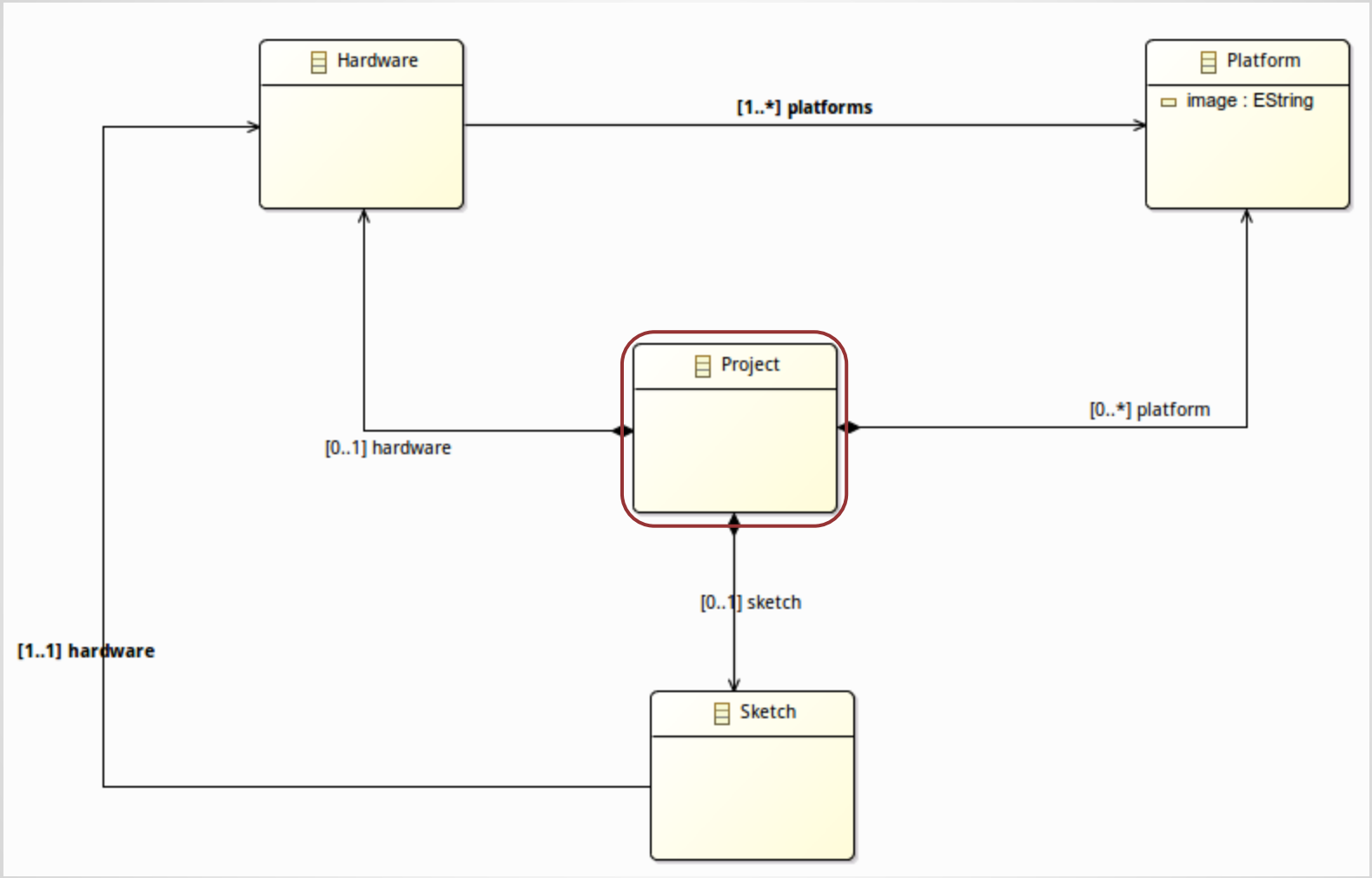
Data

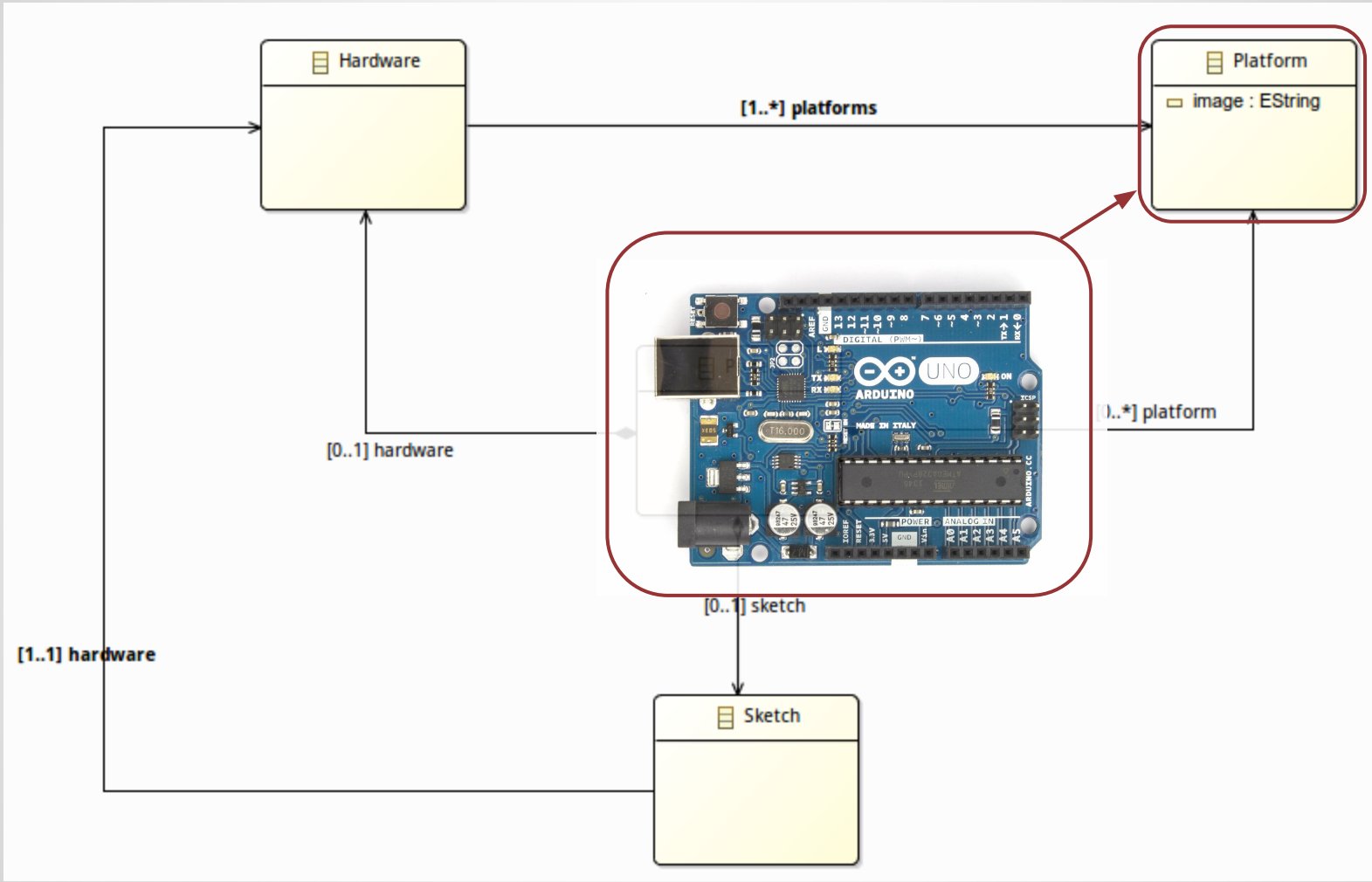

Arduino Designer
Arduino SDK
Acceleo
Sirius Runtime
GMF Runtime
EMF
Eclipse RCP

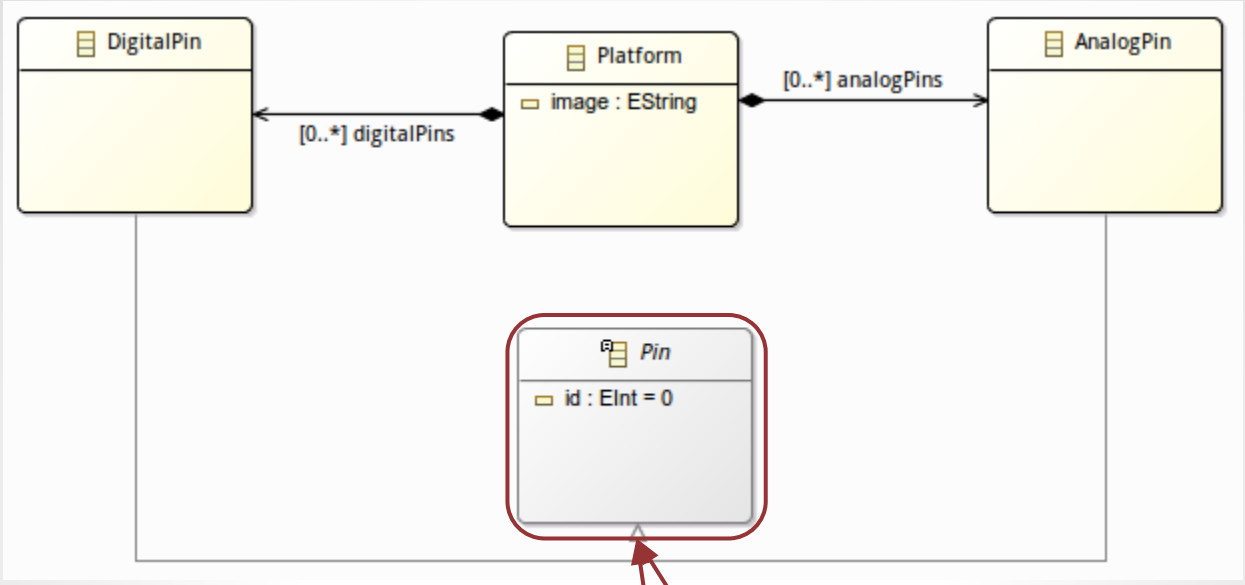
Arduino DSL

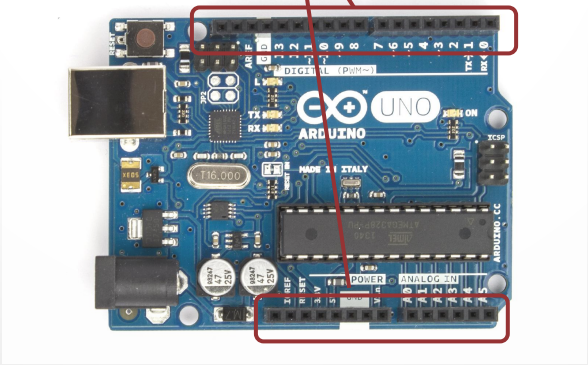

AnalogPin
DigitalPin
InputModule
OutputModule
Pin
id : EInt = 0
[0..1] pin
[0..*] analogPins
[0..*] digitalPins
Module
kind : ModuleKind = digital
image : EString
level : EBoolean = false
library : Library = none
[0..*] modules
Hardware
[1..*] platforms
Platform
image : EString
[0..*] connectors
[0..1] hardware
[0..*] platform
Project
[0..*] modules
[0..1] module
Connector
[0..1] sketch
[1..1] hardware
Sketch
[1..1] module
[0..*] instructions
Instruction
[0..1] next
[0..1] previous
ModuleInstruction
Control
[0..*] instructions
Utilities
Sensor
Level
Status
status : EBoolean = false
If
While
Repeat
iteration : EInt = 5
Delay
unit : Time = MilliSecond
value : EInt = 0
[0..1] sensor
[0..*] status

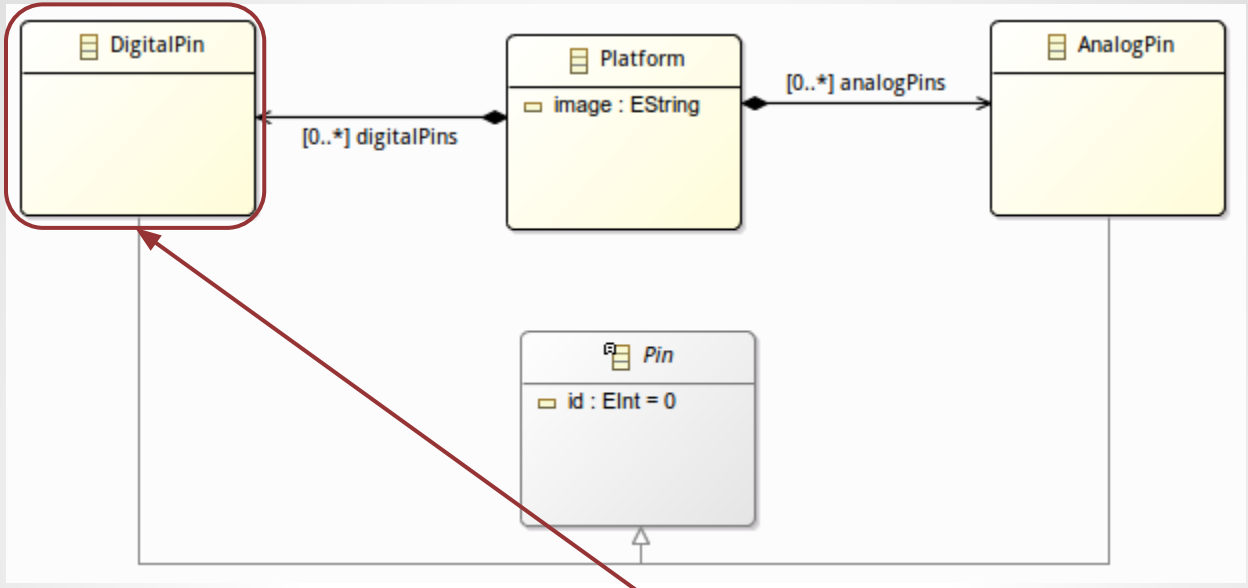

Hardware
Platform
image : EString
[1..*] platforms
Project
[0..1] hardware
[0..*] platform
[0..1] sketch
[1..1] hardware
Sketch

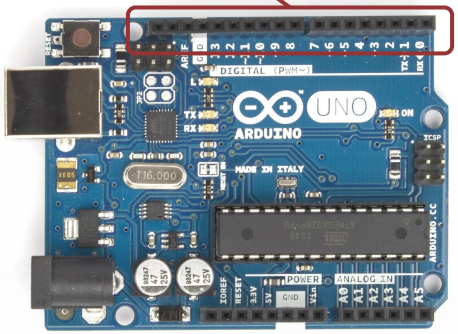

Hardware
Platform
image : EString
[1..*] platforms
Project
[0..1] hardware
[0..*] platform
[0..1] sketch
[1..1] hardware
Sketch

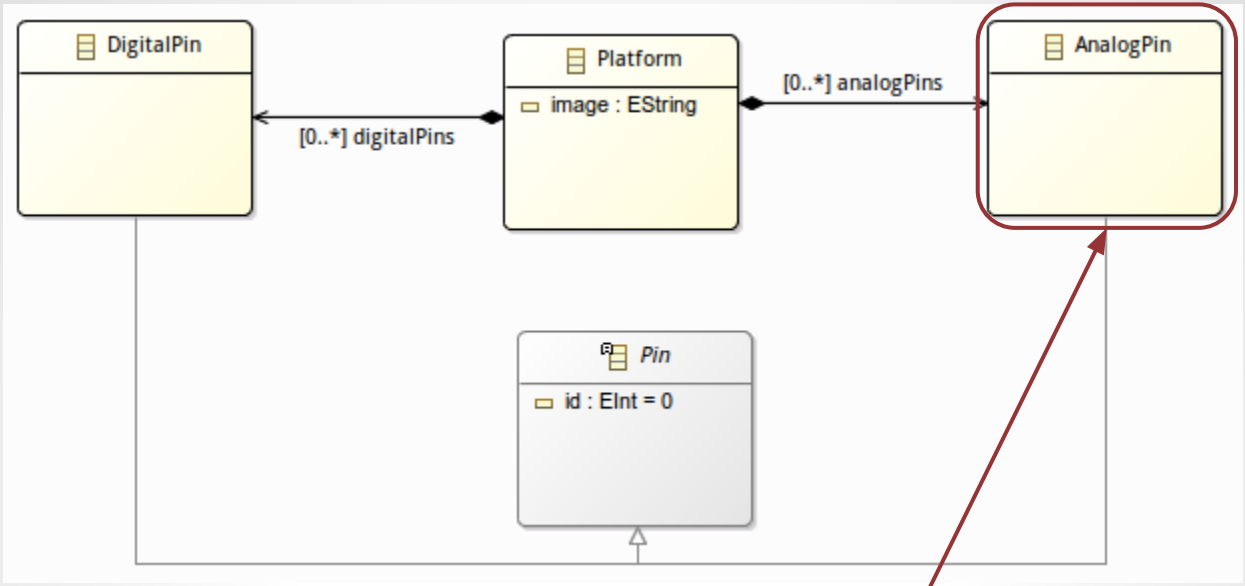

Hardware
[1..*] platforms
Platform
image : EString
[0..1] hardware
..*] platform
[0..1] sketch
[1..1] hardware
Sketch

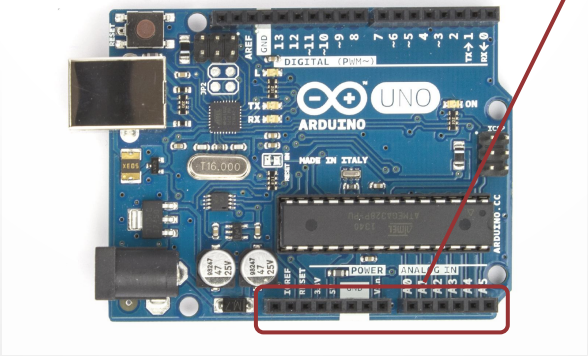

DigitalPin
Platform
image : EString
AnalogPin
[0..*] digitalPins
[0..*] analogPins
Pin
id : EInt = 0

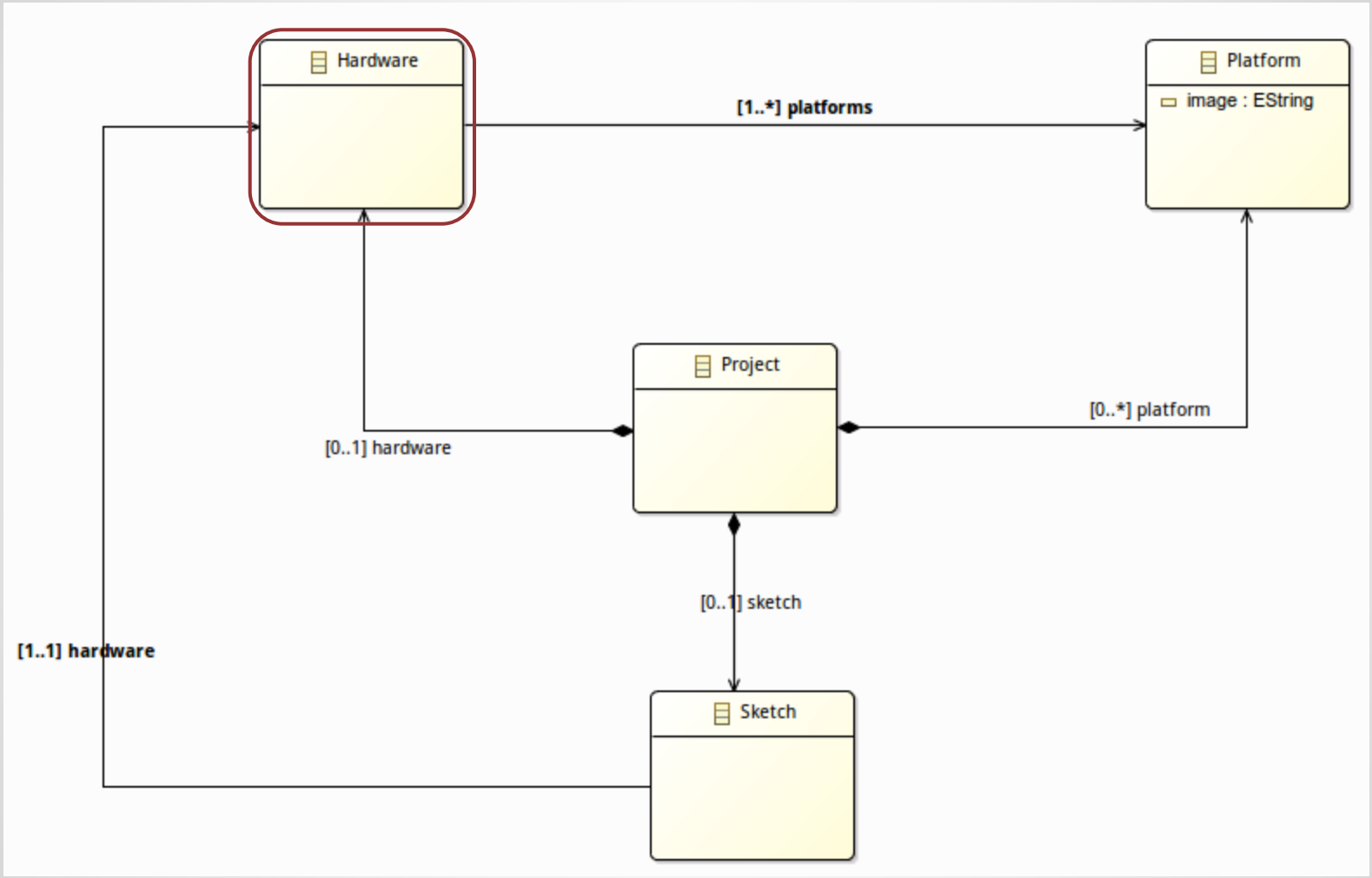

UNO
ARDUINO
POWER
ANALOG IN

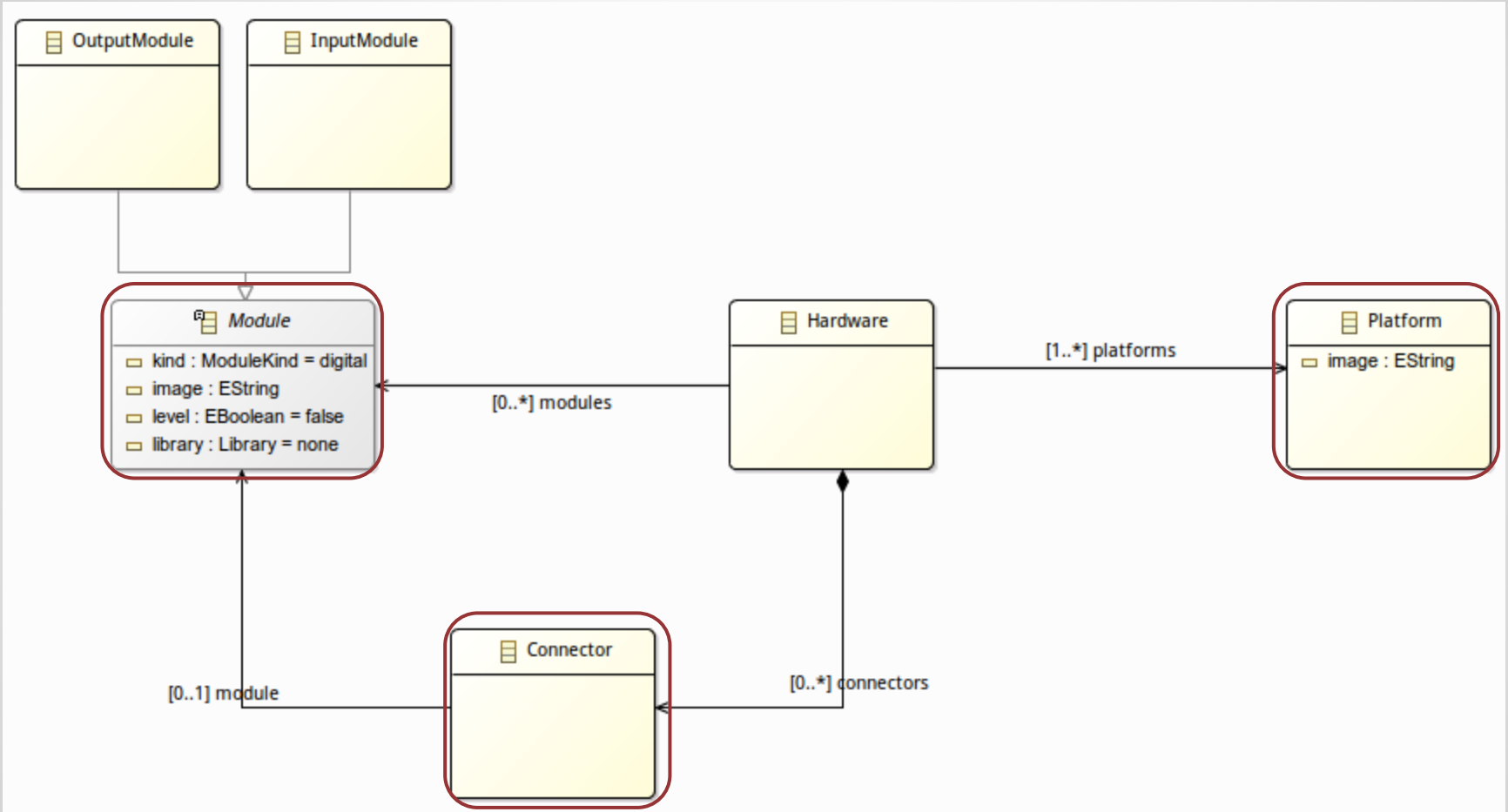

DigitalPin
Platform
image : EString
AnalogPin
[0..*] digitalPins
[0..*] analogPins
Pin
id : EInt = 0

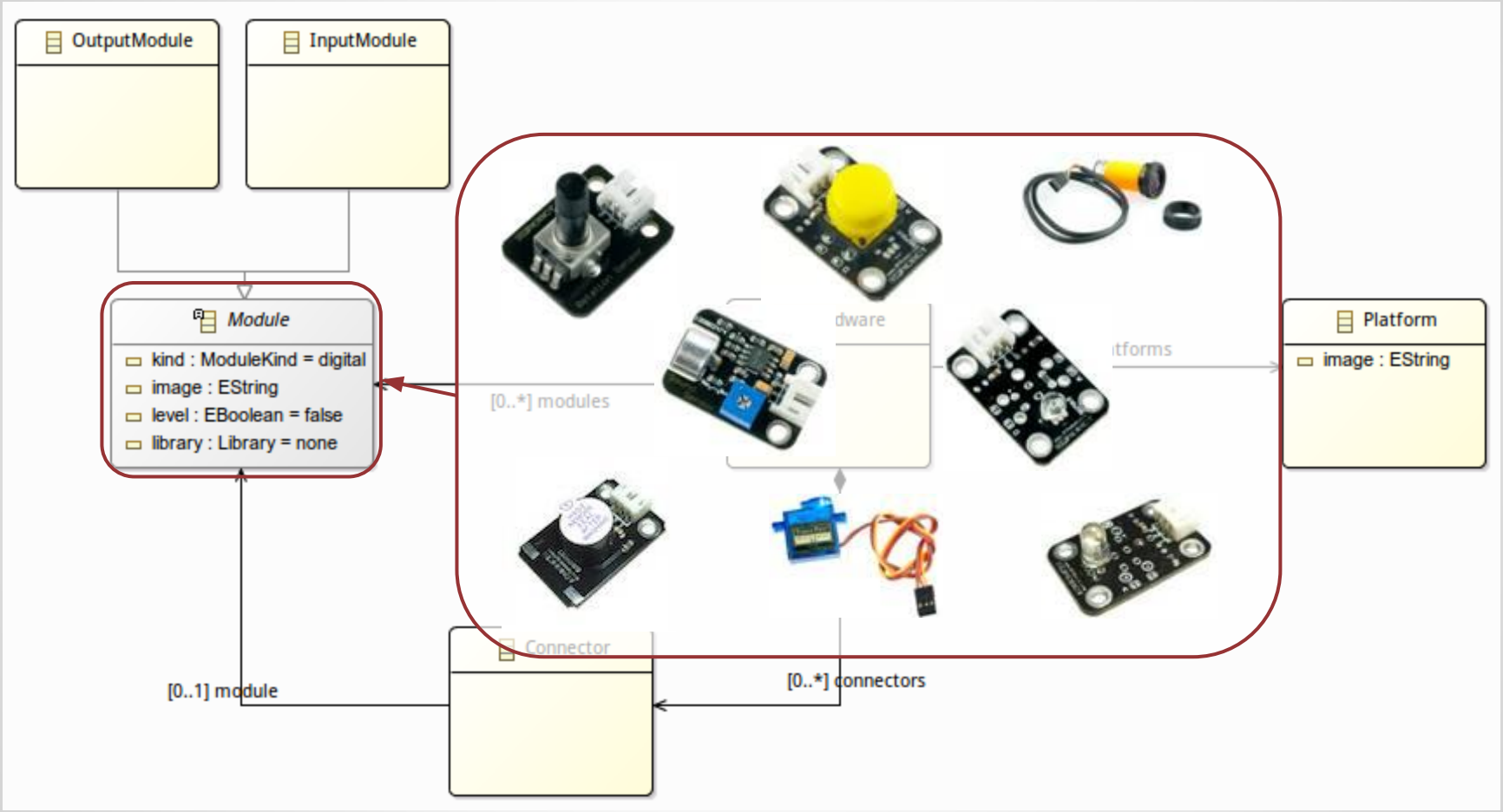

UNO
ARDUINO
MADE IN ITALY
DIGITAL (PWM~)
POWER
ANALOG IN

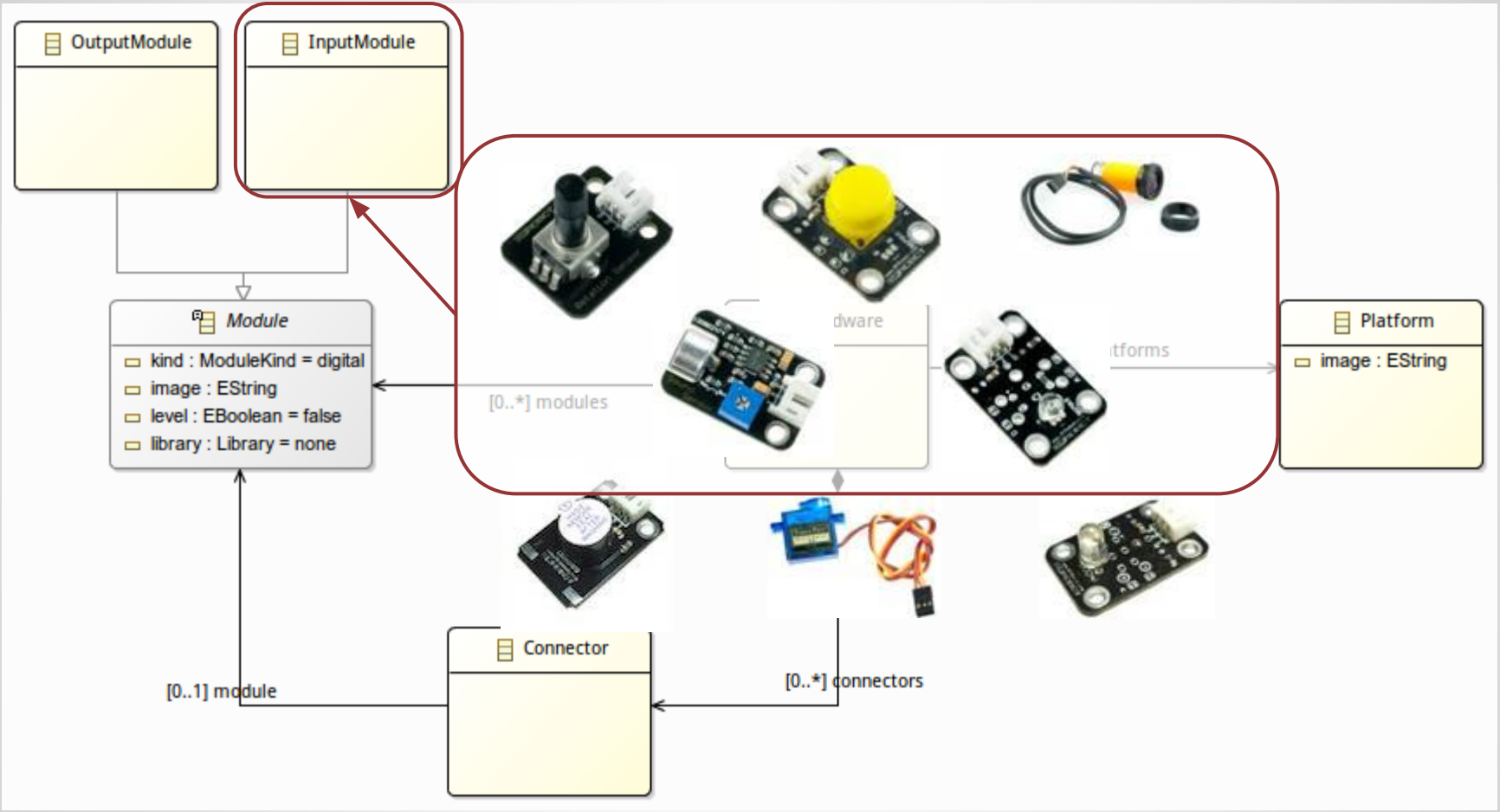

DigitalPin
Platform
image : EString
AnalogPin
[0..*] digitalPins
[0..*] analogPins
Pin
id : EInt = 0

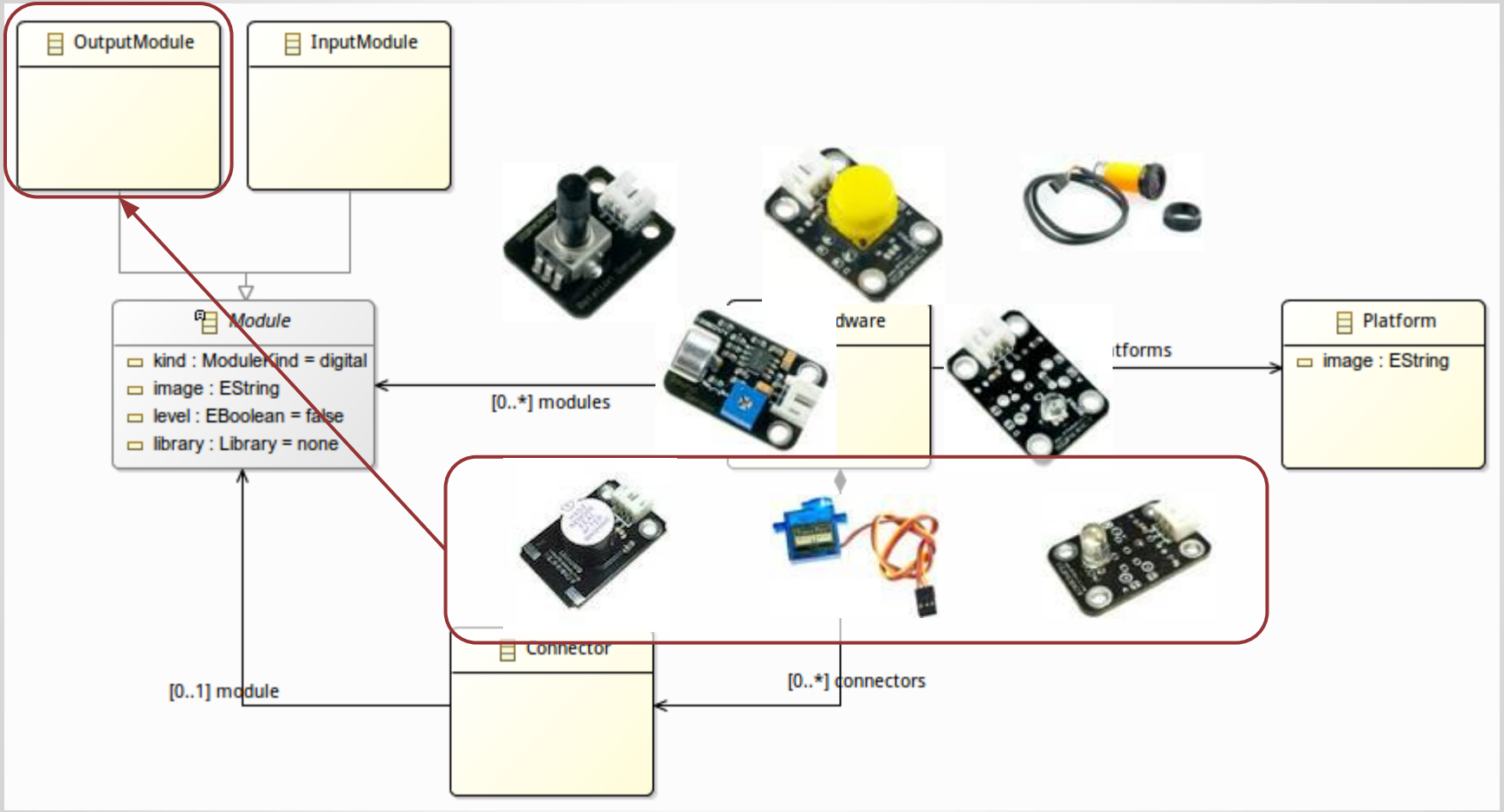

UNO
ARDUINO
MADE IN ITALY
DIGITAL (PWM~)
POWER
ANALOG IN

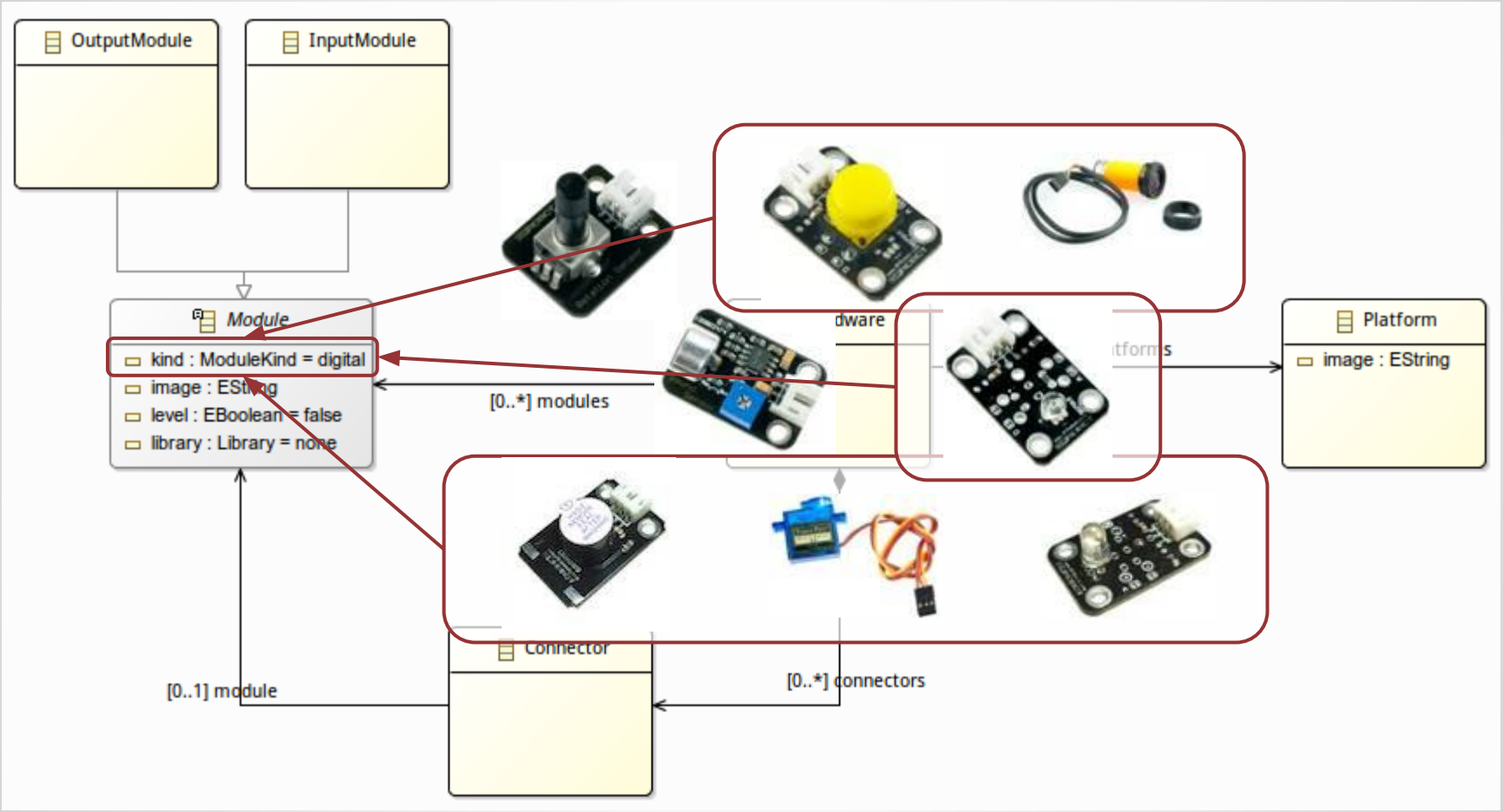

Hardware
Platform
image : EString
[1..*] platforms
Project
[0..1] hardware
[0..*] platform
[0..1] sketch
[1..1] hardware
Sketch

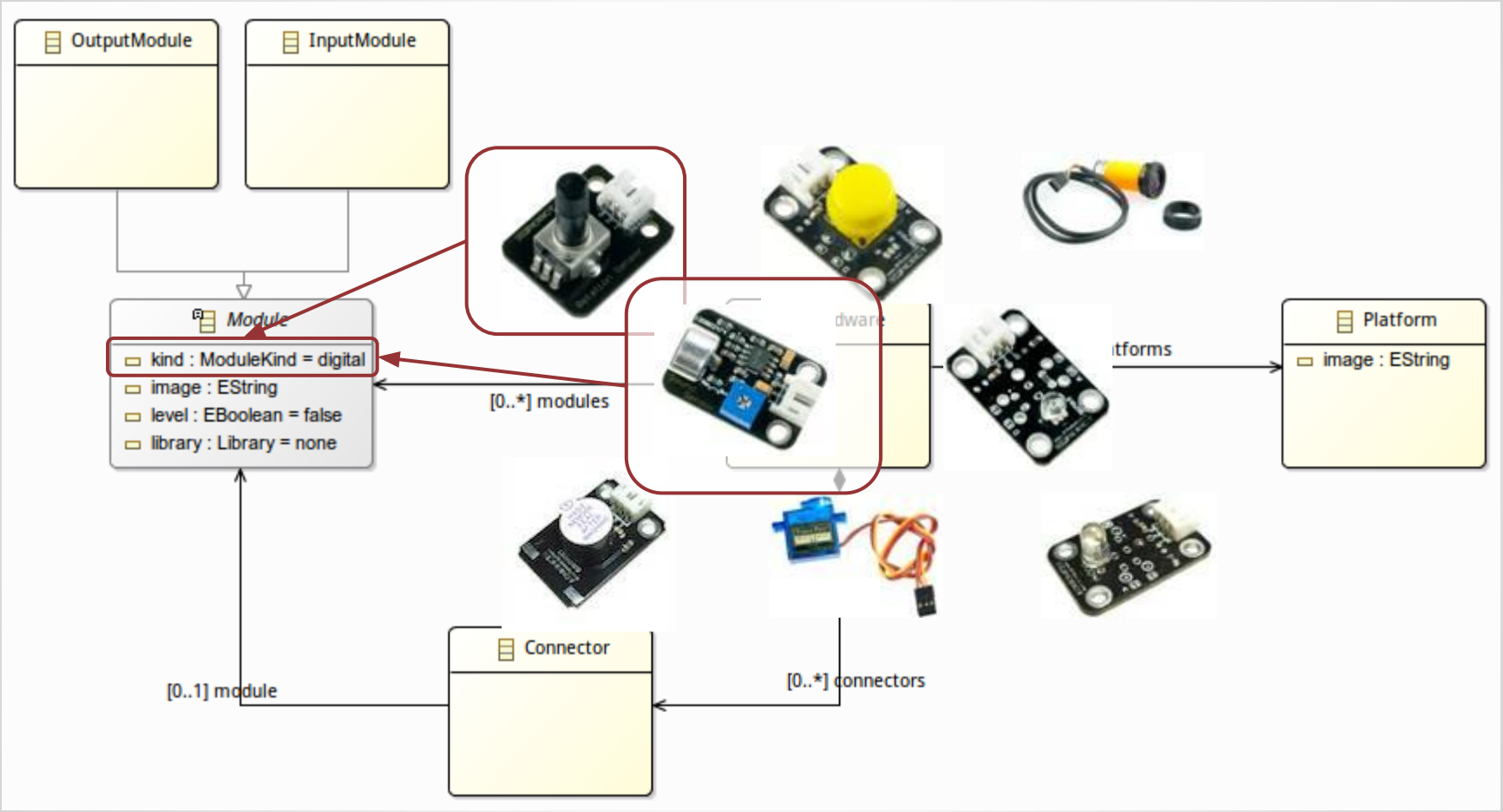

OutputModule
InputModule
Module
kind : ModuleKind = digital
image : EString
level : EBoolean = false
library : Library = none
Hardware
Platform
image : EString
[1..*] platforms
[0..*] modules
Connector
[0..1] module
[0..*] connectors

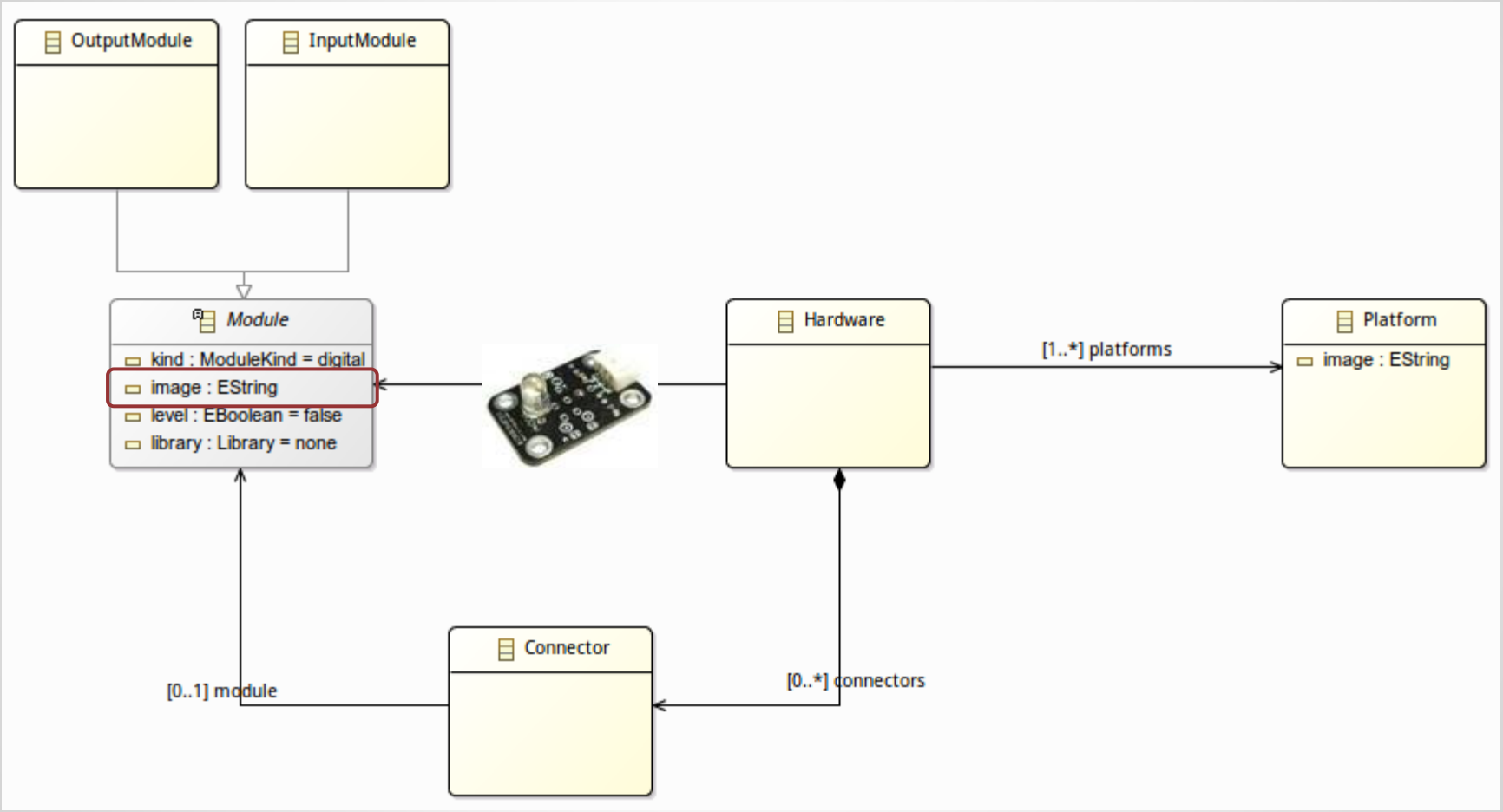

OutputModule
InputModule
Module
kind : ModuleKind = digital
image : EString
level : EBoolean = false
library : Library = none
[0..*] modules
dware
itforms
Platform
image : EString
Connector
[0..1] module
[0..*] connectors

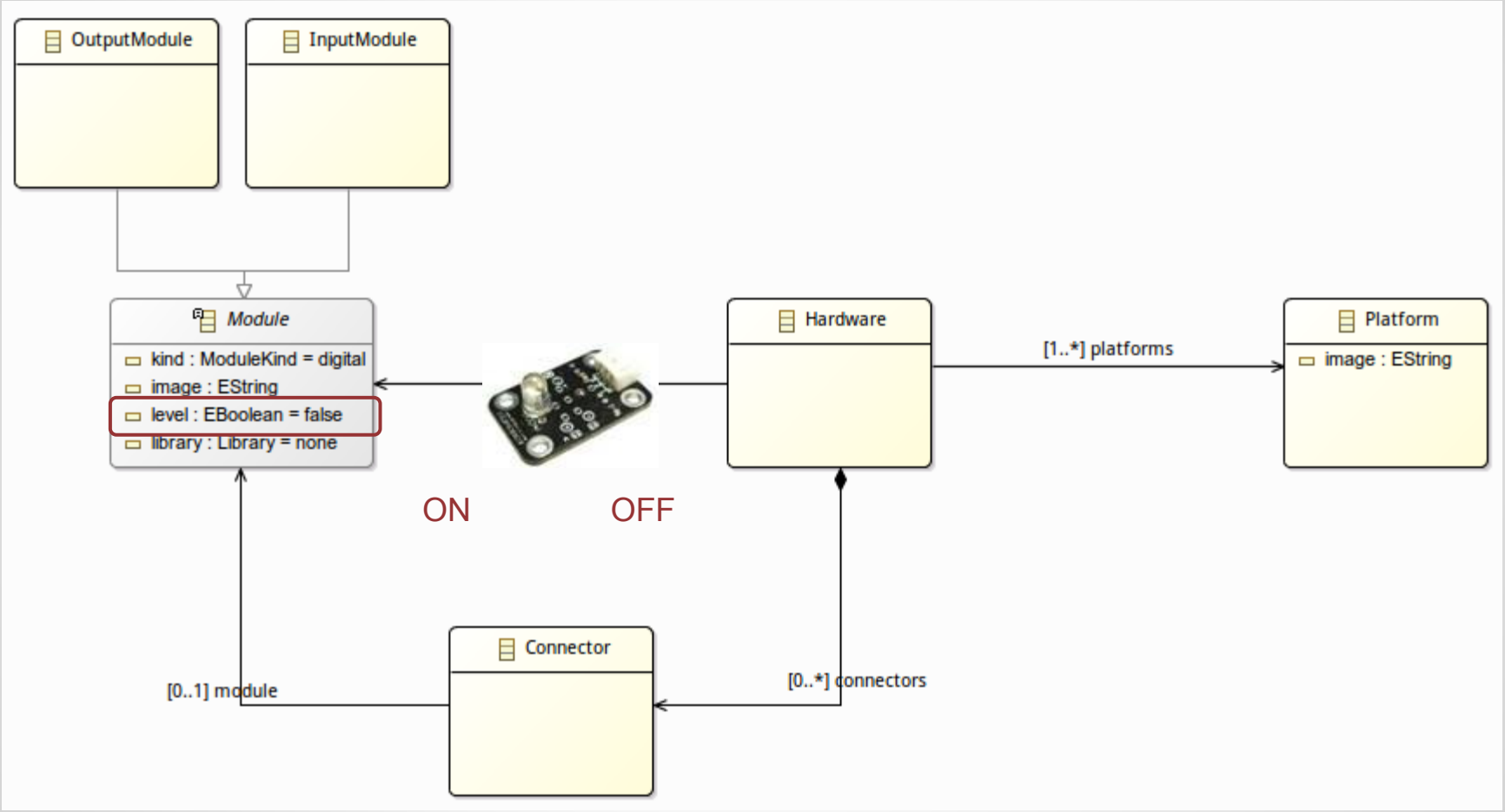

OutputModule
InputModule
Module
kind : ModuleKind = digital
image : EString
level : EBoolean = false
library : Library = none
[0..*] modules
dware
tforms
Platform
image : EString
Connector
[0..1] module
[0..*] connectors

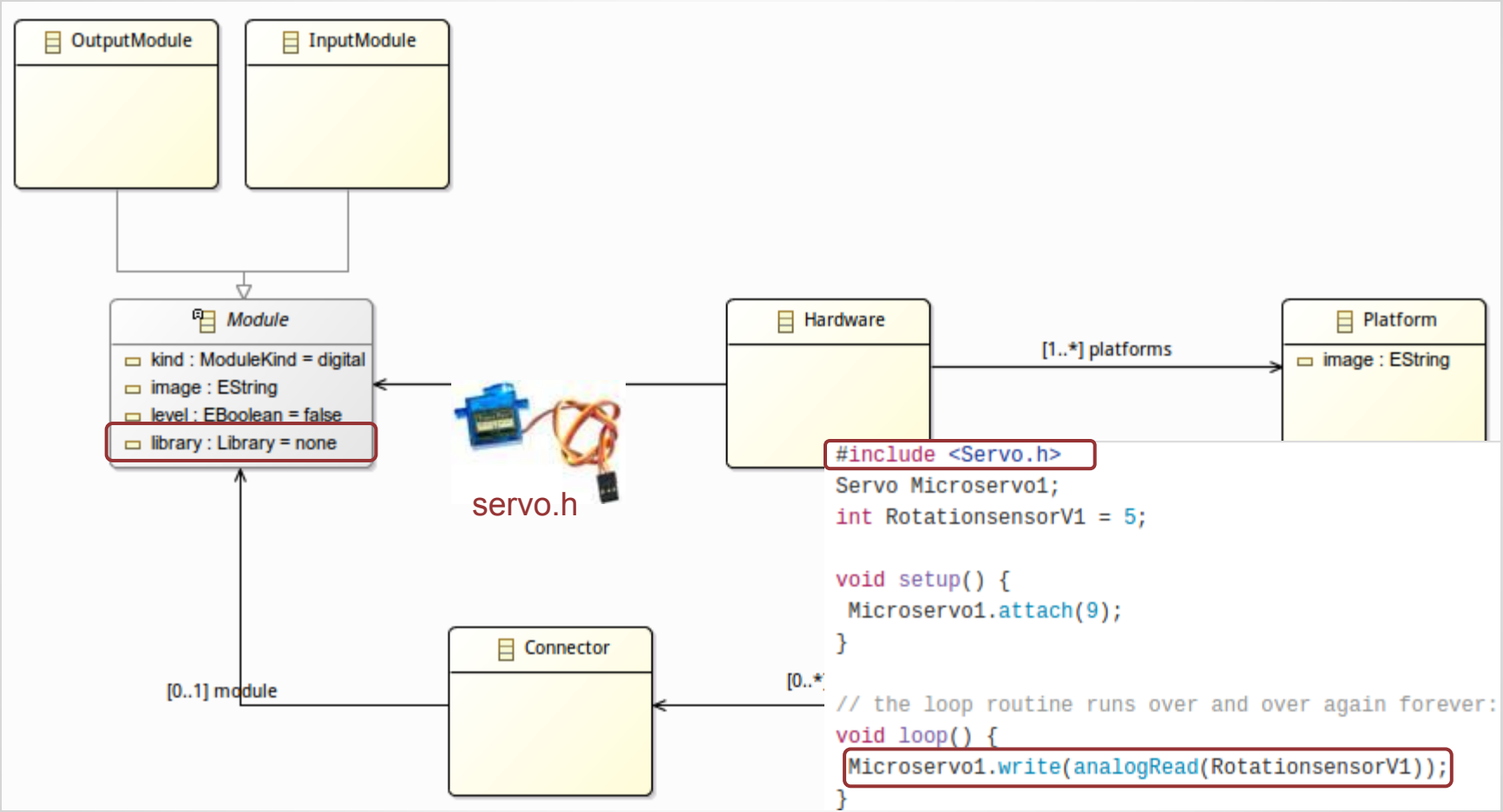

OutputModule
InputModule
Module
kind : ModuleKind = digital
image : EString
level : EBoolean = false
library : Library = none
[0..*] modules
dware
tforms
Platform
image : EString
Connector
[0..1] module
[0..*] connectors

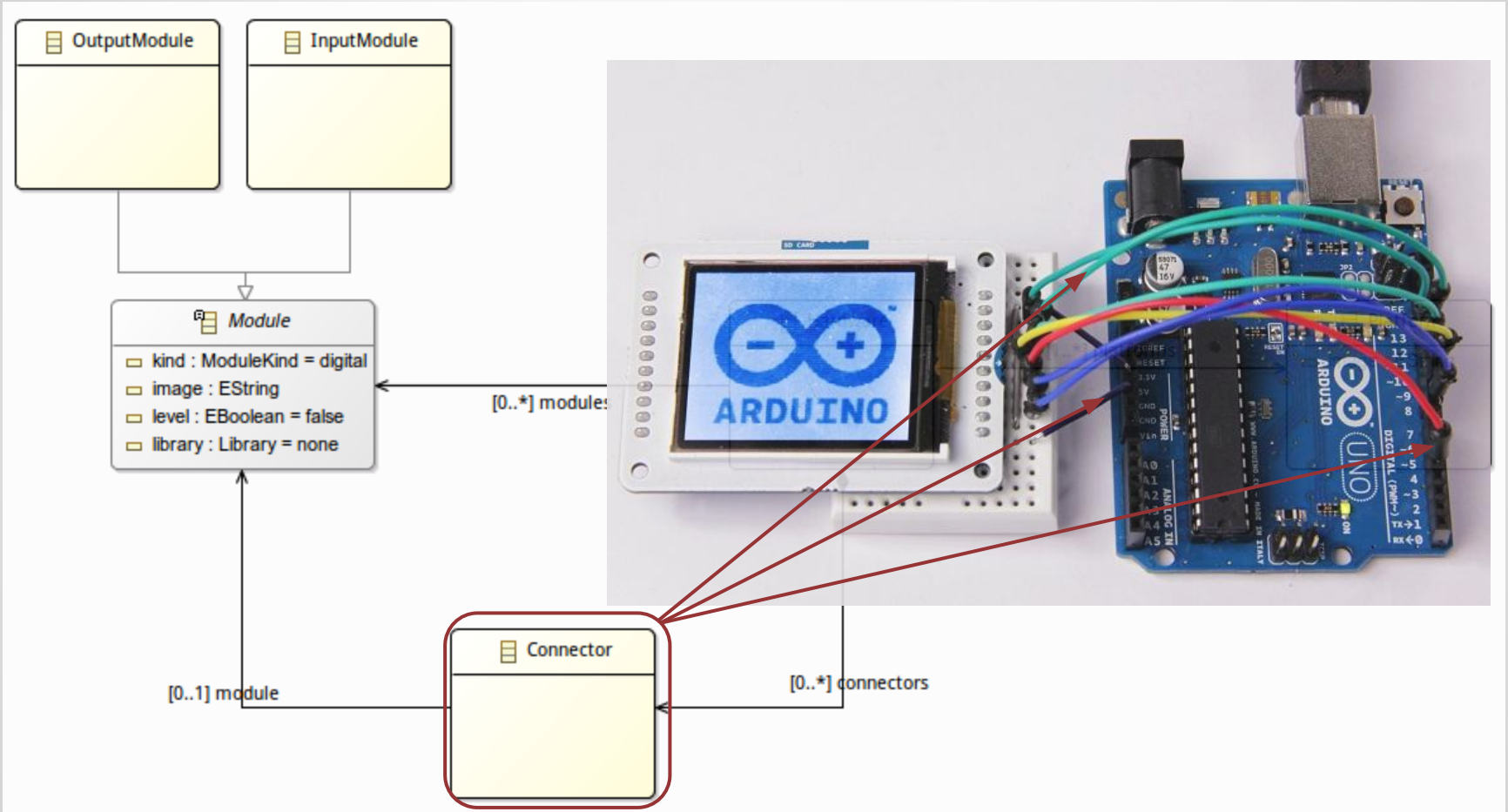

OutputModule
InputModule
Module
kind : ModuleKind = digital
image : EString
level : EBoolean = false
library : Library = none
[0..*] modules
dware
Platform
image : EString
Connector
[0..1] module
[0..*] connectors

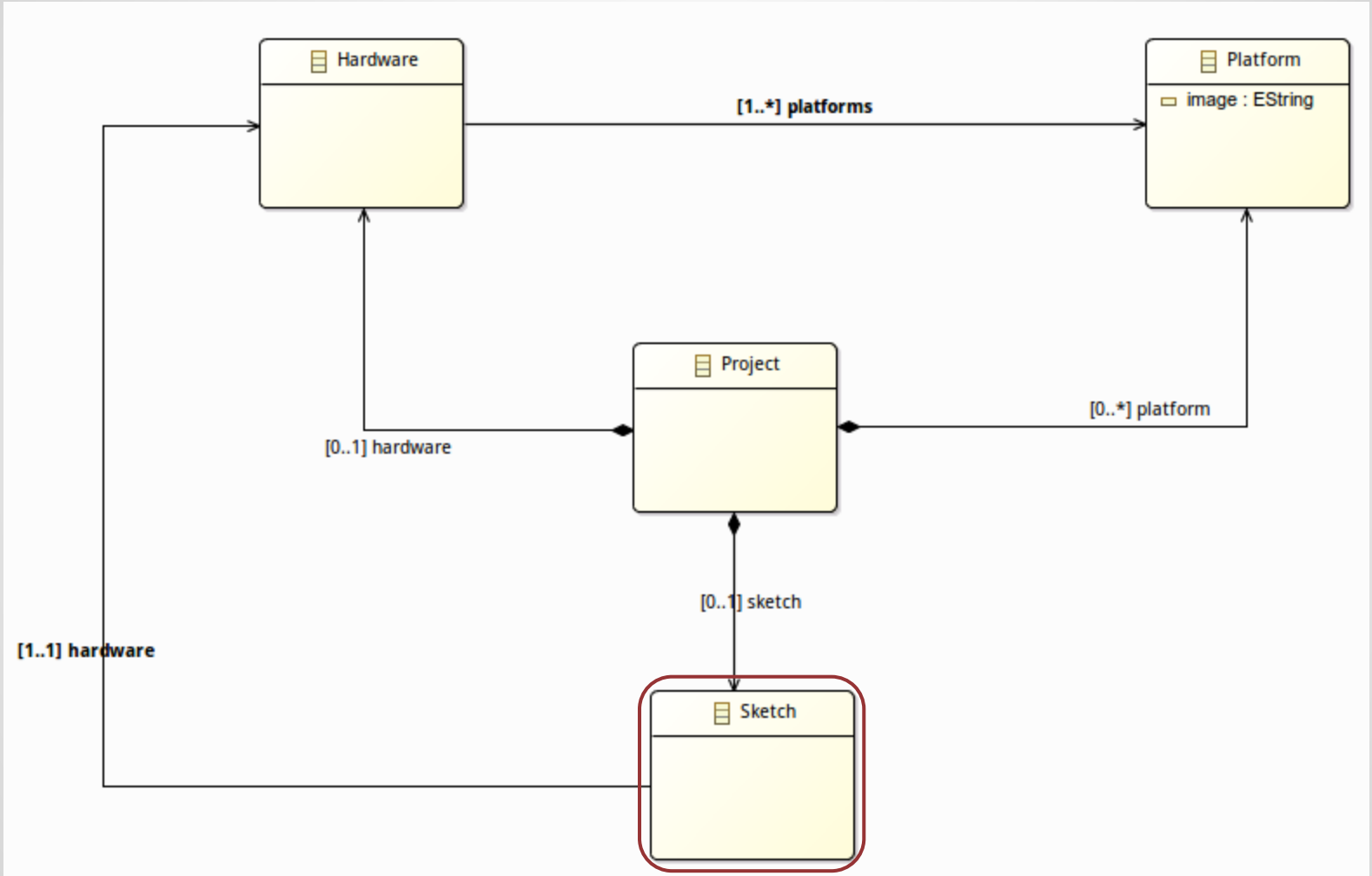

OutputModule
InputModule
Module
kind : ModuleKind = digital
image : EString
level : EBoolean = false
library : Library = none
[0..*] modules
Platform
image : EString
Connector
[0..1] module
[0..*] connectors

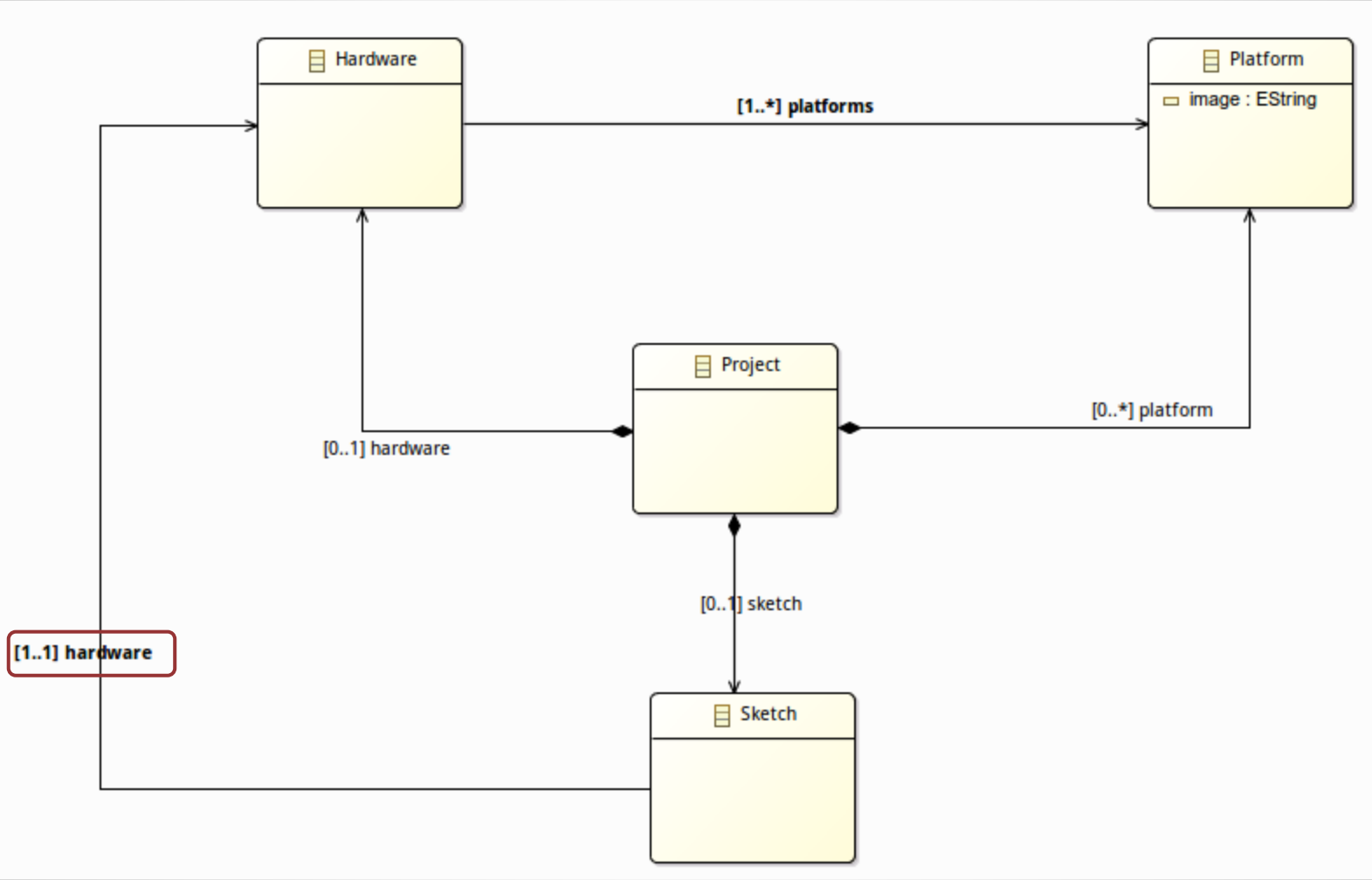

OutputModule
InputModule
Module
kind : ModuleKind = digital
image : EString
level : EBoolean = false
library : Library = none
Hardware
[1..*] platforms
Platform
image : EString
Connector
[0..1] module
[0..*] connectors

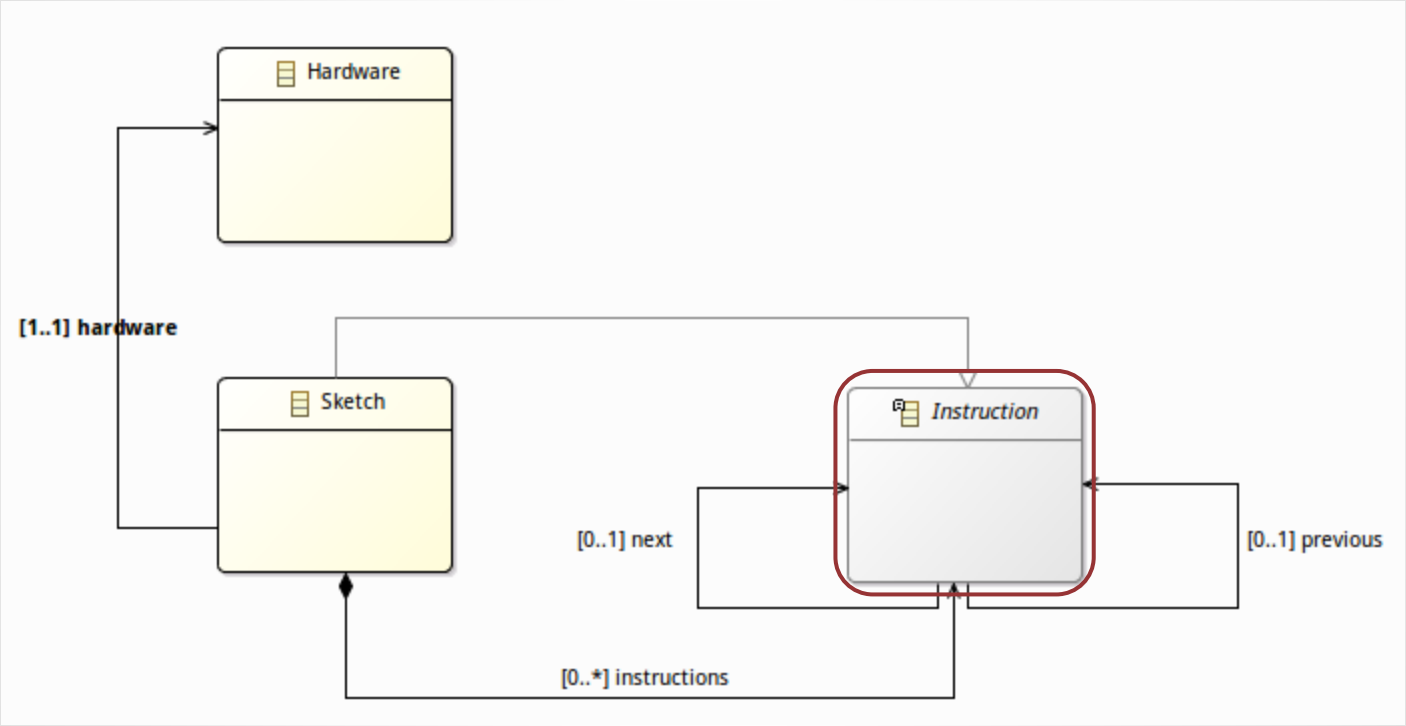

OutputModule
InputModule
Module
kind : ModuleKind = digital
image : EString
level : EBoolean = false
library : Library = none
Hardware
[1..*] platforms
Platform
image : EString
ON
OFF
Connector
[0..1] module
[0..*] connectors

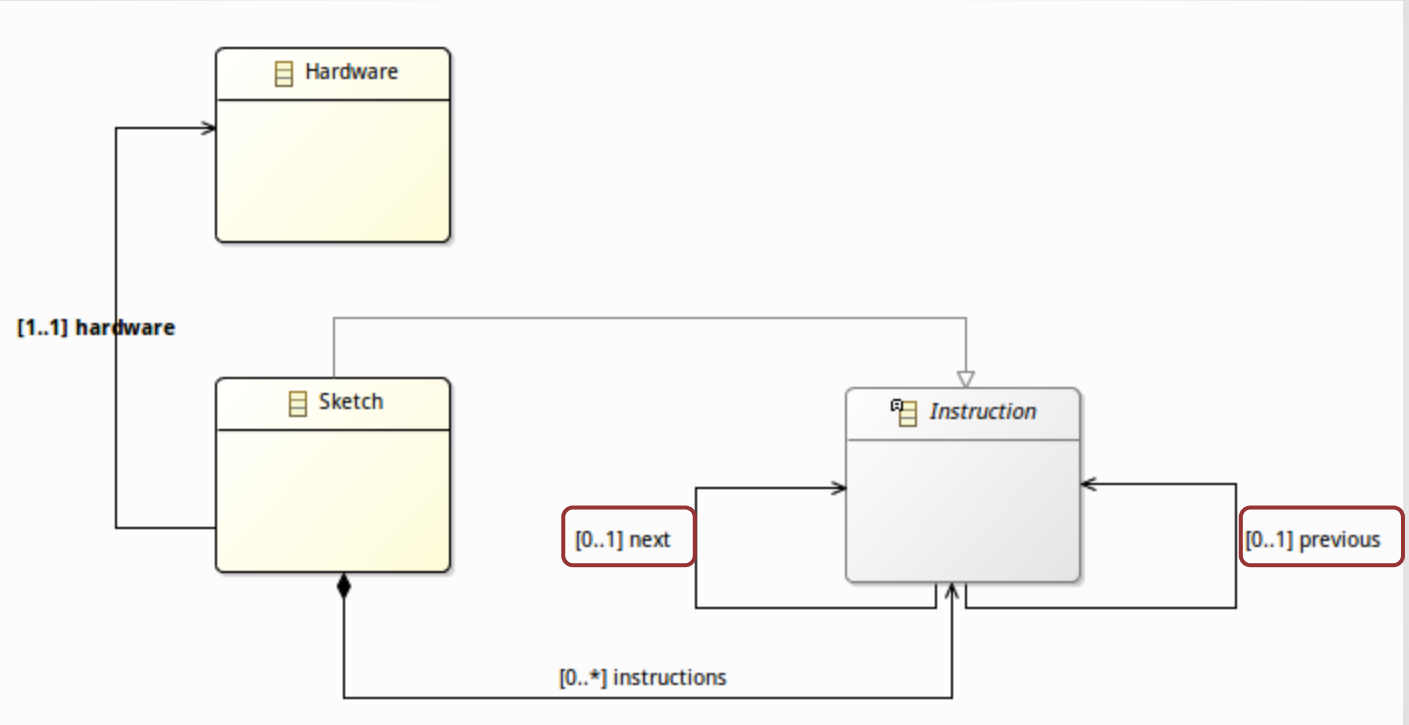

OutputModule
InputModule
Module
kind : ModuleKind = digital
image : EString
level : EBoolean = false
library : Library = none
Hardware
[1..*] platforms
Platform
image : EString
servo.h
Connector
[0..1] module
[0..*
#include <Servo.h>
Servo Microservo1;
int RotationsensorV1 = 5;
void setup() {
Microservo1.attach(9);
}
// the loop routine runs over and over again forever:
void loop() {
Microservo1.write(analogRead(RotationsensorV1));
}

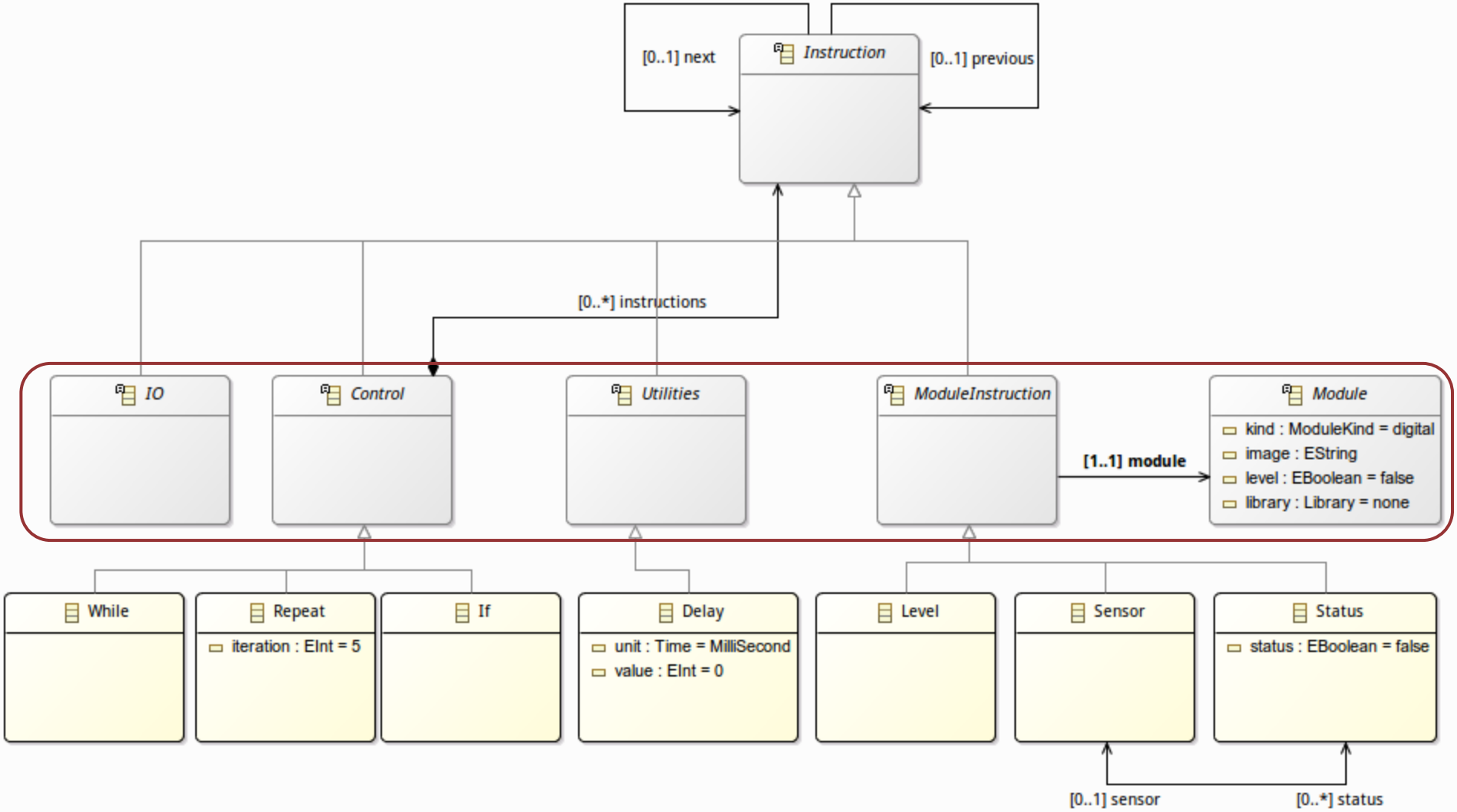

OutputModule
InputModule
Module
kind : ModuleKind = digital
image : EString
level : EBoolean = false
library : Library = none
[0..*] modules
Connector
[0..1] module
[0..*] connectors
ARDUINO
UNO

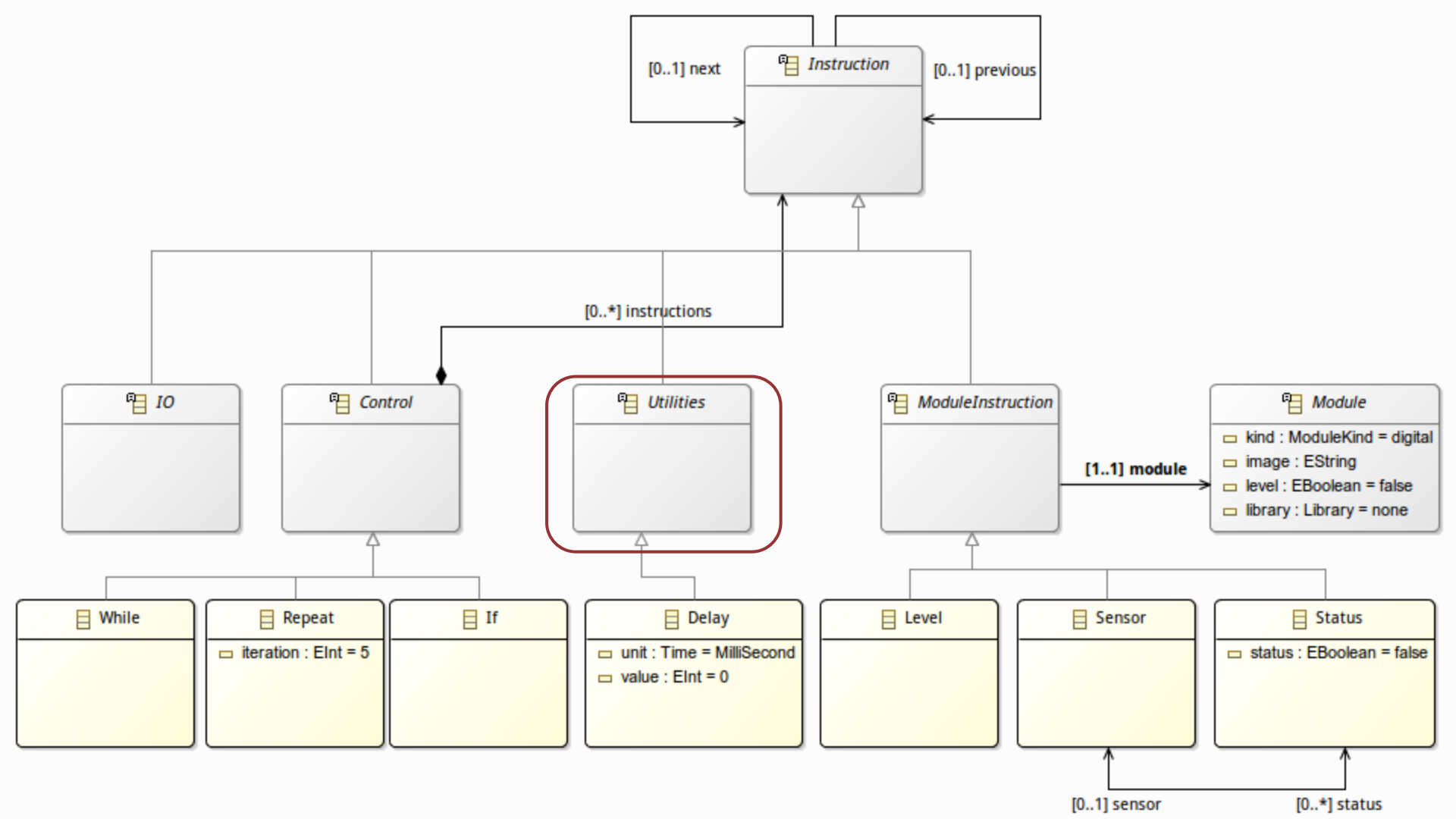

Hardware
Platform
image : EString
[1..*] platforms
Project
[0..1] hardware
[0..*] platform
[0..1] sketch
[1..1] hardware
Sketch


Hardware
Platform
image : EString
[1..*] platforms
Project
[0..1] hardware
[0..*] platform
[0..1] sketch
[1..1] hardware
Sketch


Hardware
[1..1] hardware
Sketch
Instruction
[0..1] next
[0..1] previous
[0..*] instructions

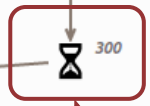

Hardware
[1..1] hardware
Sketch
Instruction
[0..1] next
[0..1] previous
[0..*] instructions

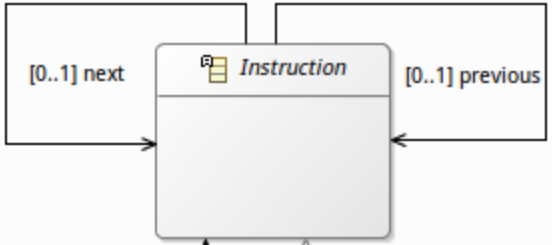

[0..1] next
Instruction
[0..1] previous
[0..*] instructions
IO
Control
Utilities
ModuleInstruction
[1..1] module
Module
kind : ModuleKind = digital
image : EString
level : EBoolean = false
library : Library = none
While
Repeat
iteration : EInt = 5
If
Delay
unit : Time = MilliSecond
value : EInt = 0
Level
Sensor
Status
status : EBoolean = false
[0..1] sensor
[0..*] status


[0..1] next
Instruction
[0..1] previous
[0..*] instructions
IO
Control
Utilities
ModuleInstruction
[1..1] module
Module
kind : ModuleKind = digital
image : EString
level : EBoolean = false
library : Library = none
While
Repeat
iteration : EInt = 5
If
Delay
unit : Time = MilliSecond
value : EInt = 0
Level
Sensor
Status
status : EBoolean = false
[0..1] sensor
[0..*] status

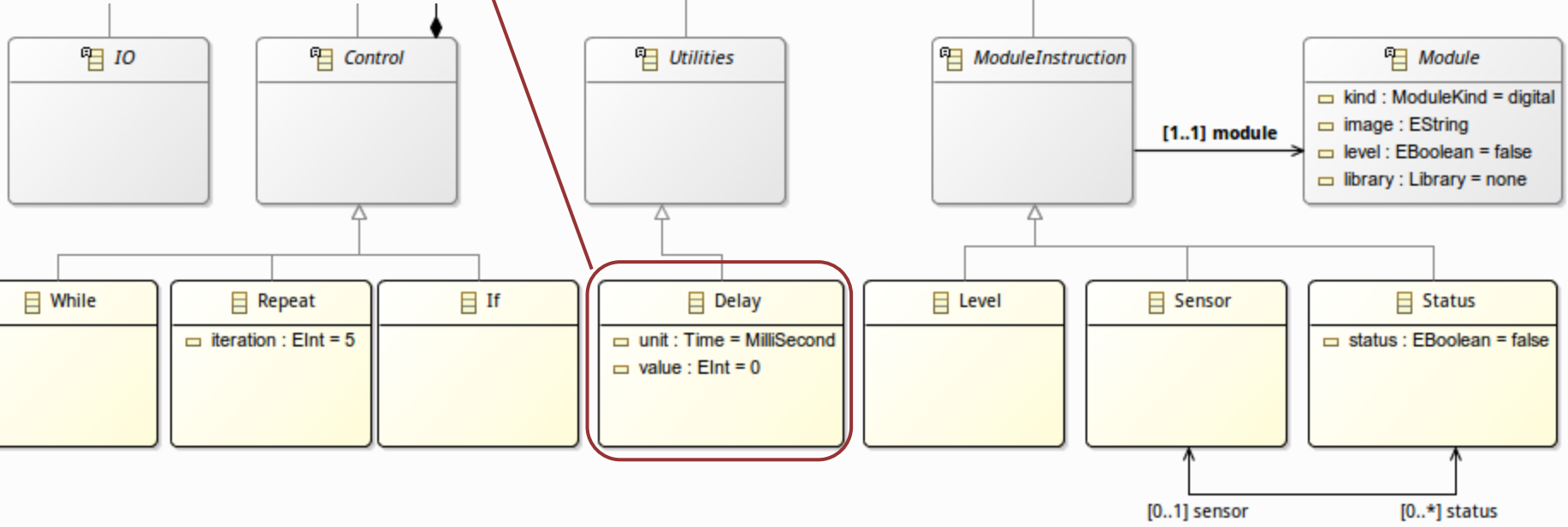

Blue LED : on
300

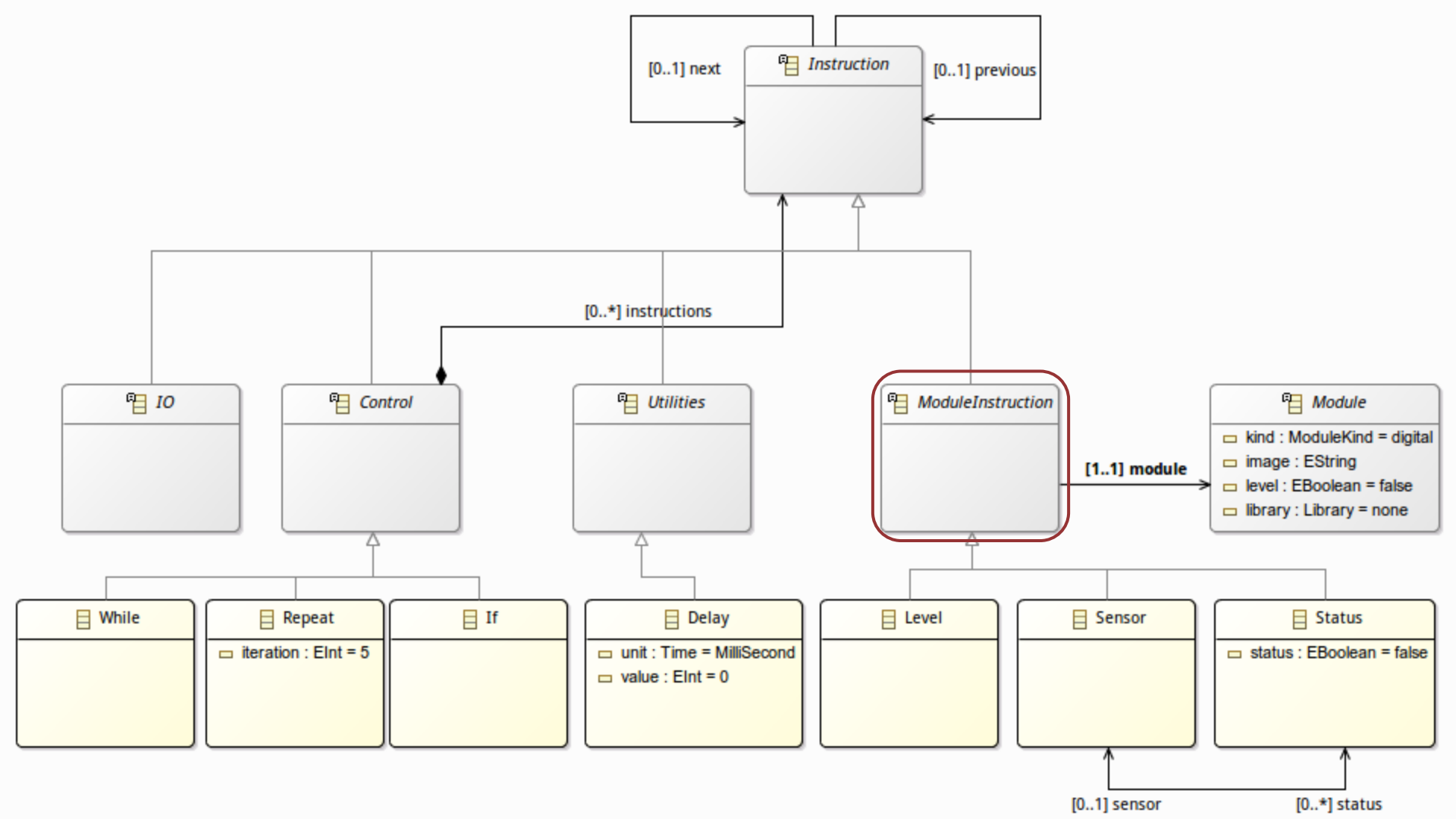

Blue LED : off

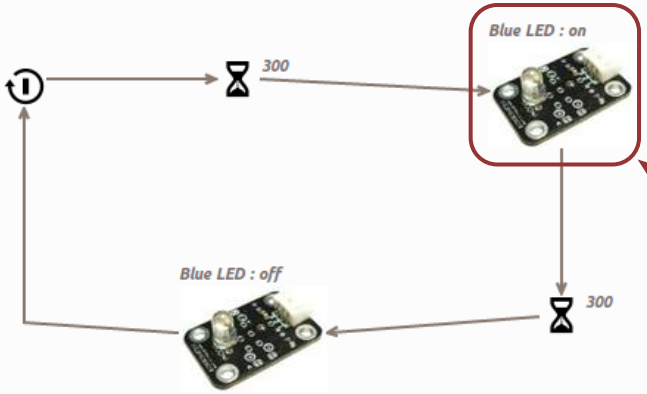

300


[0..1] next
Instruction
[0..1] previous


[0..*] instructions


IO
Control
Utilities
ModuleInstruction
[1..1] module
Module
kind : ModuleKind = digital
image : EString
level : EBoolean = false
library : Library = none
While
Repeat
iteration : EInt = 5
If
Delay
unit : Time = MilliSecond
value : EInt = 0
Level
Sensor
Status
status : EBoolean = false
[0..1] sensor
[0..*] status


[0..1] next
Instruction
[0..1] previous
[0..*] instructions
IO
Control
Utilities
ModuleInstruction
[1..1] module
Module
kind : ModuleKind = digital
image : EString
level : EBoolean = false
library : Library = none
While
Repeat
iteration : EInt = 5
If
Delay
unit : Time = MilliSecond
value : EInt = 0
Level
Sensor
Status
status : EBoolean = false
[0..1] sensor
[0..*] status



[0..1] next

Instruction

[0..1] previous

[0..*] instructions

IO

Control

Utilities

ModuleInstruction

[1..1] module

Module
kind : ModuleKind = digital
image : EString
level : EBoolean = false
library : Library = none

While

Repeat
iteration : EInt = 5

If

Delay
unit : Time = MilliSecond
value : EInt = 0

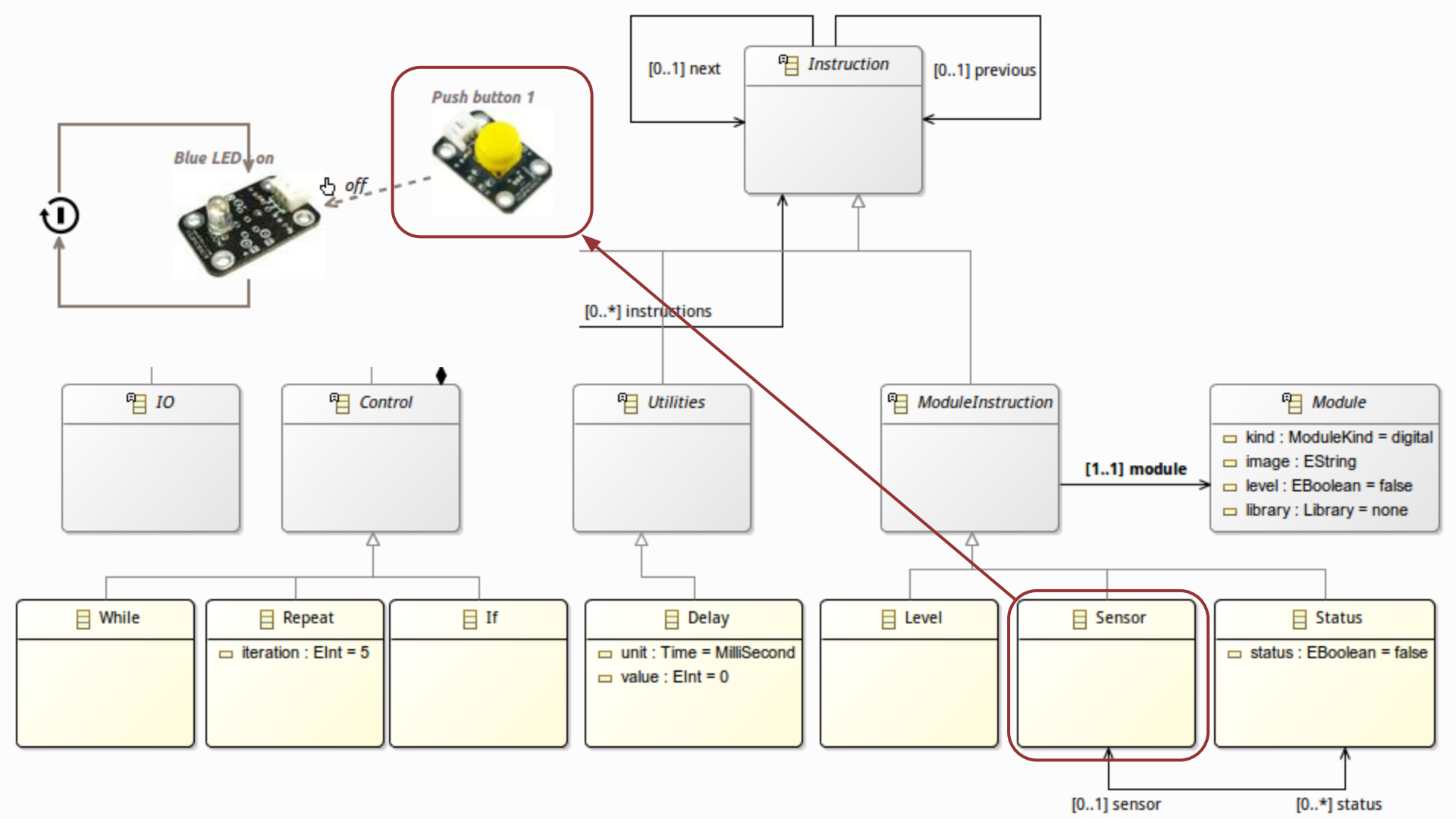
Level

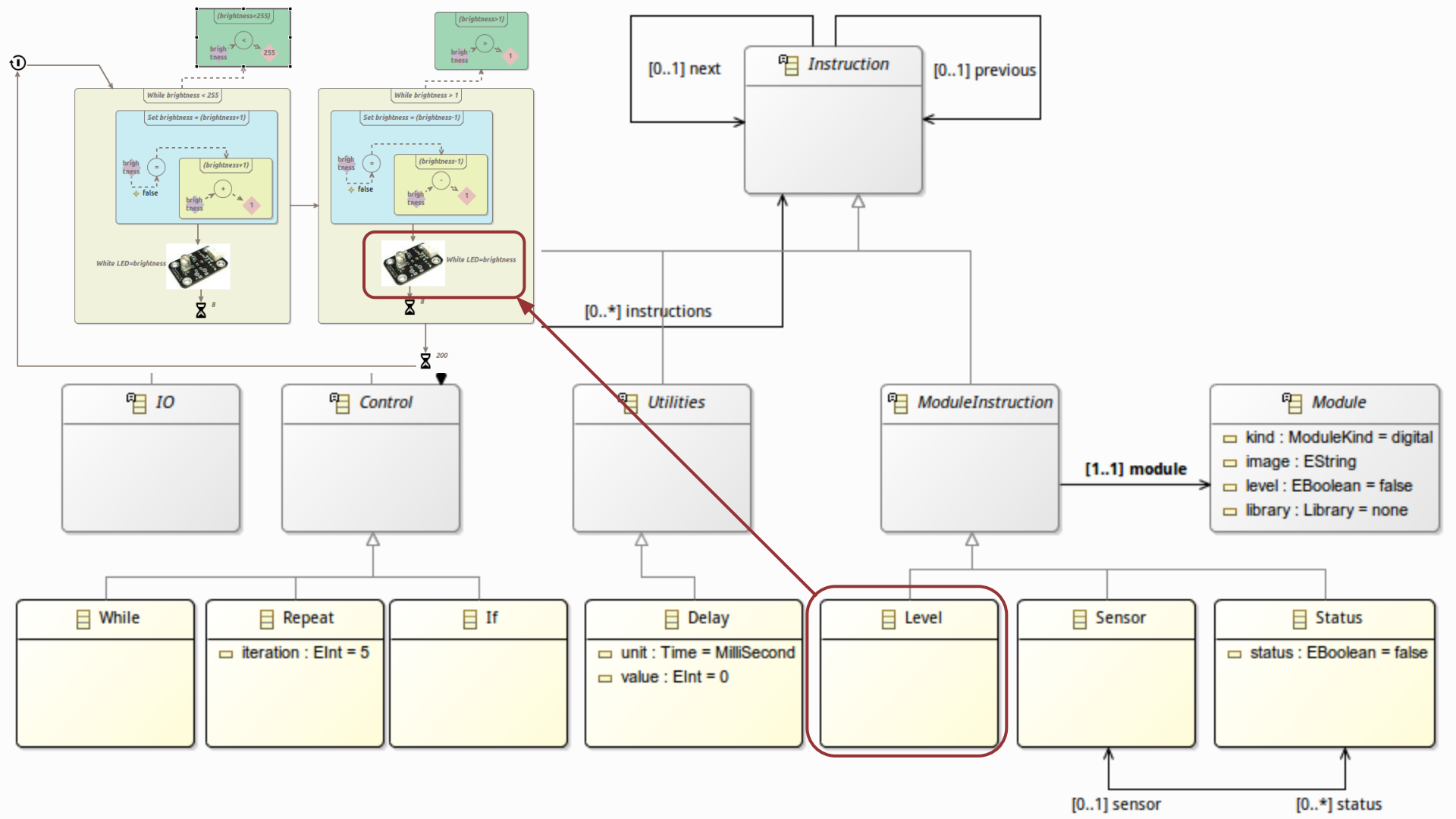
Sensor

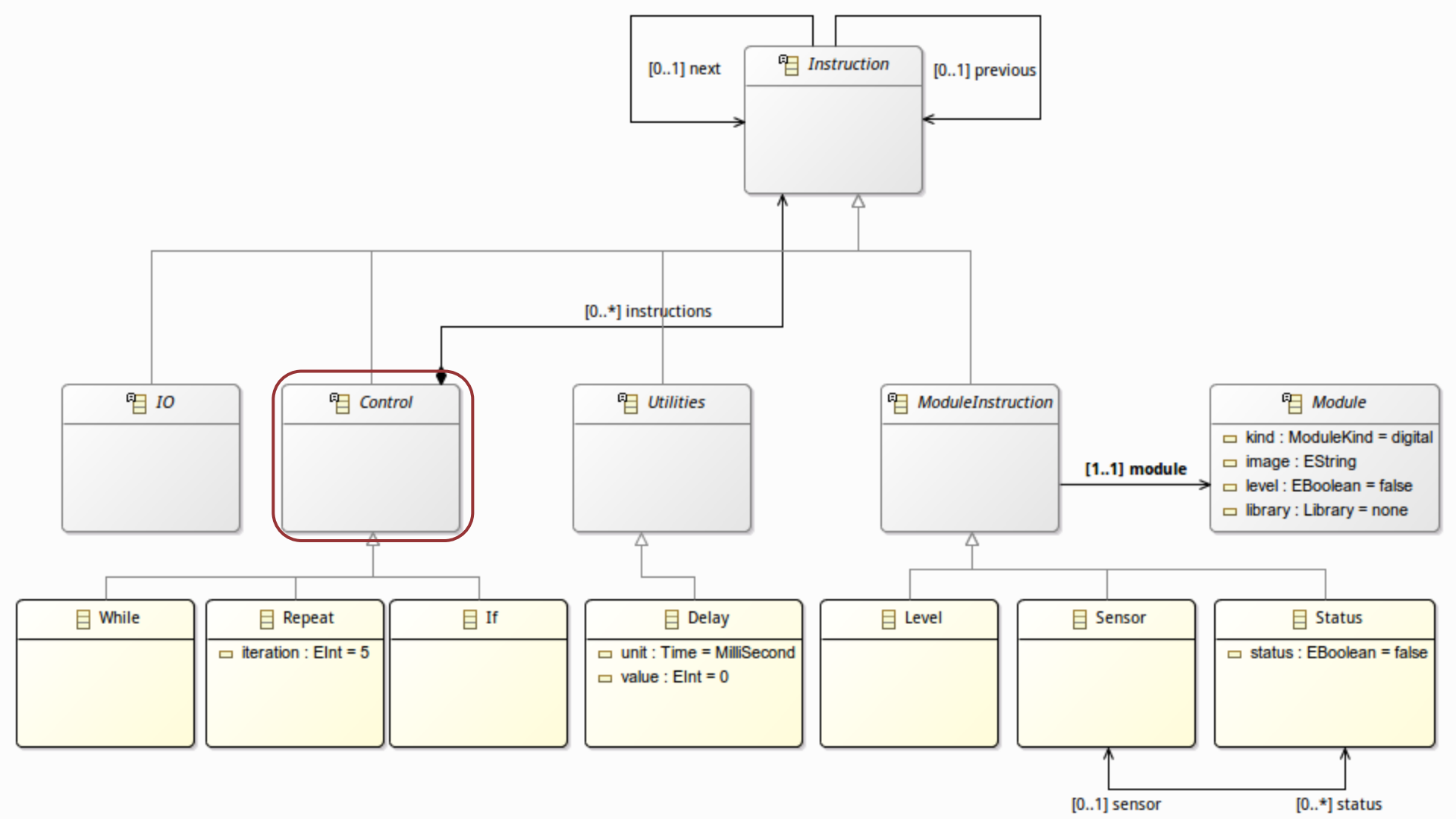
Status
status : EBoolean = false

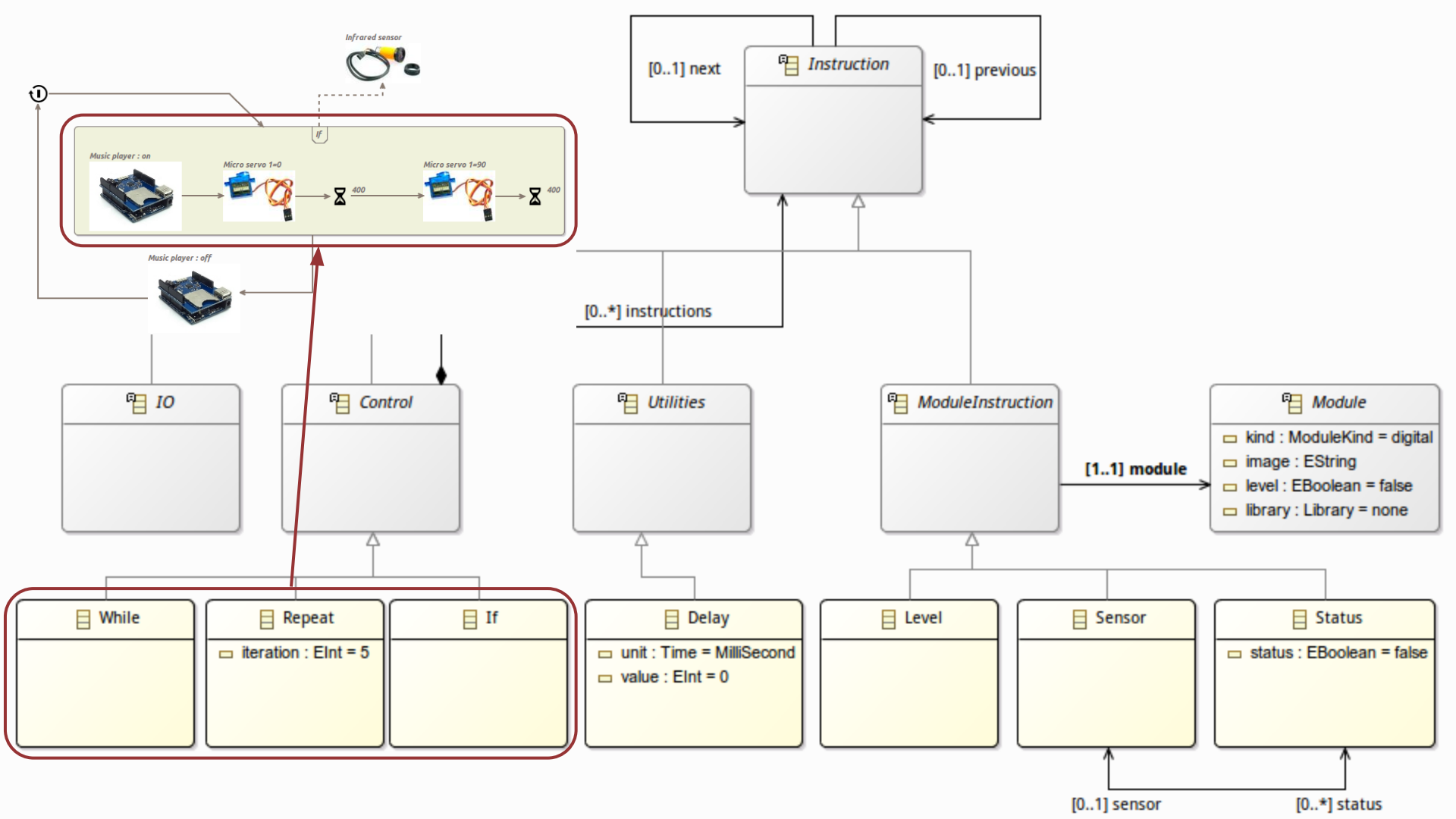
[0..1] sensor

[0..*] status

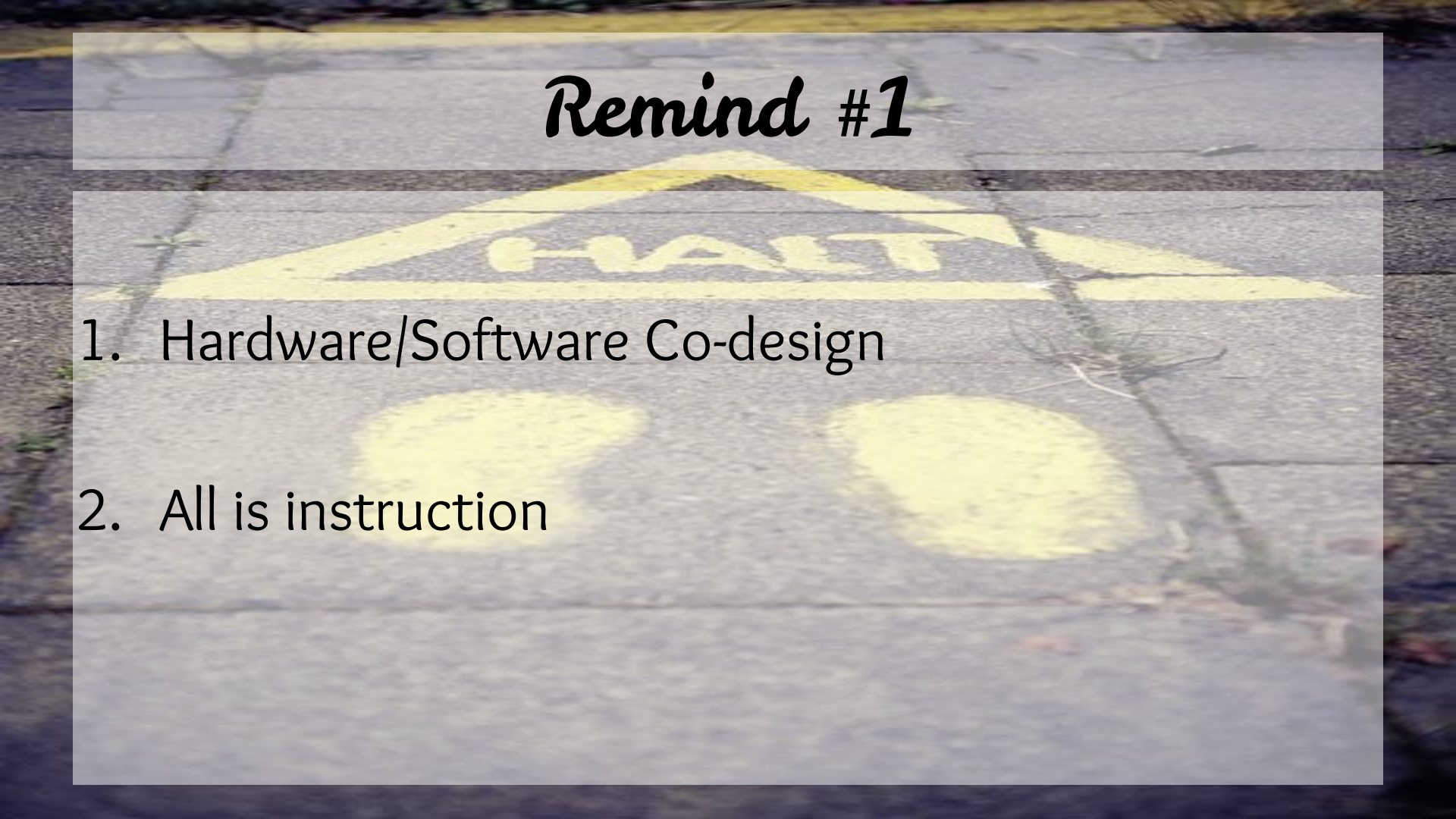

Push button 1
Blue LED on
off
[0..1] next
Instruction
[0..1] previous
[0..*] instructions
IO
Control
Utilities
ModuleInstruction
[1..1] module
Module
kind : ModuleKind = digital
image : EString
level : EBoolean = false
library : Library = none
While
Repeat
iteration : EInt = 5
If
Delay
unit : Time = MilliSecond
value : EInt = 0
Level
Sensor
Status
status : EBoolean = false
[0..1] sensor
[0..*] status


(brightness<255)
brigh tness
255
(brightness>1)
1
While brightness < 255
Set brightness = (brightness+1)
(brightness+1)
false
White LED=brightness
8
While brightness > 1
Set brightness = (brightness-1)
(brightness-1)
200
[0..1] next
Instruction
[0..1] previous
[0..*] instructions
IO
Control
Utilities
ModuleInstruction
[1..1] module
Module
kind : ModuleKind = digital
image : EString
level : EBoolean = false
library : Library = none
While
Repeat
iteration : EInt = 5
If
Delay
unit : Time = MilliSecond
value : EInt = 0
Level
Sensor
Status
status : EBoolean = false
[0..1] sensor
[0..*] status


[0..1] next
Instruction
[0..1] previous
[0..*] instructions
IO
Control
Utilities
ModuleInstruction
[1..1] module
Module
kind : ModuleKind = digital
image : EString
level : EBoolean = false
library : Library = none
While
Repeat
iteration : EInt = 5
If
Delay
unit : Time = MilliSecond
value : EInt = 0
Level
Sensor
Status
status : EBoolean = false
[0..1] sensor
[0..*] status


Infrared sensor
If
Music player : on
Micro servo 1=0
400
Micro servo 1=90
400
Music player : off
[0..1] next
Instruction
[0..1] previous
[0..*] instructions
IO
Control
Utilities
ModuleInstruction
[1..1] module
Module
kind : ModuleKind = digital
image : EString
level : EBoolean = false
library : Library = none
While
Repeat
iteration : EInt = 5
If
Delay
unit : Time = MilliSecond
value : EInt = 0
Level
Sensor
Status
status : EBoolean = false
[0..1] sensor
[0..*] status

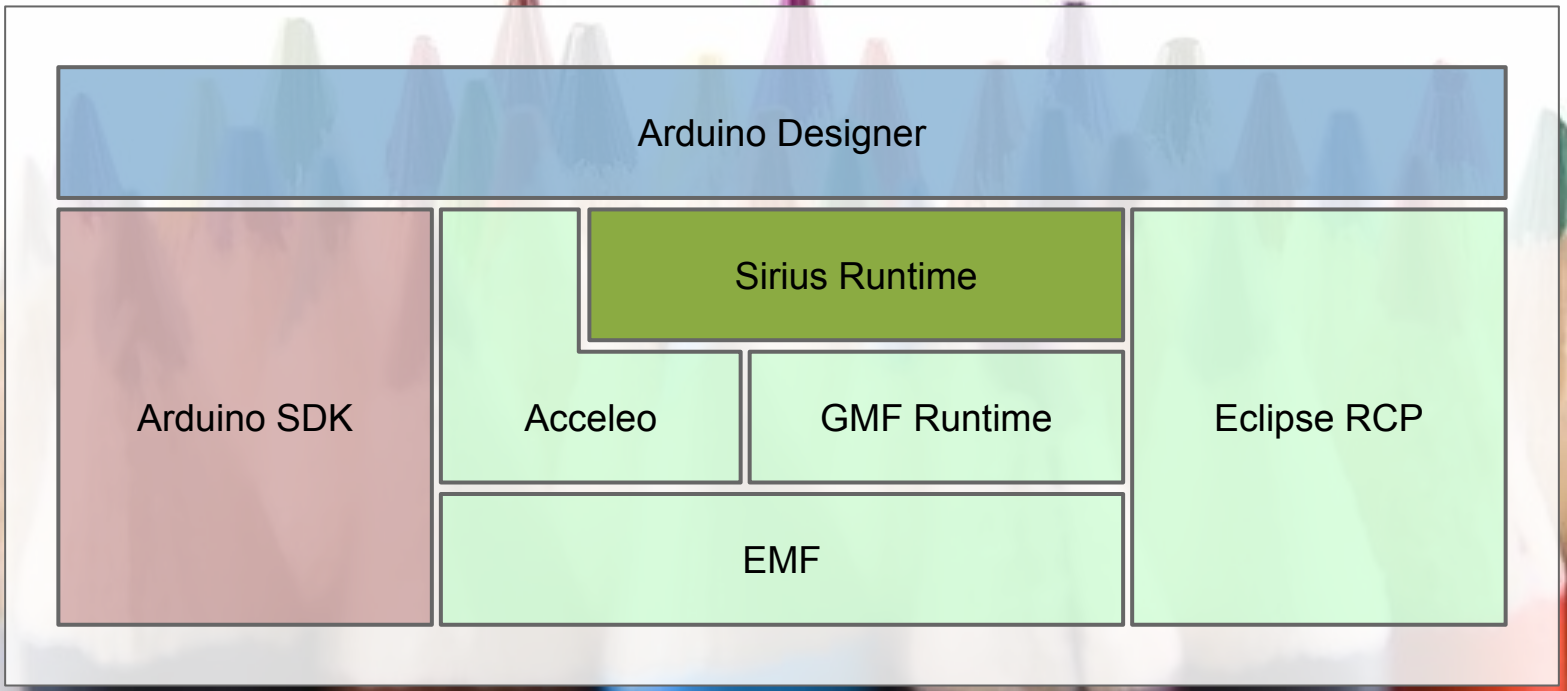
Remind #1



1. Hardware/Software Co-design

2. All is instruction

Create graphical editor



Sirius

A tool to quickly define DSL based on custom multi-view workbenches with dedicated representations

http://eclipse.org/sirius

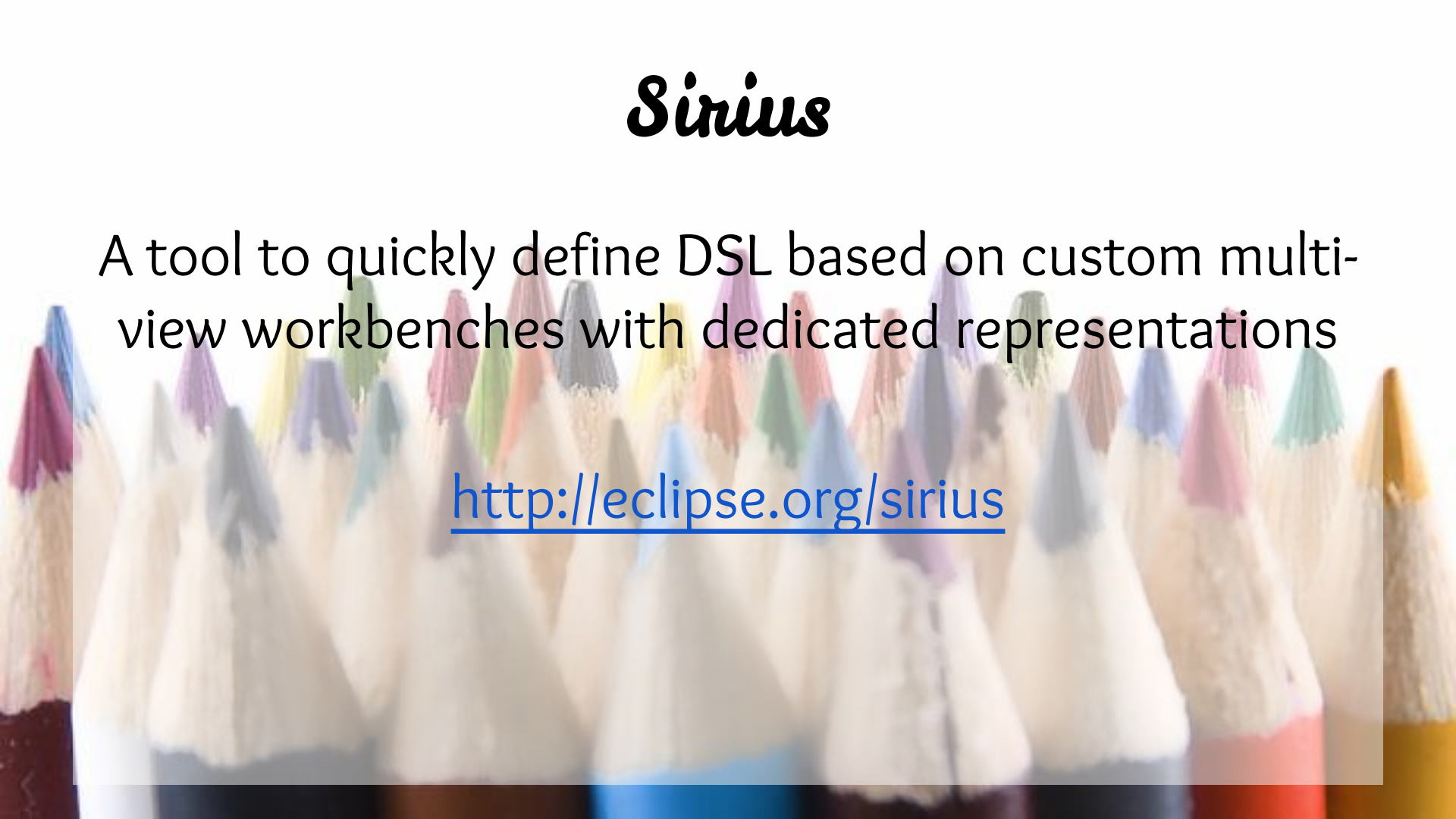
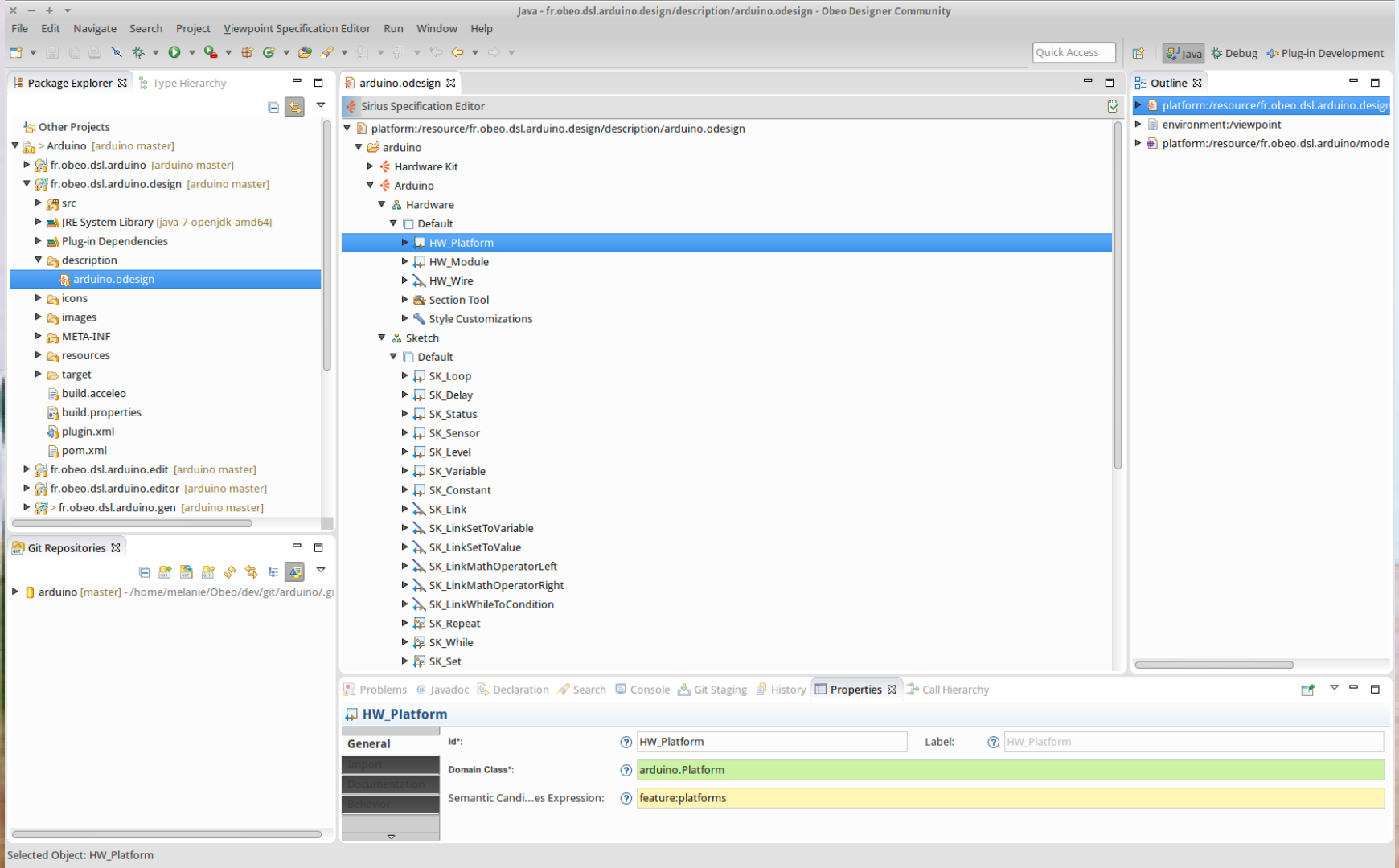


Diagram definition

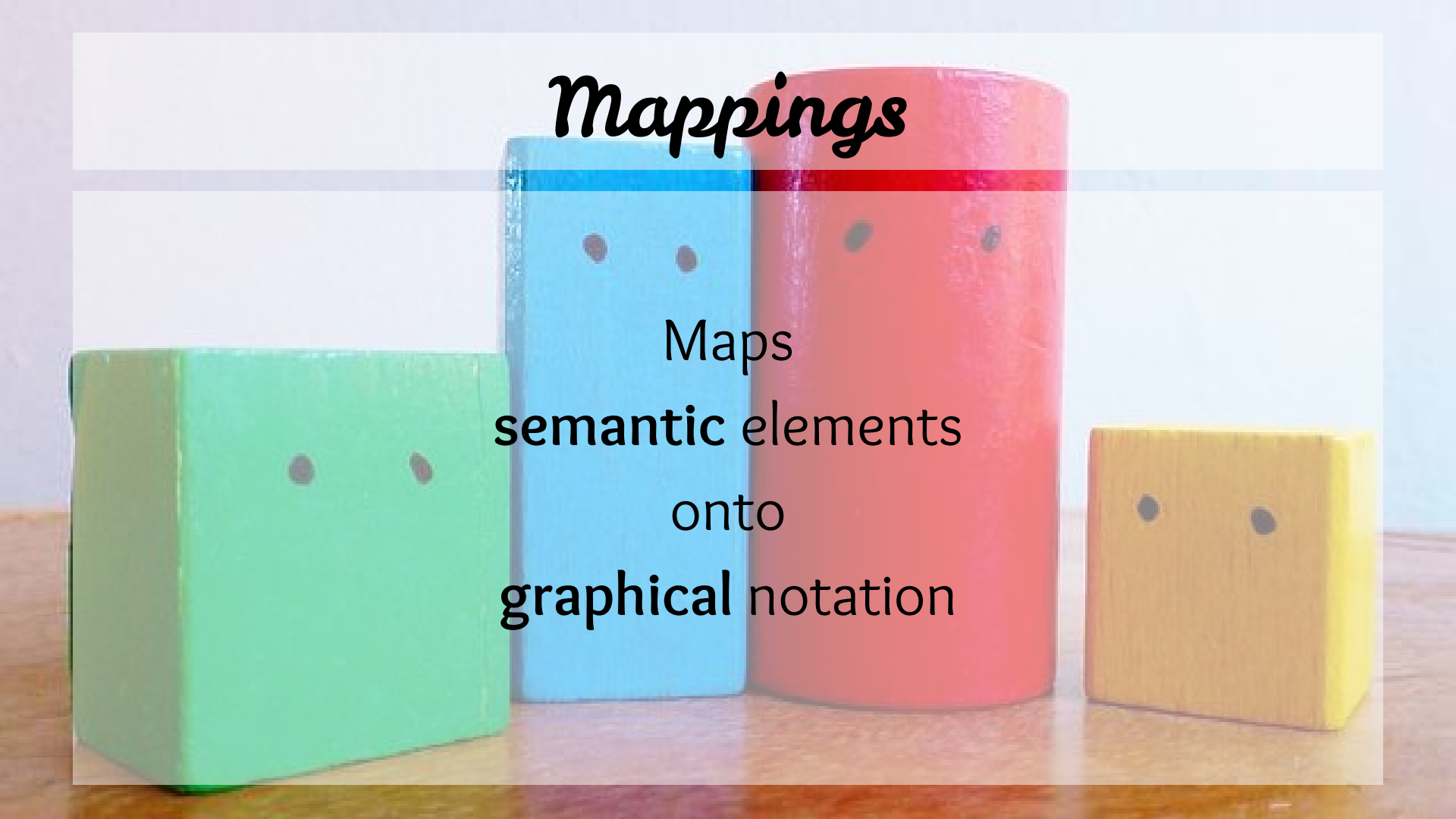
Create a **Sirius Specification Project** and provide **diagram descriptions :**

- Mappings
- Styles
- Tools

Interpreted @ runtime


Java - fr.obeo.dsl.arduino.design/description/arduino.odesign - Obeo Designer Community
File Edit Navigate Search Project Viewpoint Specification Editor Run Window Help
Quick Access
Java Debug Plug-in Development
Package Explorer
Type Hierarchy
Other Projects
Arduino [arduino master]
fr.obeo.dsl.arduino [arduino master]
fr.obeo.dsl.arduino.design [arduino master]
src
JRE System Library [java-7-openjdk-amd64]
Plug-in Dependencies
description
arduino.odesign
icons
images
META-INF
resources
target
build.acceleo
build.properties
plugin.xml
pom.xml
fr.obeo.dsl.arduino.edit [arduino master]
fr.obeo.dsl.arduino.editor [arduino master]
fr.obeo.dsl.arduino.gen [arduino master]
Git Repositories
arduino [master] - /home/melanie/Obeo/dev/git/arduino/.gi
arduino.odesign
Sirius Specification Editor
platform:/resource/fr.obeo.dsl.arduino.design/description/arduino.odesign
arduino
Hardware Kit
Arduino
Hardware
Default
HW_Platform
HW_Module
HW_Wire
Section Tool
Style Customizations
Sketch
Default
SK_Loop
SK_Delay
SK_Status
SK_Sensor
SK_Level
SK_Variable
SK_Constant
SK_Link
SK_LinkSetToVariable
SK_LinkSetToValue
SK_LinkMathOperatorLeft
SK_LinkMathOperatorRight
SK_LinkWhileToCondition
SK_Repeat
SK_While
SK_Set
Outline
platform:/resource/fr.obeo.dsl.arduino.design
environment:/viewpoint
platform:/resource/fr.obeo.dsl.arduino/mode
Problems Javadoc Declaration Search Console Git Staging History Properties Call Hierarchy
HW_Platform
General
Id*:
HW_Platform
Label:
HW_Platform
Domain Class*:
arduino.Platform
Semantic Candi...es Expression:
feature:platforms
Selected Object: HW_Platform

Mappings



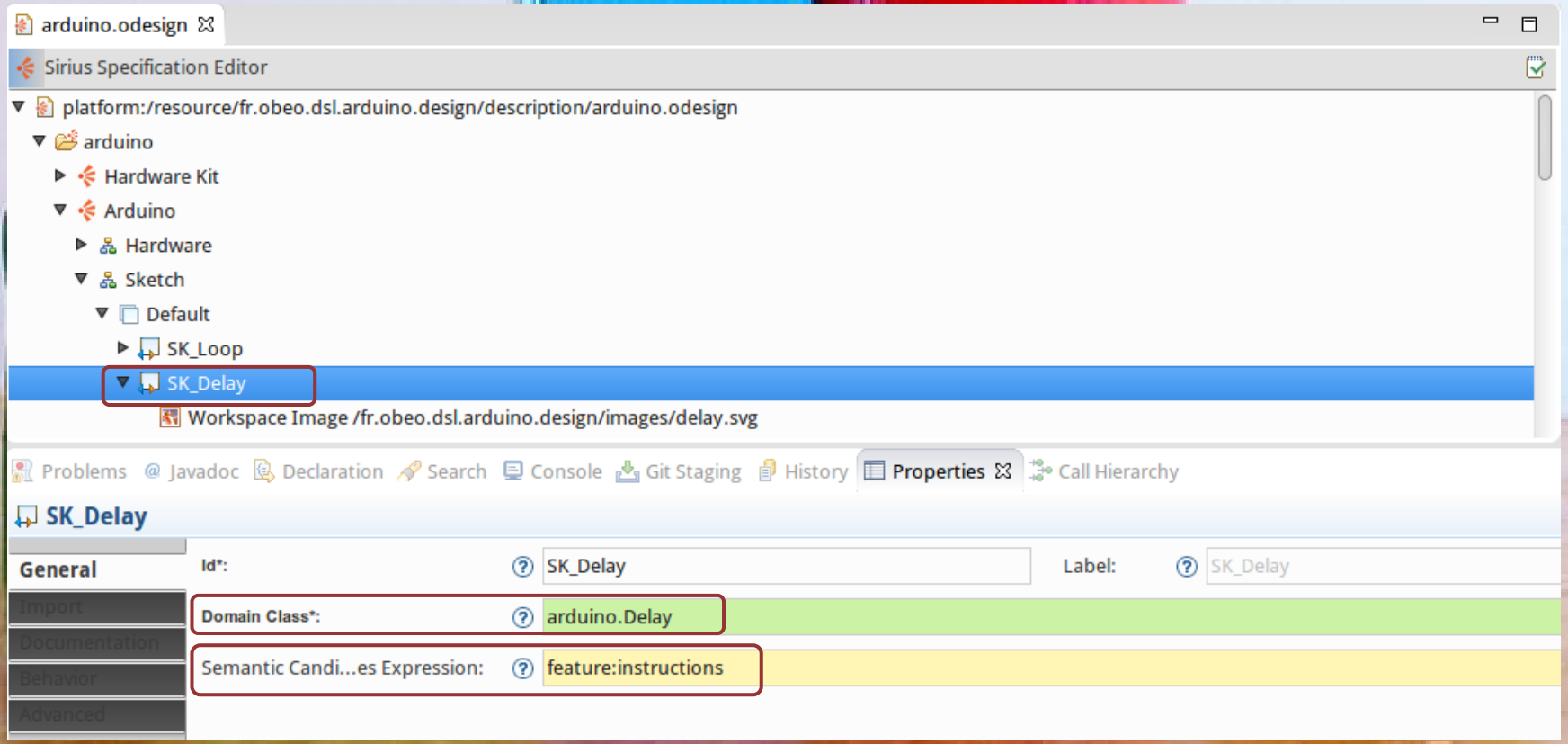
Maps

semantic elements

onto

graphical notation

Mappings

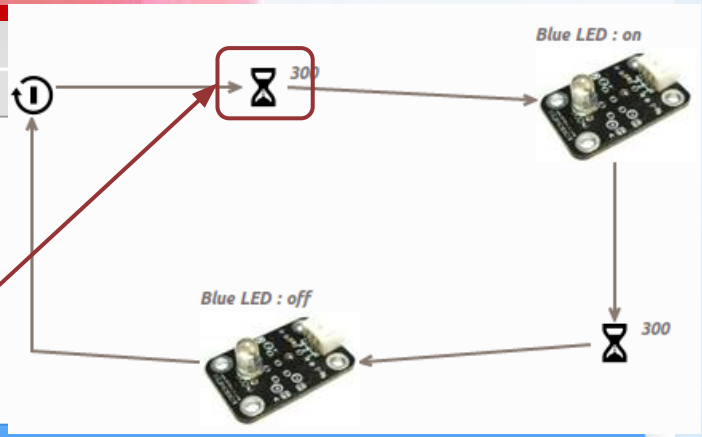

arduino.odesign
Sirius Specification Editor
platform:/resource/fr.obeo.dsl.arduino.design/description/arduino.odesign
arduino
Hardware Kit
Arduino
Hardware
Sketch
Default
SK_Loop
SK_Delay
Workspace Image /fr.obeo.dsl.arduino.design/images/delay.svg
Problems
Javadoc
Declaration
Search
Console
Git Staging
History
Properties
Call Hierarchy
SK_Delay
General
Import
Documentation
Behavior
Advanced
Id*:
SK_Delay
Label:
SK_Delay
Domain Class*:
arduino.Delay
Semantic Candi...es Expression:
feature:instructions

Mappings

arduino.odesign

Sirius Specification Editor

- platform:/resource/fr.obeo.dsl.arduino.design/description/arduino.odesign
  - arduino
    - Hardware Kit
    - Arduino
      - Hardware
      - Sketch
        - Default
          - SK_Loop
          - SK_Delay
            - Workspace Image /fr.obeo.dsl.arduino.design/images/delay.svg



Problems | Javadoc | Declaration | Search | Console | Git Staging | History | Properties | Call Hierarchy

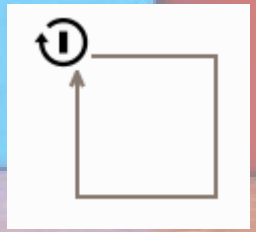
SK_Delay

General | Import | Documentation | Behavior | Advanced

Id*:	SK_Delay	Label:	SK_Delay
Domain Class*:	arduino.Delay		
Semantic Candi...es Expression:	feature:instructions		

Mappings

How to represent
the sketch
loop ?



Mappings

arduino.odesign

Sirius Specification Editor

platform:/resource/fr.obeo.dsl.arduino.design/description/arduino.odesign

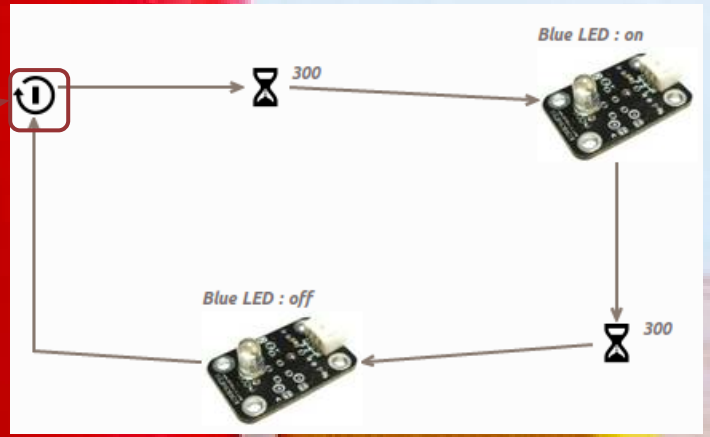
- arduino
  - Hardware Kit
  - Arduino
    - Hardware
    - Sketch
      - Default
        - SK_Loop
          - Workspace Image /fr.obeo.dsl.arduino.design/images/loop.svg

Properties

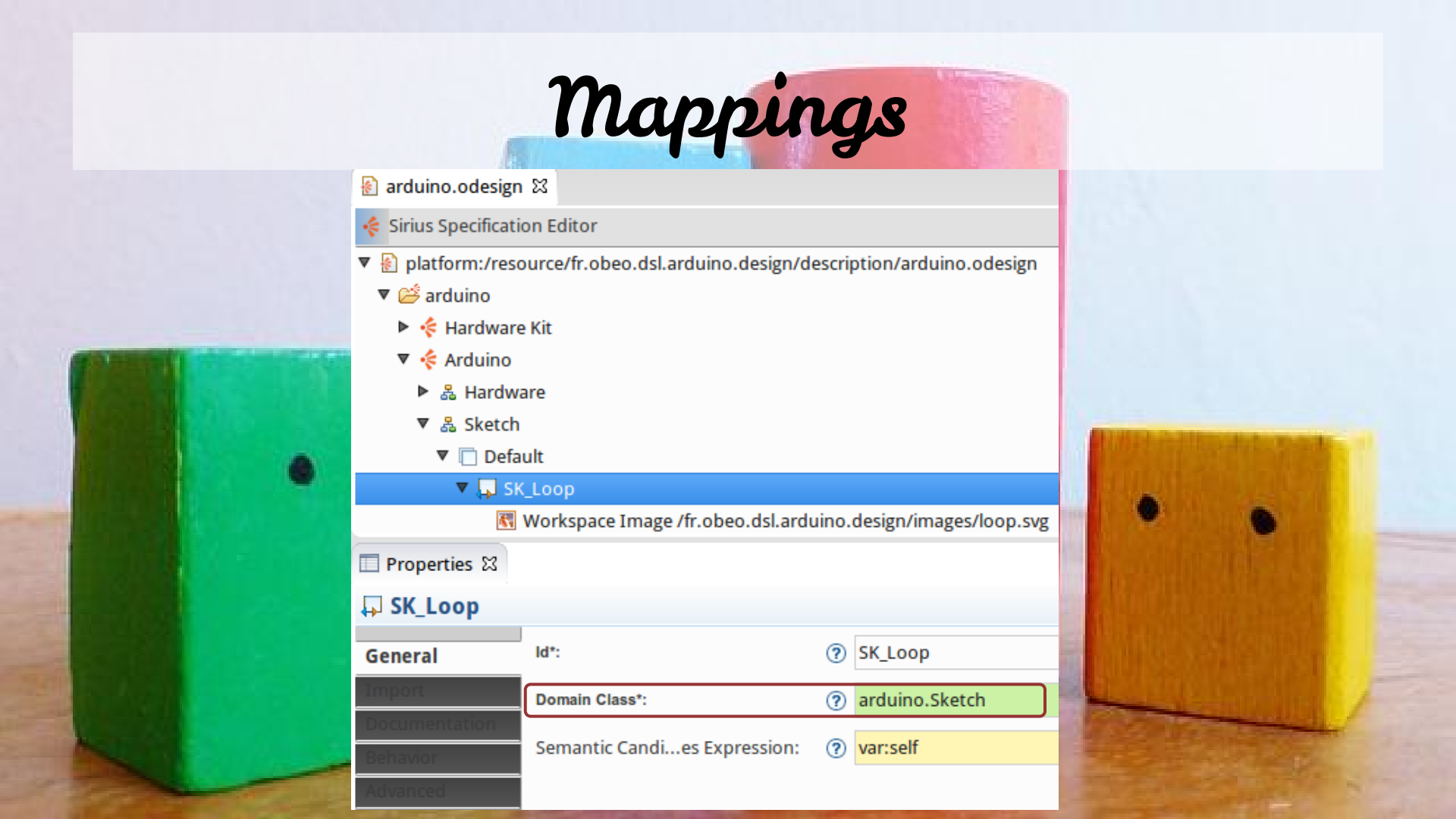
SK_Loop

General | Import | Documentation | Behavior | Advanced

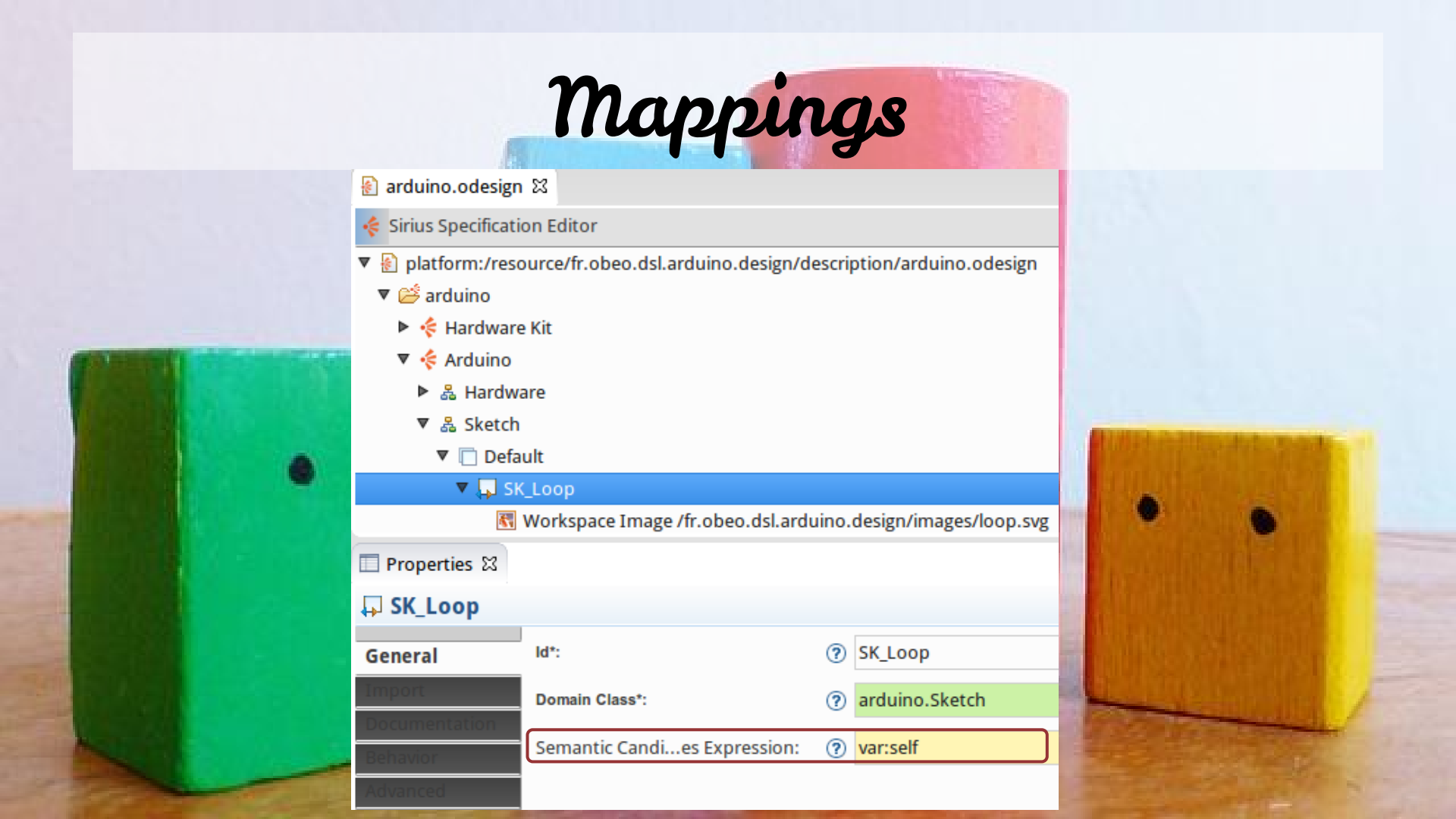
Id*:	SK_Loop
Domain Class*:	arduino.Sketch
Semantic Candi...es Expression:	var:self



Mappings


arduino.odesign
Sirius Specification Editor
platform:/resource/fr.obeo.dsl.arduino.design/description/arduino.odesign
arduino
Hardware Kit
Arduino
Hardware
Sketch
Default
SK_Loop
Workspace Image /fr.obeo.dsl.arduino.design/images/loop.svg
Properties
SK_Loop
General
Import
Documentation
Behavior
Advanced
Id*:
SK_Loop
Domain Class*:
arduino.Sketch
Semantic Candi...es Expression:
var:self

Mappings


arduino.odesign
Sirius Specification Editor
platform:/resource/fr.obeo.dsl.arduino.design/description/arduino.odesign
arduino
Hardware Kit
Arduino
Hardware
Sketch
Default
SK_Loop
Workspace Image /fr.obeo.dsl.arduino.design/images/loop.svg
Properties
SK_Loop
General
Import
Documentation
Behavior
Advanced
Id*:
SK_Loop
Domain Class*:
arduino.Sketch
Semantic Candi...es Expression:
var:self

Styles

Define

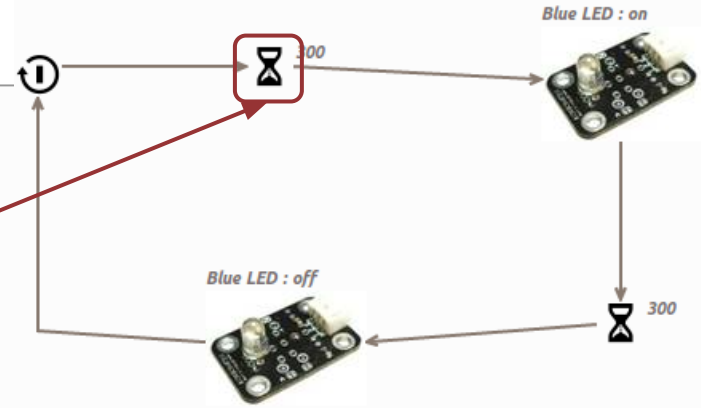
graphical appearance

arduino.odesign

Sirius Specification Editor

platform:/resource/fr.obeo.dsl.arduino.design/description/arduino.odesign

- arduino
  - Hardware Kit
  - Arduino
    - Hardware
    - Sketch
      - Default
        - SK_Loop
        - SK_Delay
          - Workspace Image /fr.obeo.dsl.arduino.design/images/delay.svg



Problems | Javadoc | Declaration | Search | Console | Git Staging | History | Properties | Call Hierarchy

Workspace Image /fr.obeo.dsl.arduino.design/images/delay.svg

General

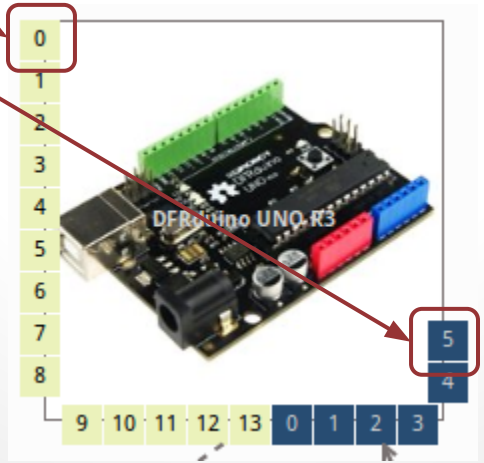
Label | Corner | Color | Border

Workspace Path*:	/fr.obeo.dsl.arduino.design/images/delay.svg
Tooltip Expression:	['Wait '+value+' '+unit/]

arduino.odesign

Sirius Specification Editor

- platform:/resource/fr.obeo.dsl.arduino.design/description/arduino.odesign
  - arduino
    - Hardware Kit
    - Arduino
      - Hardware
        - Default
          - HW_Platform
            - Bordered HS_DigitalPin
              - Square Arduino light green
            - Bordered HS_AnalogPin
              - Square dark_blue



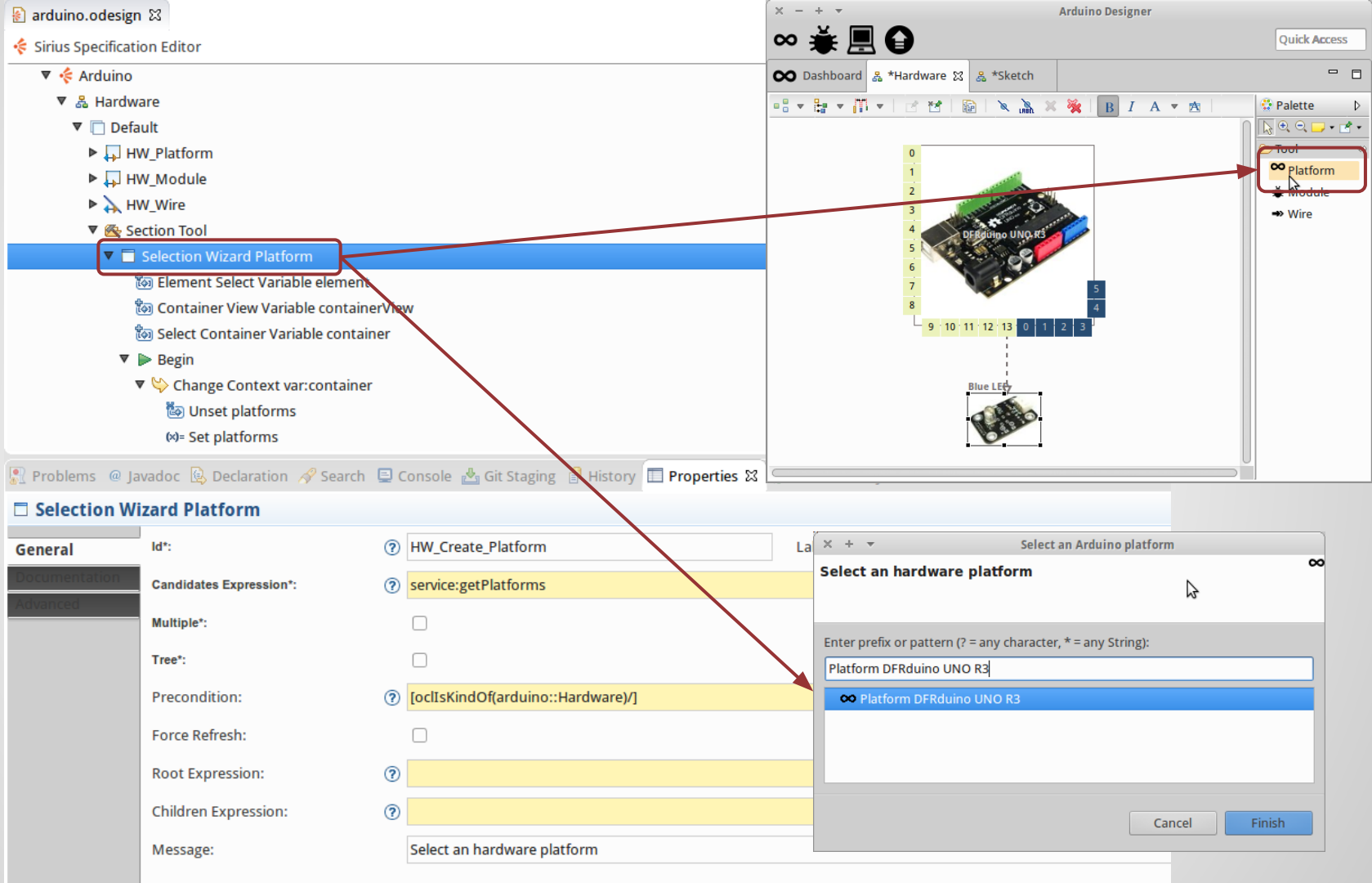
Tools

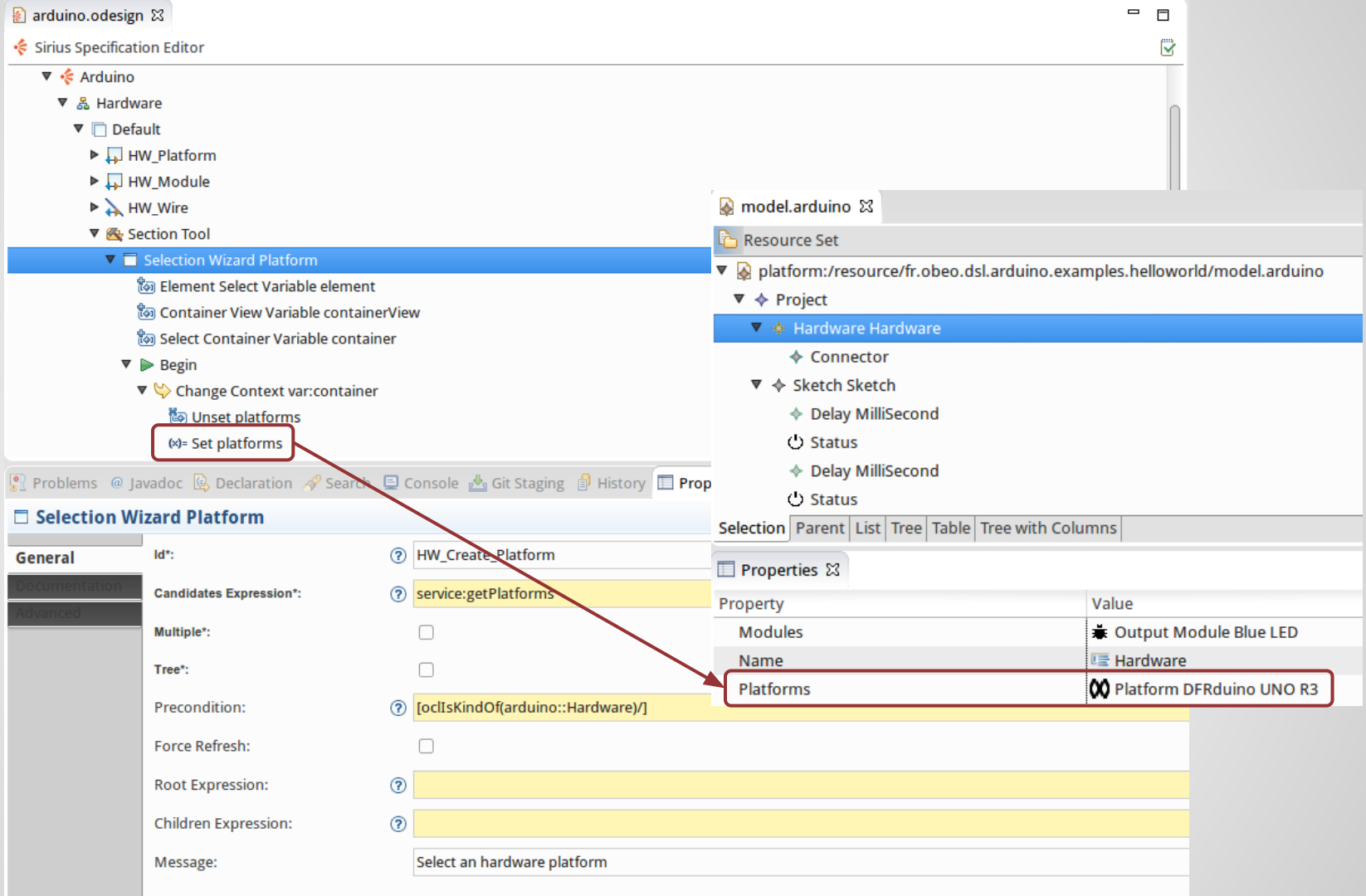
Defined thanks to

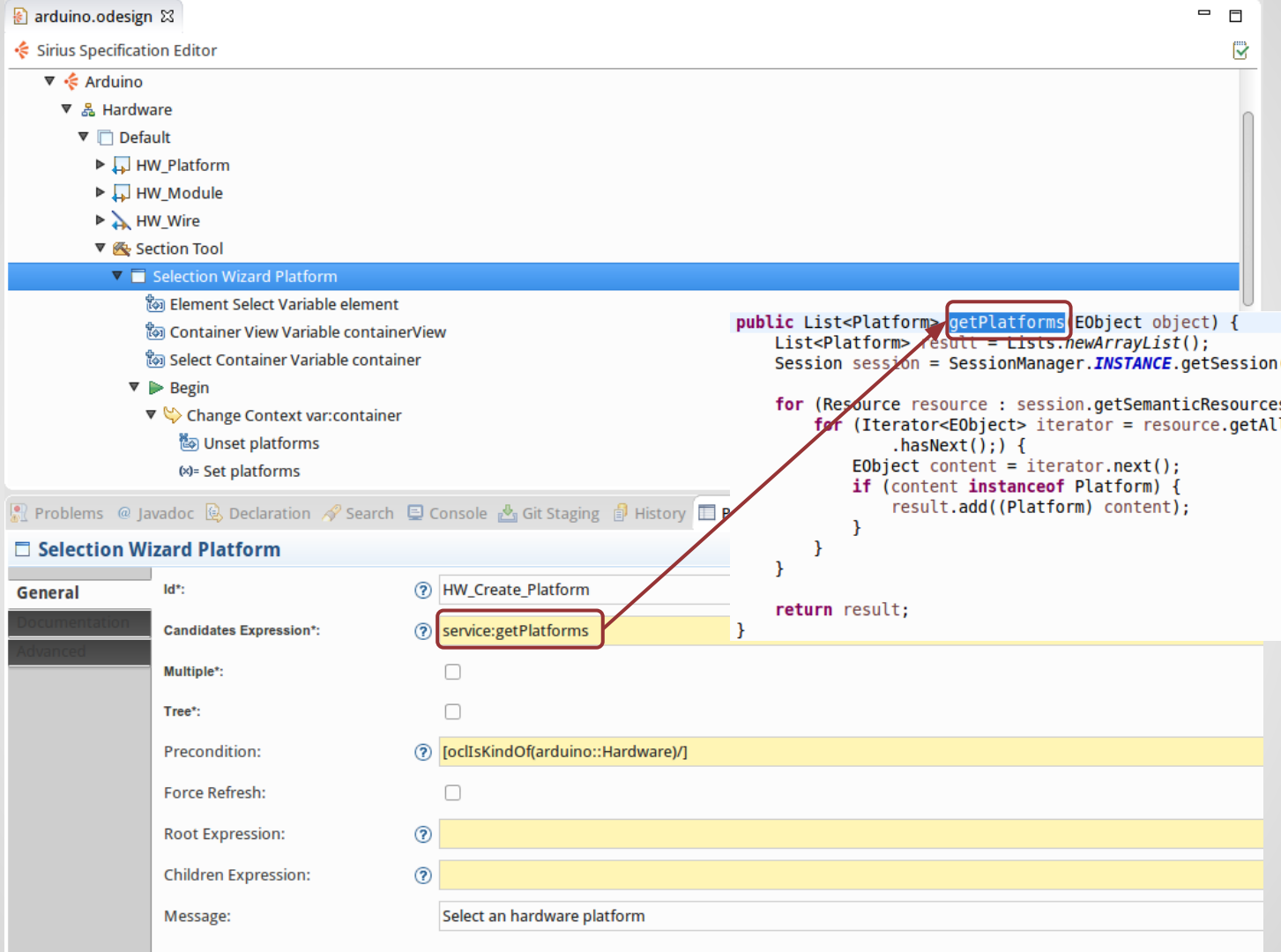
a simple **action langage**

and/or

Java services


arduino.odesign
Sirius Specification Editor
Arduino
Hardware
Default
HW_Platform
HW_Module
HW_Wire
Section Tool
Selection Wizard Platform
Element Select Variable element
Container View Variable containerView
Select Container Variable container
Begin
Change Context var:container
Unset platforms
Set platforms
Arduino Designer
Quick Access
Dashboard
*Hardware
*Sketch
Palette
Tool
Platform
Module
Wire
DFRduino UNO R3
Blue LED
Problems
Javadoc
Declaration
Search
Console
Git Staging
History
Properties
Selection Wizard Platform
General
Documentation
Advanced
Id*:
HW_Create_Platform
Candidates Expression*:
service:getPlatforms
Multiple*:
Tree*:
Precondition:
[oclIsKindOf(arduino::Hardware)/]
Force Refresh:
Root Expression:
Children Expression:
Message:
Select an hardware platform
Select an Arduino platform
Select an hardware platform
Enter prefix or pattern (? = any character, * = any String):
Platform DFRduino UNO R3
Platform DFRduino UNO R3
Cancel
Finish


arduino.odesign
Sirius Specification Editor
Arduino
Hardware
Default
HW_Platform
HW_Module
HW_Wire
Section Tool
Selection Wizard Platform
Element Select Variable element
Container View Variable containerView
Select Container Variable container
Begin
Change Context var:container
Unset platforms
(x)= Set platforms
model.arduino
Resource Set
platform:/resource/fr.obeo.dsl.arduino.examples.helloworld/model.arduino
Project
Hardware Hardware
Connector
Sketch Sketch
Delay MilliSecond
Status
Delay MilliSecond
Status
Selection
Parent
List
Tree
Table
Tree with Columns
Properties
Property
Value
Modules
Output Module Blue LED
Name
Hardware
Platforms
Platform DFRduino UNO R3
Problems
Javadoc
Declaration
Search
Console
Git Staging
History
Selection Wizard Platform
General
Documentation
Advanced
Id*:
HW_Create_Platform
Candidates Expression*:
service:getPlatforms
Multiple*:
Tree*:
Precondition:
[oclIsKindOf(arduino::Hardware)/]
Force Refresh:
Root Expression:
Children Expression:
Message:
Select an hardware platform

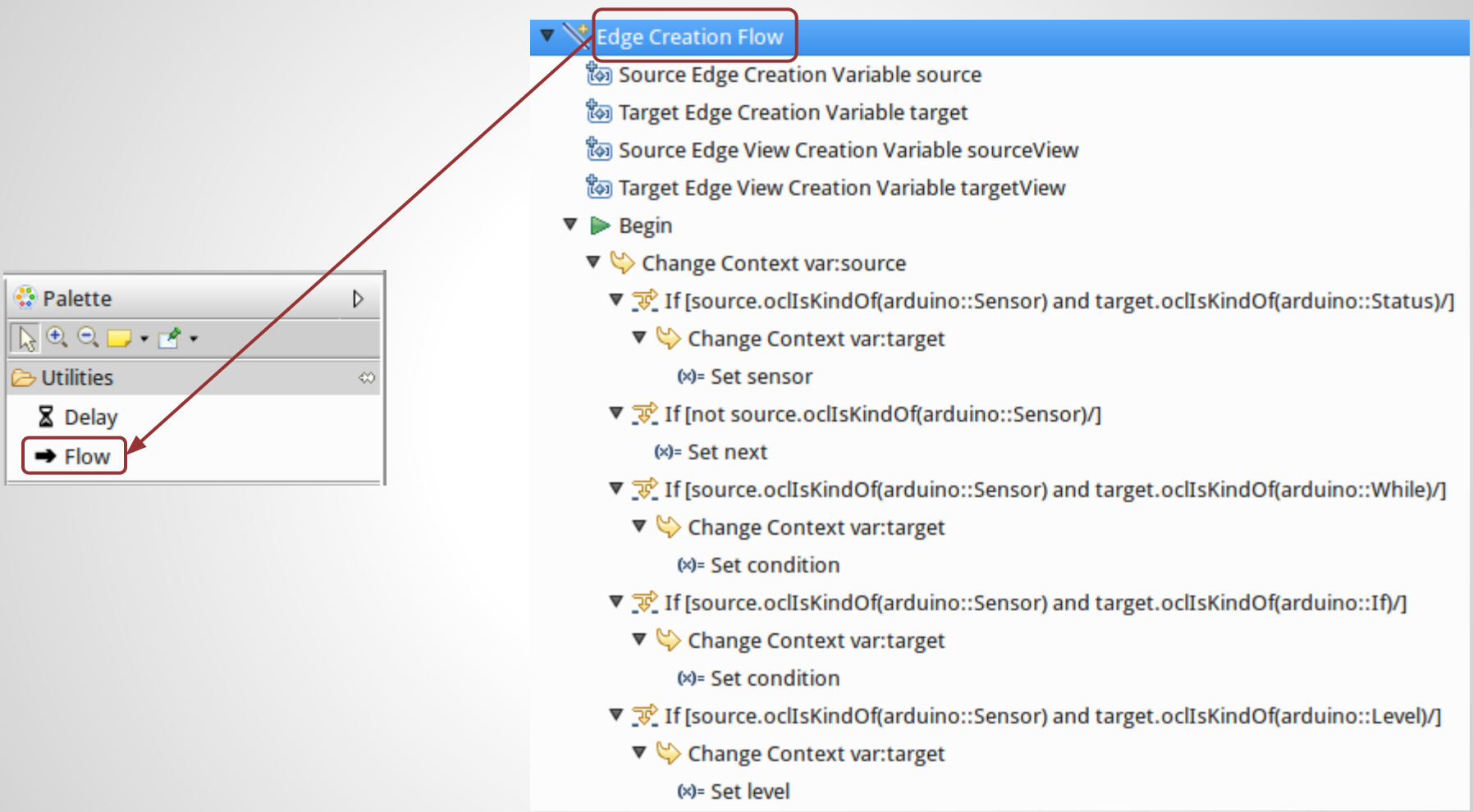

arduino.odesign
Sirius Specification Editor
Arduino
Hardware
Default
HW_Platform
HW_Module
HW_Wire
Section Tool
Selection Wizard Platform
Element Select Variable element
Container View Variable containerView
Select Container Variable container
Begin
Change Context var:container
Unset platforms
Set platforms
public List<Platform> getPlatforms(EObject object) {
List<Platform> result = Lists.newArrayList();
Session session = SessionManager.INSTANCE.getSession(object);
for (Resource resource : session.getSemanticResources()) {
for (Iterator<EObject> iterator = resource.getAllContents(); iterator
.hasNext();) {
EObject content = iterator.next();
if (content instanceof Platform) {
result.add((Platform) content);
}
}
}
return result;
}
Problems
Javadoc
Declaration
Search
Console
Git Staging
History
Selection Wizard Platform
General
Documentation
Advanced
Id*:
HW_Create_Platform
Candidates Expression*:
service:getPlatforms
Multiple*:
Tree*:
Precondition:
[oclIsKindOf(arduino::Hardware)/]
Force Refresh:
Root Expression:
Children Expression:
Message:
Select an hardware platform

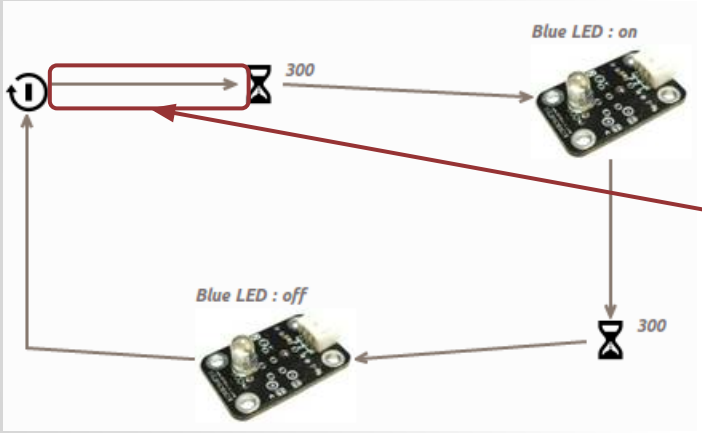
Improve UX

Improve the user experience

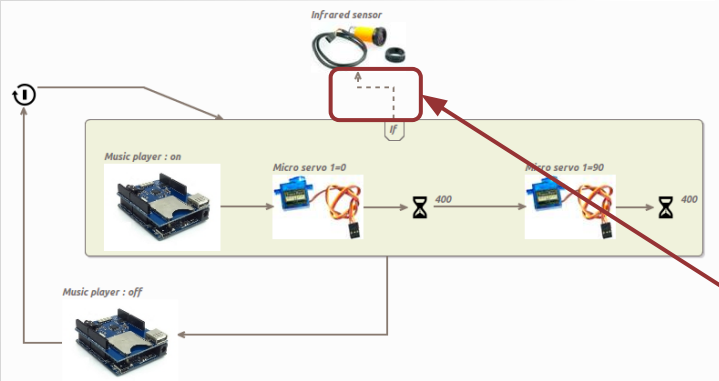
thanks to

the **tools**!

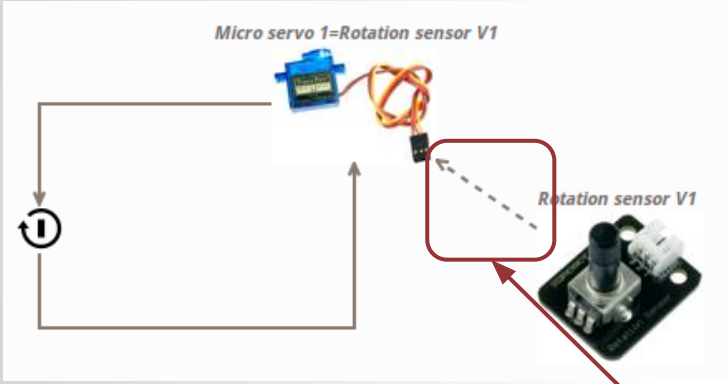

Edge Creation Flow
Source Edge Creation Variable source
Target Edge Creation Variable target
Source Edge View Creation Variable sourceView
Target Edge View Creation Variable targetView
Begin
Change Context var:source
If [source.oclIsKindOf(arduino::Sensor) and target.oclIsKindOf(arduino::Status)/]
Change Context var:target
Set sensor
If [not source.oclIsKindOf(arduino::Sensor)/]
Set next
If [source.oclIsKindOf(arduino::Sensor) and target.oclIsKindOf(arduino::While)/]
Change Context var:target
Set condition
If [source.oclIsKindOf(arduino::Sensor) and target.oclIsKindOf(arduino::If)/]
Change Context var:target
Set condition
If [source.oclIsKindOf(arduino::Sensor) and target.oclIsKindOf(arduino::Level)/]
Change Context var:target
Set level
Palette
Utilities
Delay
Flow



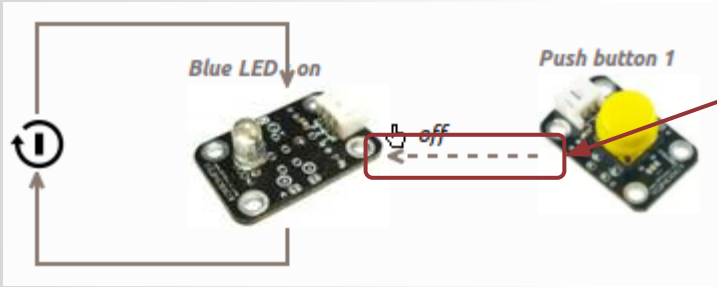
- Edge Creation Flow
  - Source Edge Creation Variable source
  - Target Edge Creation Variable target
  - Source Edge View Creation Variable sourceView
  - Target Edge View Creation Variable targetView
  - Begin
    - Change Context var:source
      - If [source.oclIsKindOf(arduino::Sensor) and target.oclIsKindOf(arduino::Status)/]
        - Change Context var:target
          - Set sensor
      - If [not source.oclIsKindOf(arduino::Sensor)/]
        - Set next
      - If [source.oclIsKindOf(arduino::Sensor) and target.oclIsKindOf(arduino::While)/]
        - Change Context var:target
          - Set condition
      - If [source.oclIsKindOf(arduino::Sensor) and target.oclIsKindOf(arduino::If)/]
        - Change Context var:target
          - Set condition
      - If [source.oclIsKindOf(arduino::Sensor) and target.oclIsKindOf(arduino::Level)/]
        - Change Context var:target
          - Set level



- Edge Creation Flow
  - Source Edge Creation Variable source
  - Target Edge Creation Variable target
  - Source Edge View Creation Variable sourceView
  - Target Edge View Creation Variable targetView
  - Begin
    - Change Context var:source
      - If [source.oclIsKindOf(arduino::Sensor) and target.oclIsKindOf(arduino::Status)/]
        - Change Context var:target
          - Set sensor
      - If [not source.oclIsKindOf(arduino::Sensor)/]
        - Set next
      - If [source.oclIsKindOf(arduino::Sensor) and target.oclIsKindOf(arduino::While)/]
        - Change Context var:target
          - Set condition
      - If [source.oclIsKindOf(arduino::Sensor) and target.oclIsKindOf(arduino::If)/]
        - Change Context var:target
          - Set condition
      - If [source.oclIsKindOf(arduino::Sensor) and target.oclIsKindOf(arduino::Level)/]
        - Change Context var:target
          - Set level

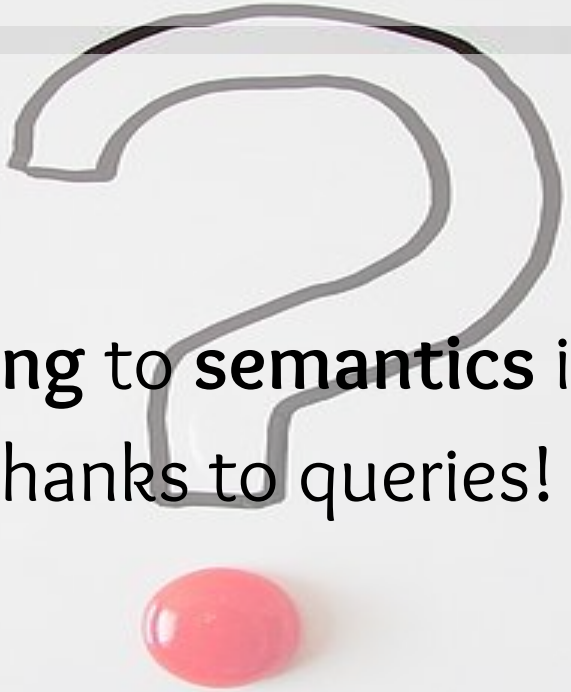


- Edge Creation Flow
  - Source Edge Creation Variable source
  - Target Edge Creation Variable target
  - Source Edge View Creation Variable sourceView
  - Target Edge View Creation Variable targetView
  - Begin
    - Change Context var:source
      - If [source.oclIsKindOf(arduino::Sensor) and target.oclIsKindOf(arduino::Status)/]
        - Change Context var:target
          - (x)= Set sensor
      - If [not source.oclIsKindOf(arduino::Sensor)/]
        - (x)= Set next
      - If [source.oclIsKindOf(arduino::Sensor) and target.oclIsKindOf(arduino::While)/]
        - Change Context var:target
          - (x)= Set condition
      - If [source.oclIsKindOf(arduino::Sensor) and target.oclIsKindOf(arduino::If)/]
        - Change Context var:target
          - (x)= Set condition
      - If [source.oclIsKindOf(arduino::Sensor) and target.oclIsKindOf(arduino::Level)/]
        - Change Context var:target
          - (x)= Set level



- Edge Creation Flow
  - Source Edge Creation Variable source
  - Target Edge Creation Variable target
  - Source Edge View Creation Variable sourceView
  - Target Edge View Creation Variable targetView
  - Begin
    - Change Context var:source
      - If [source.oclIsKindOf(arduino::Sensor) and target.oclIsKindOf(arduino::Status)/]
        - Change Context var:target
          - (x)= Set sensor
      - If [not source.oclIsKindOf(arduino::Sensor)/]
        - (x)= Set next
      - If [source.oclIsKindOf(arduino::Sensor) and target.oclIsKindOf(arduino::While)/]
        - Change Context var:target
          - (x)= Set condition
      - If [source.oclIsKindOf(arduino::Sensor) and target.oclIsKindOf(arduino::If)/]
        - Change Context var:target
          - (x)= Set condition
      - If [source.oclIsKindOf(arduino::Sensor) and target.oclIsKindOf(arduino::Level)/]
        - Change Context var:target
          - (x)= Set level

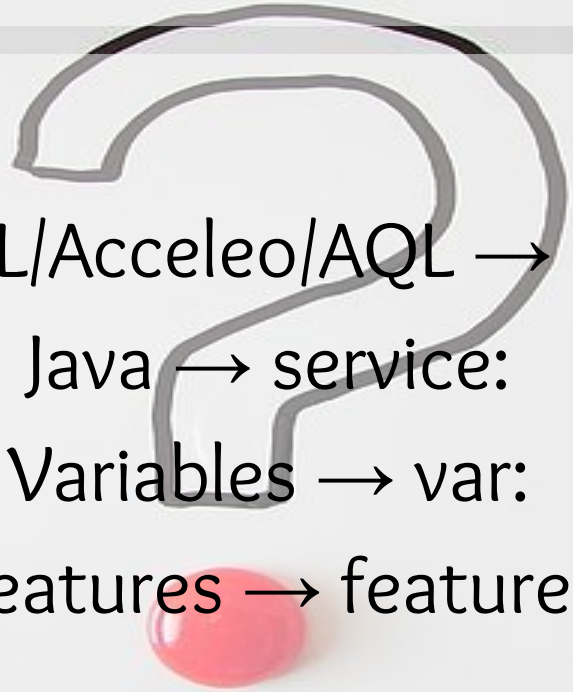
Queries



Coupling to **semantics** is **low**

thanks to queries!

Queries

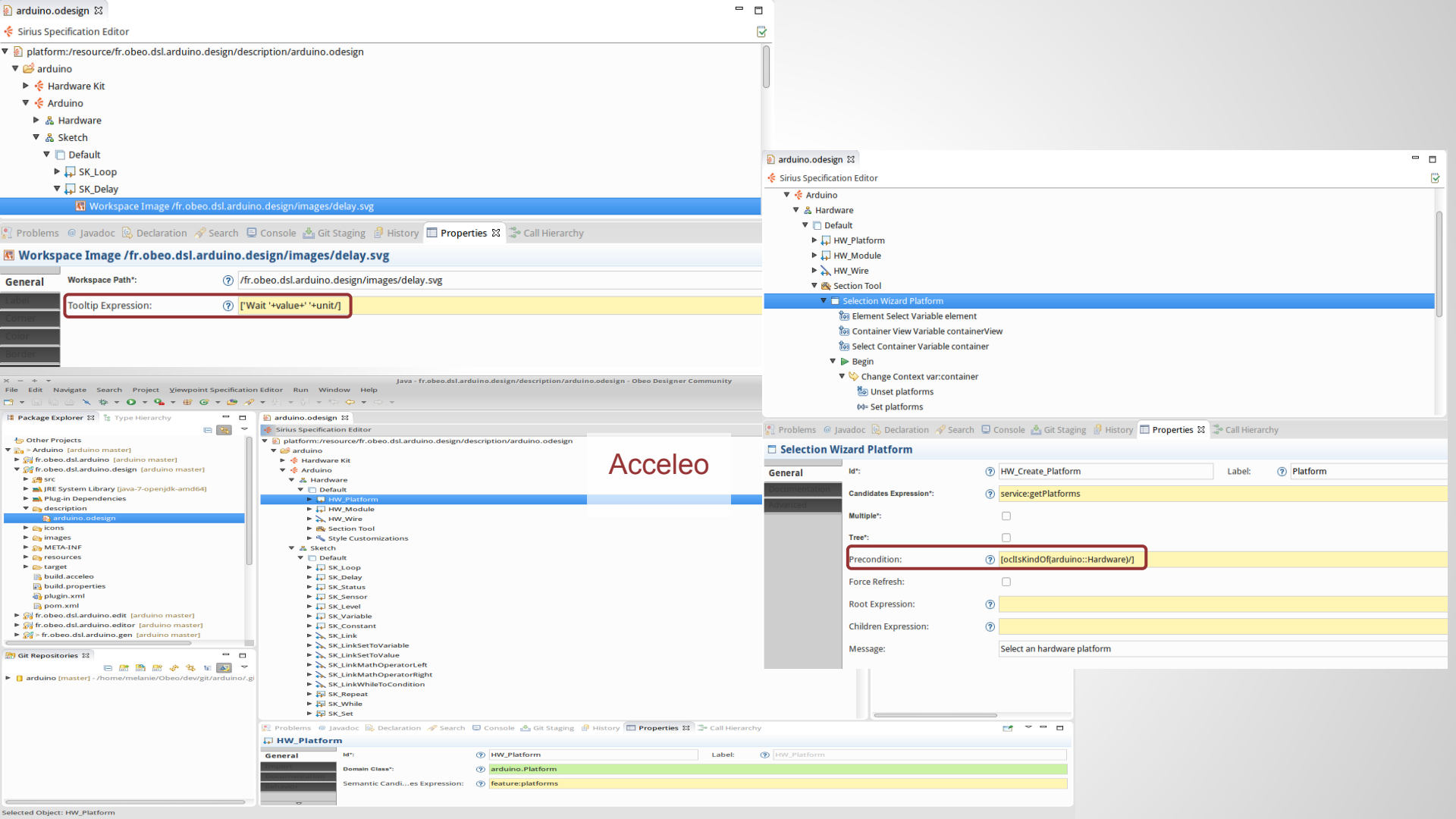


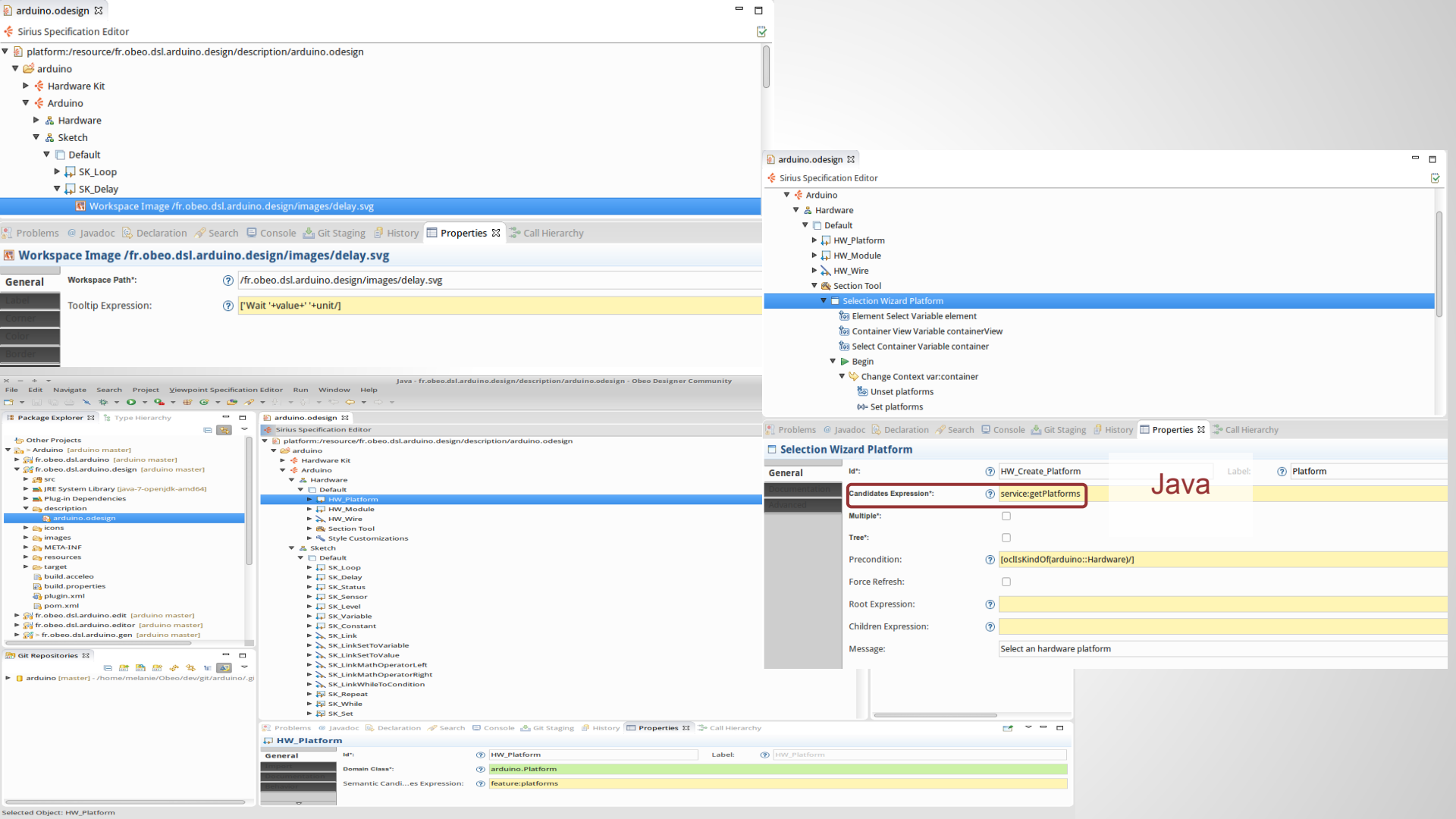
OCL/Acceleo/AQL → [/]

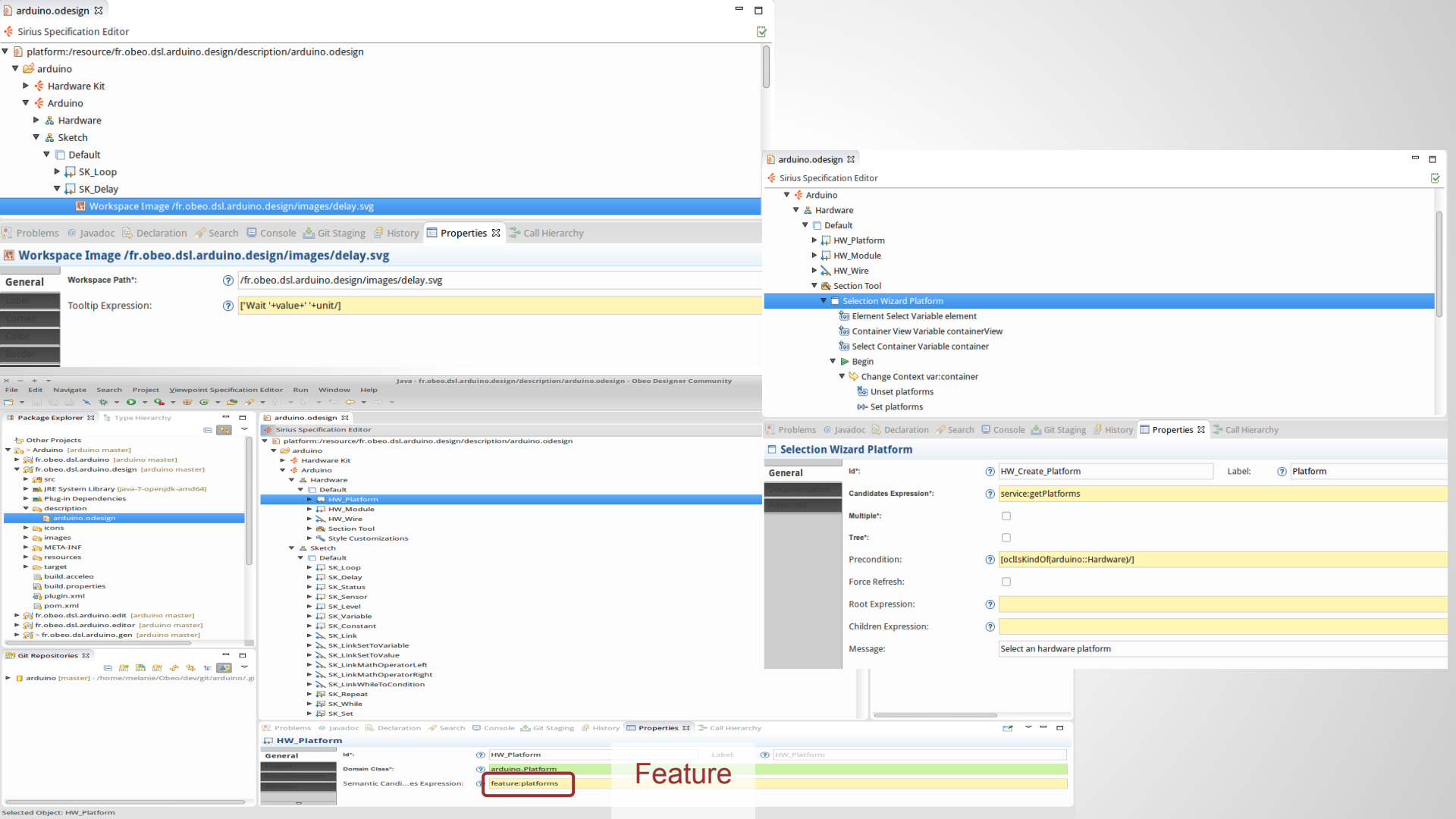
Java → service:

Variables → var:

Features → feature:


arduino.odesign
Sirius Specification Editor
platform:/resource/fr.obeo.dsl.arduino.design/description/arduino.odesign
arduino
Hardware Kit
Arduino
Hardware
Sketch
Default
SK_Loop
SK_Delay
Workspace Image /fr.obeo.dsl.arduino.design/images/delay.svg
Problems
Javadoc
Declaration
Search
Console
Git Staging
History
Properties
Call Hierarchy
Workspace Image /fr.obeo.dsl.arduino.design/images/delay.svg
General
Workspace Path*:
/fr.obeo.dsl.arduino.design/images/delay.svg
Tooltip Expression:
['Wait '+value+' '+unit/]
HW_Platform
HW_Module
HW_Wire
Section Tool
Selection Wizard Platform
Element Select Variable element
Container View Variable containerView
Select Container Variable container
Begin
Change Context var:container
Unset platforms
Set platforms
Acceleo
Id*:
HW_Create_Platform
Label:
Platform
Candidates Expression*:
service:getPlatforms
Multiple*:
Tree*:
Precondition:
[oclIsKindOf(arduino::Hardware)/]
Force Refresh:
Root Expression:
Children Expression:
Message:
Select an hardware platform
Domain Class*:
arduino.Platform
Semantic Candi...es Expression:
feature:platforms


arduino.odesign
Sirius Specification Editor
platform:/resource/fr.obeo.dsl.arduino.design/description/arduino.odesign
arduino
Hardware Kit
Arduino
Hardware
Sketch
Default
SK_Loop
SK_Delay
Workspace Image /fr.obeo.dsl.arduino.design/images/delay.svg
Problems
Javadoc
Declaration
Search
Console
Git Staging
History
Properties
Call Hierarchy
Workspace Image /fr.obeo.dsl.arduino.design/images/delay.svg
General
Workspace Path*:
/fr.obeo.dsl.arduino.design/images/delay.svg
Tooltip Expression:
['Wait '+value+' '+unit/]
Arduino
Hardware
Default
HW_Platform
HW_Module
HW_Wire
Section Tool
Selection Wizard Platform
Element Select Variable element
Container View Variable containerView
Select Container Variable container
Begin
Change Context var:container
Unset platforms
Set platforms
Selection Wizard Platform
Id*:
HW_Create_Platform
Label:
Platform
Candidates Expression*:
service:getPlatforms
Java
Multiple*:
Tree*:
Precondition:
[oclIsKindOf(arduino::Hardware)/]
Force Refresh:
Root Expression:
Children Expression:
Message:
Select an hardware platform
HW_Platform
Id*:
HW_Platform
Label:
HW_Platform
Domain Class*:
arduino.Platform
Semantic Candi...es Expression:
feature:platforms


arduino.odesign
Sirius Specification Editor
platform:/resource/fr.obeo.dsl.arduino.design/description/arduino.odesign
arduino
Hardware Kit
Arduino
Hardware
Sketch
Default
SK_Loop
SK_Delay
Workspace Image /fr.obeo.dsl.arduino.design/images/delay.svg
Problems
Javadoc
Declaration
Search
Console
Git Staging
History
Properties
Call Hierarchy
Workspace Image /fr.obeo.dsl.arduino.design/images/delay.svg
General
Workspace Path*:
/fr.obeo.dsl.arduino.design/images/delay.svg
Tooltip Expression:
['Wait '+value+' '+unit/]
HW_Platform
HW_Module
HW_Wire
Section Tool
Selection Wizard Platform
Element Select Variable element
Container View Variable containerView
Select Container Variable container
Begin
Change Context var:container
Unset platforms
Set platforms
Id*:
HW_Create_Platform
Label:
Platform
Candidates Expression*:
service:getPlatforms
Multiple*:
Tree*:
Precondition:
[oclIsKindOf(arduino::Hardware)/]
Force Refresh:
Root Expression:
Children Expression:
Message:
Select an hardware platform
Id*:
HW_Platform
Domain Class*:
arduino.Platform
Semantic Candi...es Expression:
feature:platforms
Feature

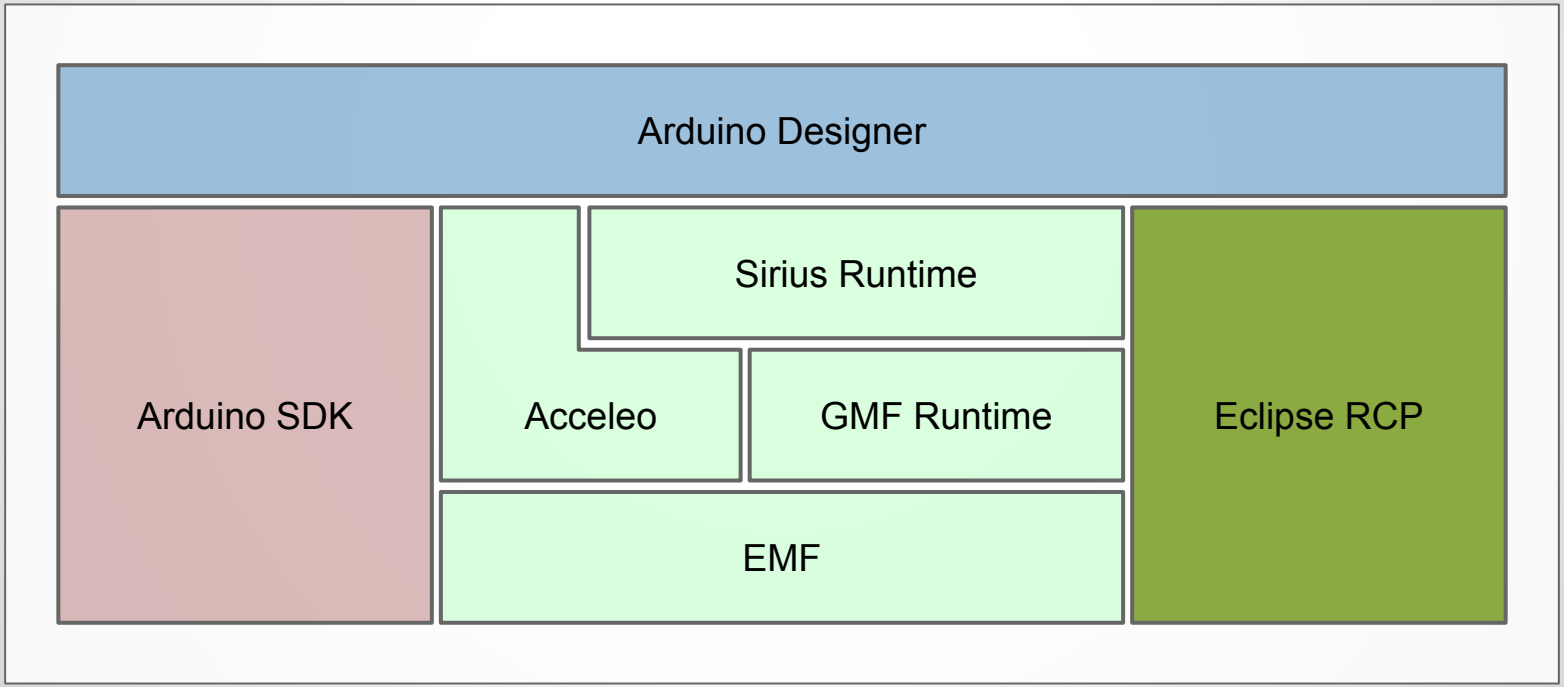
Remind #2

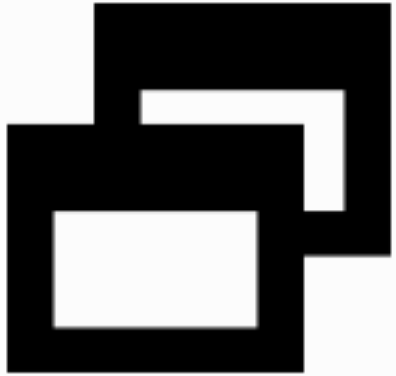
Simplify UI thanks to advanced tools

Diet RCP

Keep **only**
in the **UI**
what's really
necessary

Simplified UI




Dashboard



```
public void createPartControl(final Composite parent) {

    shell = parent.getShell();

    // Create the form
    toolkit = new FormToolkit(parent.getDisplay());
    form = toolkit.createScrolledForm(parent);
    setFormText();
    GridLayout formLayout = new GridLayout();
    formLayout.numColumns = 3;
    formLayout.horizontalSpacing = 100;
    formLayout.verticalSpacing = 50;
    formLayout.marginWidth = 100;
    formLayout.marginHeight = 100;
    form.getBody().setLayout(formLayout);

    // Create all the hyperlinks
    createNewProjectHyperLink(parent, shell);
    createOpenProjectHyperLink(parent, shell);
    createPreferencesHyperLink(shell);
    createHardwareHyperLink(parent, shell);
    createSketchHyperLink(parent, shell);
    createUploadHyperLink(parent, shell);

    this.getSite().getPage().addPartListener(this);
    ResourcesPlugin.getWorkspace().addResourceChangeListener(this);
}
```

```
public void createPartControl(final Composite parent) {

    shell = parent.getShell();

    // Create the form
    toolkit = new FormToolkit(parent.getDisplay());
    form = toolkit.createScrolledForm(parent);
    setFormText();
    GridLayout formLayout = new GridLayout();
    formLayout.numColumns = 3;
    formLayout.horizontalSpacing = 100;
    formLayout.verticalSpacing = 50;
    formLayout.marginWidth = 100;
    formLayout.marginHeight = 100;
    form.getBody().setLayout(formLayout);

    // Create all the hyperlinks
    createNewProjectHyperLink(parent, shell);
    createOpenProjectHyperLink(parent, shell);
    createPreferencesHyperLink(shell);
    createHardwareHyperLink(parent, shell);
    createSketchHyperLink(parent, shell);
    createUploadHyperLink(parent, shell);

    this.getSite().getPage().addPartListener(this);
    ResourcesPlugin.getWorkspace().addResourceChangeListener(this);
}
```

```
private void createNewProjectHyperLink(final Composite parent,
        final Shell shell) {
    ImageHyperlink newProjectLink = createImageHyperlink(form.getBody(),
            "icons/128x128/newProject.png", "icons/128x128/newProjectHover.png",
            "Create a new project");
    newProjectLink.addHyperlinkListener(new HyperlinkAdapter() {
        public void linkActivated(HyperlinkEvent e) {
            newProjectDialog(shell);
        }
    });
}
```

```
private ImageHyperlink createImageHyperlink(Composite parent,
        String imagePath, String hoverImagePath, String toolTipText) {
    ImageHyperlink imageLink = toolkit.createImageHyperlink(parent,
            SWT.WRAP);
    Image image = getImage(imagePath);
    imageLink.setImage(image);
    Image hoverImage = getImage(hoverImagePath);
    imageLink.setHoverImage(hoverImage);
    imageLink.setToolTipText(toolTipText);
    return imageLink;
}
```

```
private void createNewProjectHyperLink(final Composite parent,
        final Shell shell) {
    ImageHyperlink newProjectLink = createImageHyperlink(form.getBody(),
            "icons/128x128/newProject.png", "icons/128x128/newProjectHover.png",
            "Create a new project");
    newProjectLink.addHyperlinkListener(new HyperlinkAdapter() {
        public void linkActivated(HyperlinkEvent e) {
            newProjectDialog(shell);
        }
    });
}
```

```
private ImageHyperlink createImageHyperlink(Composite parent,
        String imagePath, String hoverImagePath, String toolTipText) {
    ImageHyperlink imageLink = toolkit.createImageHyperlink(parent,
            SWT.WRAP);
    Image image = getImage(imagePath);
    imageLink.setImage(image);
    Image hoverImage = getImage(hoverImagePath);
    imageLink.setHoverImage(hoverImage);
    imageLink.setToolTipText(toolTipText);
    return imageLink;
}
```

```
public class DashboardView extends ViewPart implements IPartListener2,
        IResourceChangeListener {
```

```
public void createPartControl(final Composite parent) {

    shell = parent.getShell();

    // Create the form
    toolkit = new FormToolkit(parent.getDisplay());
    form = toolkit.createScrolledForm(parent);
    setFormText();
    GridLayout formLayout = new GridLayout();
    formLayout.numColumns = 3;
    formLayout.horizontalSpacing = 100;
    formLayout.verticalSpacing = 50;
    formLayout.marginWidth = 100;
    formLayout.marginHeight = 100;
    form.getBody().setLayout(formLayout);

    // Create all the hyperlinks
    createNewProjectHyperLink(parent, shell);
    createOpenProjectHyperLink(parent, shell);
    createPreferencesHyperLink(shell);
    createHardwareHyperLink(parent, shell);
    createSketchHyperLink(parent, shell);
    createUploadHyperLink(parent, shell);

    this.getSite().getPage().addPartListener(this);
    ResourcesPlugin.getWorkspace().addResourceChangeListener(this);
}
```

```
public void partBroughtToTop(IWorkbenchPartReference partRef) {
    refreshForm();
}
```

```
public class DashboardView extends ViewPart implements IPartListener2,
        IResourceChangeListener {

    public void createPartControl(final Composite parent) {

        shell = parent.getShell();

        // Create the form
        toolkit = new FormToolkit(parent.getDisplay());
        form = toolkit.createScrolledForm(parent);
        setFormText();
        GridLayout formLayout = new GridLayout();
        formLayout.numColumns = 3;
        formLayout.horizontalSpacing = 100;
        formLayout.verticalSpacing = 50;
        formLayout.marginWidth = 100;
        formLayout.marginHeight = 100;
        form.getBody().setLayout(formLayout);

        // Create all the hyperlinks
        createNewProjectHyperLink(parent, shell);
        createOpenProjectHyperLink(parent, shell);
        createPreferencesHyperLink(shell);
        createHardwareHyperLink(parent, shell);
        createSketchHyperLink(parent, shell);
        createUploadHyperLink(parent, shell);

        this.getSite().getPage().addPartListener(this);
        ResourcesPlugin.getWorkspace().addResourceChangeListener(this);
    }

    public void partBroughtToTop(IWorkbenchPartReference partRef) {
        refreshForm();
    }
```




```
public class DashboardView extends ViewPart implements IPartListener2,
        IResourceChangeListener {

    public void createPartControl(final Composite parent) {

        shell = parent.getShell();

        // Create the form
        toolkit = new FormToolkit(parent.getDisplay());
        form = toolkit.createScrolledForm(parent);
        setFormText();
        GridLayout formLayout = new GridLayout();
        formLayout.numColumns = 3;
        formLayout.horizontalSpacing = 100;
        formLayout.verticalSpacing = 50;
        formLayout.marginWidth = 100;
        formLayout.marginHeight = 100;
        form.getBody().setLayout(formLayout);

        // Create all the hyperlinks
        createNewProjectHyperLink(parent, shell);
        createOpenProjectHyperLink(parent, shell);
        createPreferencesHyperLink(shell);
        createHardwareHyperLink(parent, shell);
        createSketchHyperLink(parent, shell);
        createUploadHyperLink(parent, shell);

        this.getSite().getPage().addPartListener(this);
        ResourcesPlugin.getWorkspace().addResourceChangeListener(this);
    }

    public void partBroughtToTop(IWorkbenchPartReference partRef) {
        refreshForm();
    }
```

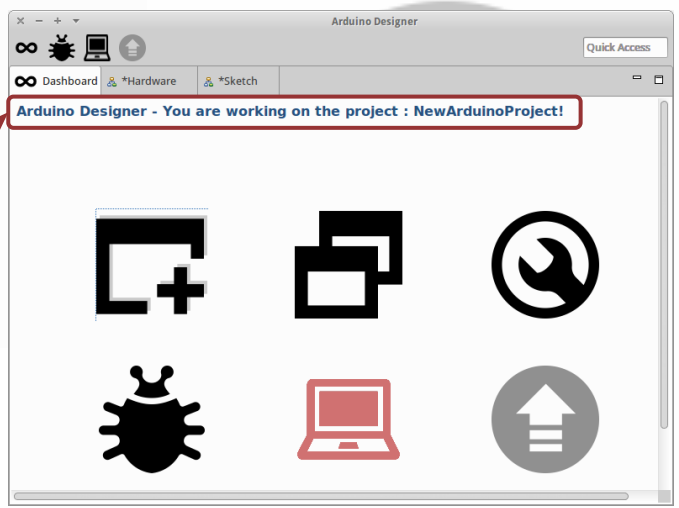
```
private void refreshForm() {
    Display.getDefault().syncExec(new Runnable() {
        @Override
        public void run() {
            setFormText();
            setHardwareLinkEnablement();
            setSketchLinkEnablement();
            setUploadLinkEnablement();
            form.getParent().update();
        }
    });
}
```

```
private void setFormText() {
    String text = "Arduino Designer - ";
    if (!service.isProjectOpened()) {
        text += "Select or create a project!";
    } else {
        text += "You are working on the project : "
                + service.getWorkspaceProject().getName() + "!";
    }
    form.setText(text);
}
```

```
private void setSketchLinkEnablement() {
    if (service.isValidHardware()) {
        sketchLink.setEnabled(true);
        if (service.isValidSketch()) {
            sketchLink.setToolTipText(SKETCH_MSG);
            sketchLink.setImage(getImage(SKETCH_IMAGE));
            sketchLink.setHoverImage(getImage(SKETCH_HOVER_IMAGE));
        } else {
            sketchLink.setToolTipText(SKETCH_INVALID_MSG);
            sketchLink.setImage(getImage(SKETCH_INVALID_IMAGE));
            sketchLink.setHoverImage(getImage(SKETCH_INVALID_HOVER_IMAGE));
        }
    } else {
        sketchLink.setEnabled(false);
    }
}
```

```
private void refreshForm() {
    Display.getDefault().syncExec(new Runnable() {
        @Override
        public void run() {
            setFormText();
            setHardwareLinkEnablement();
            setSketchLinkEnablement();
            setUploadLinkEnablement();
            form.getParent().update();
        }
    });
}
```

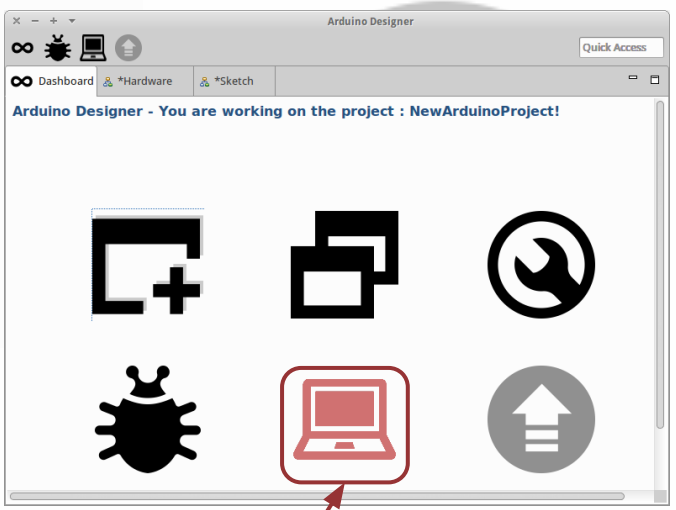
```
private void setFormText() {
    String text = "Arduino Designer - ";
    if (!service.isProjectOpened()) {
        text += "Select or create a project!";
    } else {
        text += "You are working on the project : "
                + service.getWorkspaceProject().getName() + "!";
    }
    form.setText(text);
}
```




```
private void setSketchLinkEnablement() {
    if (service.isValidHardware()) {
        sketchLink.setEnabled(true);
        if (service.isValidSketch()) {
            sketchLink.setToolTipText(SKETCH_MSG);
            sketchLink.setImage(getImage(SKETCH_IMAGE));
            sketchLink.setHoverImage(getImage(SKETCH_HOVER_IMAGE));
        } else {
            sketchLink.setToolTipText(SKETCH_INVALID_MSG);
            sketchLink.setImage(getImage(SKETCH_INVALID_IMAGE));
            sketchLink.setHoverImage(getImage(SKETCH_INVALID_HOVER_IMAGE));
        }
    } else {
        sketchLink.setEnabled(false);
    }
}
```

```
private void refreshForm() {
    Display.getDefault().syncExec(new Runnable() {
        @Override
        public void run() {
            setFormText();
            setHardwareLinkEnablement();
            setSketchLinkEnablement();
            setUploadLinkEnablement();
            form.getParent().update();
        }
    });
}
```

```
private void setFormText() {
    String text = "Arduino Designer - ";
    if (!service.isProjectOpened()) {
        text += "Select or create a project!";
    } else {
        text += "You are working on the project : "
                + service.getWorkspaceProject().getName() + "!";
    }
    form.setText(text);
}
```




```
private void setSketchLinkEnablement() {
    if (service.isValidHardware()) {
        sketchLink.setEnabled(true);
        if (service.isValidSketch()) {
            sketchLink.setToolTipText(SKETCH_MSG);
            sketchLink.setImage(getImage(SKETCH_IMAGE));
            sketchLink.setHoverImage(getImage(SKETCH_HOVER_IMAGE));
        } else {
            sketchLink.setToolTipText(SKETCH_INVALID_MSG);
            sketchLink.setImage(getImage(SKETCH_INVALID_IMAGE));
            sketchLink.setHoverImage(getImage(SKETCH_INVALID_HOVER_IMAGE));
        }
    } else {
        sketchLink.setEnabled(false);
    }
}
```

Diet RCP

Simplify the workflow

Use Sirius API

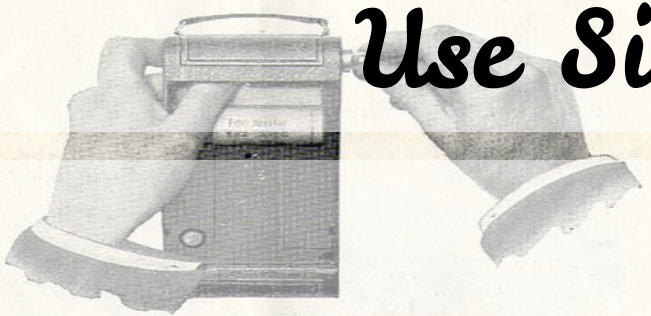


Fig 5 Replacing the Reel.

until the web catches in the slotted end (Fig. 5). Enough of the paper should then be unrolled to

Fig. 6. Inserting the Spool in the "Kodak"

allow the full spool to be placed in the recess at the other end of the camera (Fig. 6).

V. Now lower the upper end of the body into the case (Fig. 7), push back into position, and allow the lower end to drop into the case; lock the spring catch.

While loading the spool into the camera, from the time the gummed slip on the fresh roll of

Fig. 7. Replacing the Body in the Back.

film is broken until the body is back into the case, keep the paper wound tightly on the spool. If it is allowed to loosen, light will get in and the film will be spoiled.

VI. The film is covered with paper, the excess of which must be wound off before a picture can be taken.

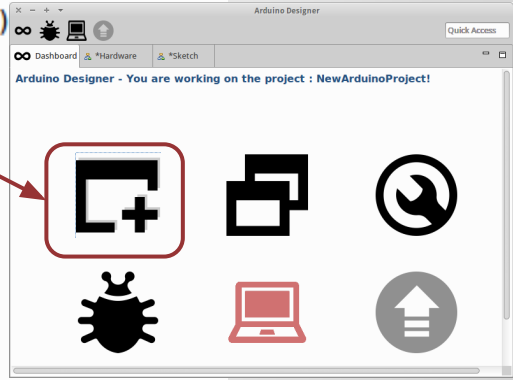
Turn the key *slowly*, and watch the little red window in the back of the camera (Fig. 8). When 15 to 18 half turns have been given, first a hand,

Have a look to the

Sirius Developer Manual

Create a project

```
public void createProject(IProgressMonitor monitor, IProject project) {
    try {
        project.create(monitor);
        project.open(monitor);
        ModelingProjectManager.INSTANCE.convertToModelingProject(project, monitor);
    } catch (CoreException e) {
        ArduinoUiActivator.log(Status.ERROR, "Open project failed", e);
    }

    String modelPath = '/' + project.getName(); //$NON-NLS-1$
    final Session session = createAird(
            project,
            URI.createPlatformResourceURI(modelPath
                    + "/representations.aird", true), monitor);

    final String semanticModelPath = getSemanticModelPath(session);
    initSemanticModel(session, semanticModelPath, monitor);

    final String[] viewpointsToActivate = { ARDUINO_VP };
    enableViewpoints(session, viewpointsToActivate);

    openHardware(session);
}
```



Convert to a modeling project

```
public void createProject(IProgressMonitor monitor, IProject project) {
    try {
        project.create(monitor);
        project.open(monitor);
        ModelingProjectManager.INSTANCE.convertToModelingProject(project, monitor);
    } catch (CoreException e) {
        ArduinoUiActivator.log(Status.ERROR, "Open project failed", e);
    }

    String modelPath = '/' + project.getName(); //$NON-NLS-1$
    final Session session = createAird(
            project,
            URI.createPlatformResourceURI(modelPath
                    + "/representations.aird", true), monitor);

    final String semanticModelPath = getSemanticModelPath(session);
    initSemanticModel(session, semanticModelPath, monitor);

    final String[] viewpointsToActivate = { ARDUINO_VP };
    enableViewpoints(session, viewpointsToActivate);

    openHardware(session);
}
```

Convert to a modeling project

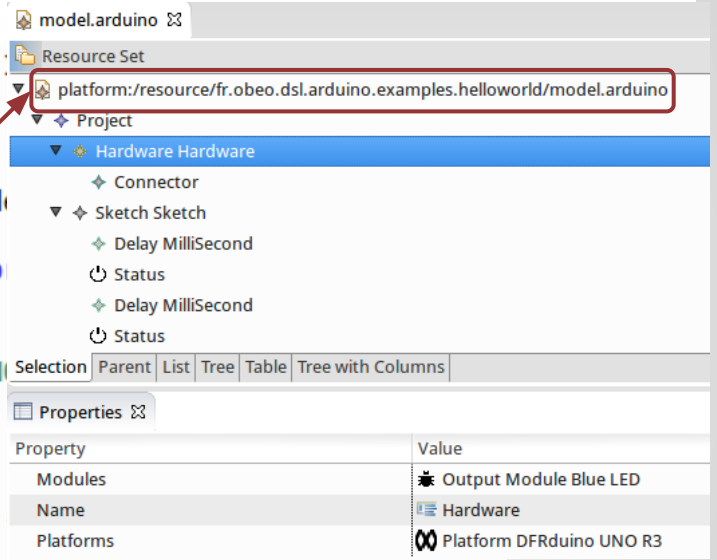
```
public void createProject(IProgressMonitor monitor, IProject project) {
    try {
        project.create(monitor);
        project.open(monitor);
        ModelingProjectManager.INSTANCE.convertToModelingProject(project, monitor);
    } catch (CoreException e) {
        ArduinoUiActivator.log(Status.ERROR, "Open project failed", e);
    }

    String modelPath = '/' + project.getName(); //$NON-NLS-1$
    final Session session = createAird(
            project,
            URI.createPlatformResourceURI(modelPath
                    + "/representations.aird", true), monitor);

    final String semanticModelPath = getSemanticModelPath(session);
    initSemanticModel(session, semanticModelPath, monitor);

    final String[] viewpointsToActivate = { ARDUINO_VP };
    enableViewpoints(session, viewpointsToActivate);

    openHardware(session);
}
```

Create a session

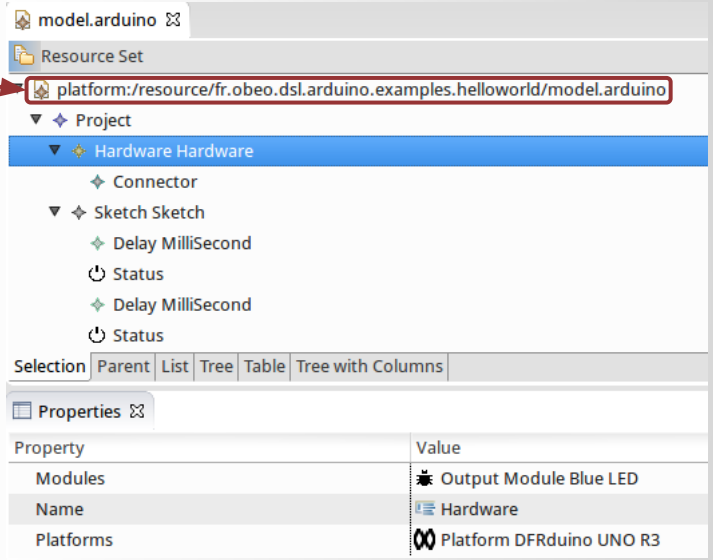
```
public void createProject(IProgressMonitor monitor, IProject project) {
    try {
        project.create(monitor);
        project.open(monitor);
        ModelingProjectManager.INSTANCE.convertToModelingProject(project, monitor);
    } catch (CoreException e) {
        ArduinoUiActivator.log(Status.ERROR, "Open project failed", e);
    }

    String modelPath = '/' + project.getName(); //$NON-NLS-1$
    final Session session = createAird(
            project,
            URI.createPlatformResourceURI(modelPath
                    + "/representations.aird", true), monitor);

    final String semanticModelPath = getSemanticModelPath(session);
    initSemanticModel(session, semanticModelPath, monitor);

    final String[] viewpointsToActivate = { ARDUINO_VP };
    enableViewpoints(session, viewpointsToActivate);

    openHardware(session);
}
```

Init the semantic model

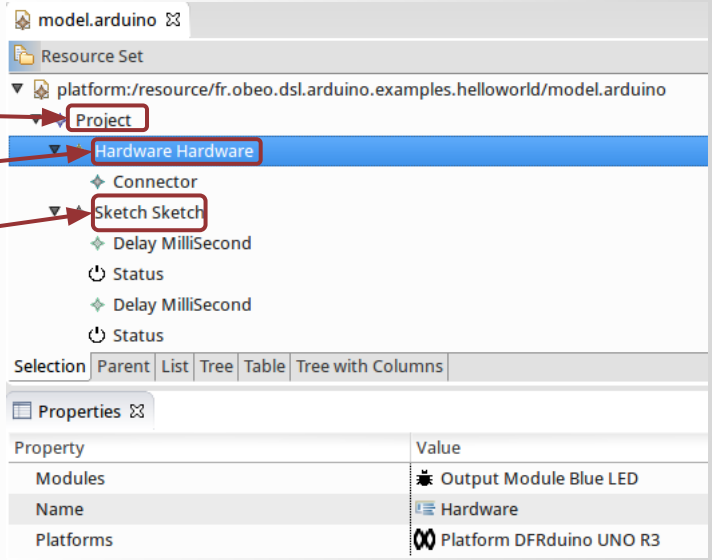
```
public void createProject(IProgressMonitor monitor, 
    try {
        project.create(monitor);
        project.open(monitor);
        ModelingProjectManager.INSTANCE.convertToMod
    } catch (CoreException e) {
        ArduinoUiActivator.log(Status.ERROR, "Open p
    }

    String modelPath = '/' + project.getName(); //$N
    final Session session = createAird(
            project,
            URI.createPlatformResourceURI(modelPath
                    + "/representations.aird", true)

    final String semanticModelPath = getSemanticModelPath(session);
    initSemanticModel(session, semanticModelPath, monitor);

    final String[] viewpointsToActivate = { ARDUINO_VP };
    enableViewpoints(session, viewpointsToActivate);

    openHardware(session);
}
```



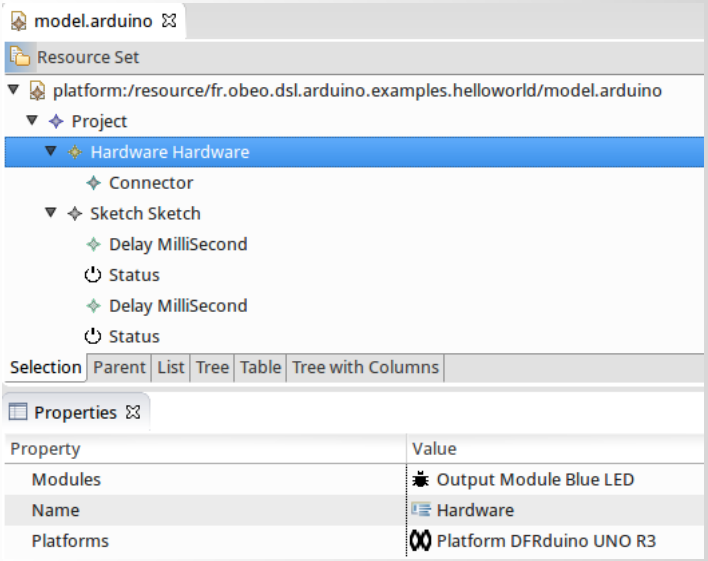
```
private void initSemanticModel(final Session session,
        final String semanticModelPath, final IProgressMonitor monitor) {
    session.getTransactionalEditingDomain()
            .getCommandStack()
            .execute(
                    new RecordingCommand(session
                            .getTransactionalEditingDomain()) {
                        @Override
                        protected void doExecute() {

                            final URI semanticModelURI = URI
                                    .createPlatformResourceURI(
                                            semanticModelPath, true);
                            Resource res = new ResourceSetImpl()
                                    .createResource(semanticModelURI);
                            // Add the initial model object to the contents.
                            final Project rootObject = ArduinoFactory.eINSTANCE
                                    .createProject();

                            if (rootObject != null) {
                                res.getContents().add(rootObject);
                                final Hardware hardware = ArduinoFactory.eINSTANCE
                                        .createHardware();
                                hardware.setName("Hardware");
                                rootObject.setHardware(hardware);
                                final Sketch sketch = ArduinoFactory.eINSTANCE
                                        .createSketch();
                                sketch.setName("Sketch");
                                sketch.setHardware(hardware);
                                rootObject.setSketch(sketch);
                            }
                            try {
                                res.save(Maps.newHashMap());
                            } catch (IOException e) {
                                ArduinoUiActivator.log(Status.ERROR,
                                        "Init semantic model failed", e);
                            }

                            session.addSemanticResource(semanticModelURI,
                                    monitor);

                            // Add ardublock kit
                            final URI defaultKitModelURI = URI
                                    .createPlatformPluginURI(
                                            "/fr.obeo.dsl.arduino.design/resources/ArdublockKit.arduino",
                                            true);
                            session.addSemanticResource(defaultKitModelURI,
                                    monitor);

                            session.save(monitor);
                        }
                    });
```



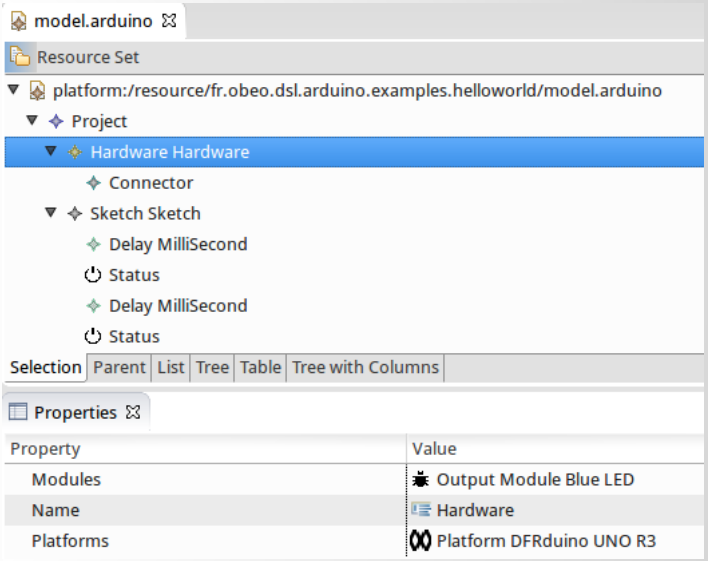
```
private void initSemanticModel(final Session session,
        final String semanticModelPath, final IProgressMonitor monitor) {
    session.getTransactionalEditingDomain()
            .getCommandStack()
            .execute(
                    new RecordingCommand(session
                            .getTransactionalEditingDomain()) {
                        @Override
                        protected void doExecute() {

                            final URI semanticModelURI = URI
                                    .createPlatformResourceURI(
                                            semanticModelPath, true);
                            Resource res = new ResourceSetImpl()
                                    .createResource(semanticModelURI);
                            // Add the initial model object to the contents.
                            final Project rootObject = ArduinoFactory.eINSTANCE
                                    .createProject();

                            if (rootObject != null) {
                                res.getContents().add(rootObject);
                                final Hardware hardware = ArduinoFactory.eINSTANCE
                                        .createHardware();
                                hardware.setName("Hardware");
                                rootObject.setHardware(hardware);
                                final Sketch sketch = ArduinoFactory.eINSTANCE
                                        .createSketch();
                                sketch.setName("Sketch");
                                sketch.setHardware(hardware);
                                rootObject.setSketch(sketch);
                            }
                            try {
                                res.save(Maps.newHashMap());
                            } catch (IOException e) {
                                ArduinoUiActivator.log(Status.ERROR,
                                        "Init semantic model failed", e);
                            }

                            session.addSemanticResource(semanticModelURI,
                                    monitor);

                            // Add ardublock kit
                            final URI defaultKitModelURI = URI
                                    .createPlatformPluginURI(
                                            "/fr.obeo.dsl.arduino.design/resources/ArdublockKit.arduino",
                                            true);
                            session.addSemanticResource(defaultKitModelURI,
                                    monitor);

                            session.save(monitor);
                        }
                    });
```



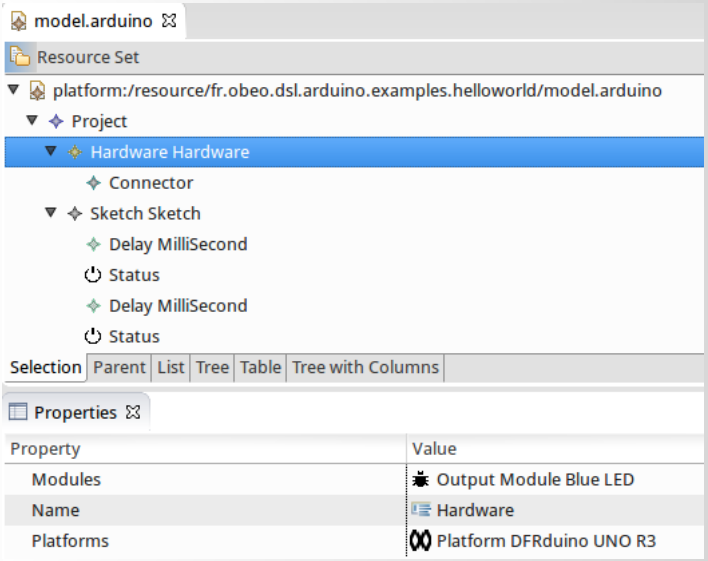
```java
private void initSemanticModel(final Session session,
        final String semanticModelPath, final IProgressMonitor monitor) {
    session.getTransactionalEditingDomain()
            .getCommandStack()
            .execute(
                    new RecordingCommand(session
                            .getTransactionalEditingDomain()) {
                        @Override
                        protected void doExecute() {

                            final URI semanticModelURI = URI
                                    .createPlatformResourceURI(
                                            semanticModelPath, true);
                            Resource res = new ResourceSetImpl()
                                    .createResource(semanticModelURI);
                            // Add the initial model object to the contents.
                            final Project rootObject = ArduinoFactory.eINSTANCE
                                    .createProject();

                            if (rootObject != null) {
                                res.getContents().add(rootObject);
                                final Hardware hardware = ArduinoFactory.eINSTANCE
                                        .createHardware();
                                hardware.setName("Hardware");
                                rootObject.setHardware(hardware);
                                final Sketch sketch = ArduinoFactory.eINSTANCE
                                        .createSketch();
                                sketch.setName("Sketch");
                                sketch.setHardware(hardware);
                                rootObject.setSketch(sketch);
                            }
                            try {
                                res.save(Maps.newHashMap());
                            } catch (IOException e) {
                                ArduinoUiActivator.log(Status.ERROR,
                                        "Init semantic model failed", e);
                            }

                            session.addSemanticResource(semanticModelURI,
                                    monitor);

                            // Add ardublock kit
                            final URI defaultKitModelURI = URI
                                    .createPlatformPluginURI(
                                            "/fr.obeo.dsl.arduino.design/resources/ArdublockKit.arduino",
                                            true);
                            session.addSemanticResource(defaultKitModelURI,
                                    monitor);

                            session.save(monitor);
                        }
                    });
```



```java
private void initSemanticModel(final Session session,
        final String semanticModelPath, final IProgressMonitor monitor) {
    session.getTransactionalEditingDomain()
            .getCommandStack()
            .execute(
                    new RecordingCommand(session
                            .getTransactionalEditingDomain()) {
                        @Override
                        protected void doExecute() {

                            final URI semanticModelURI = URI
                                    .createPlatformResourceURI(
                                            semanticModelPath, true);
                            Resource res = new ResourceSetImpl()
                                    .createResource(semanticModelURI);
                            // Add the initial model object to the contents.
                            final Project rootObject = ArduinoFactory.eINSTANCE
                                    .createProject();

                            if (rootObject != null) {
                                res.getContents().add(rootObject);
                                final Hardware hardware = ArduinoFactory.eINSTANCE
                                        .createHardware();
                                hardware.setName("Hardware");
                                rootObject.setHardware(hardware);
                                final Sketch sketch = ArduinoFactory.eINSTANCE
                                        .createSketch();
                                sketch.setName("Sketch");
                                sketch.setHardware(hardware);
                                rootObject.setSketch(sketch);
                            }
                            try {
                                res.save(Maps.newHashMap());
                            } catch (IOException e) {
                                ArduinoUiActivator.log(Status.ERROR,
                                        "Init semantic model failed", e);
                            }

                            session.addSemanticResource(semanticModelURI,
                                    monitor);

                            // Add ardublock kit
                            final URI defaultKitModelURI = URI
                                    .createPlatformPluginURI(
                                            "/fr.obeo.dsl.arduino.design/resources/ArdublockKit.arduino",
                                            true);
                            session.addSemanticResource(defaultKitModelURI,
                                    monitor);

                            session.save(monitor);
                        }
                    });
```

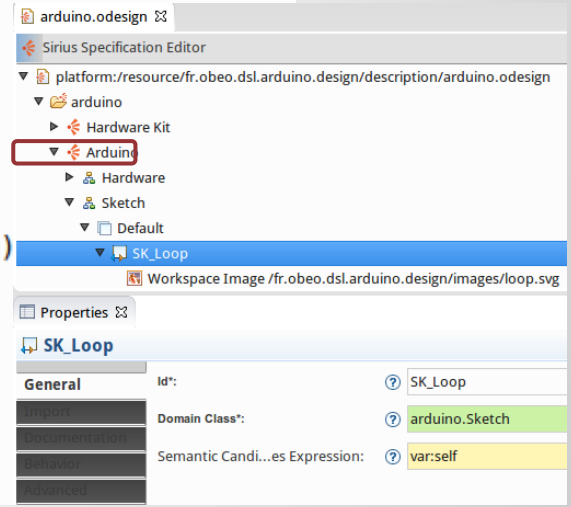


```java
private void initSemanticModel(final Session session,
        final String semanticModelPath, final IProgressMonitor monitor) {
    session.getTransactionalEditingDomain()
            .getCommandStack()
            .execute(
                    new RecordingCommand(session
                            .getTransactionalEditingDomain()) {
                        @Override
                        protected void doExecute() {

                            final URI semanticModelURI = URI
                                    .createPlatformResourceURI(
                                            semanticModelPath, true);
                            Resource res = new ResourceSetImpl()
                                    .createResource(semanticModelURI);
                            // Add the initial model object to the contents.
                            final Project rootObject = ArduinoFactory.eINSTANCE
                                    .createProject();

                            if (rootObject != null) {
                                res.getContents().add(rootObject);
                                final Hardware hardware = ArduinoFactory.eINSTANCE
                                        .createHardware();
                                hardware.setName("Hardware");
                                rootObject.setHardware(hardware);
                                final Sketch sketch = ArduinoFactory.eINSTANCE
                                        .createSketch();
                                sketch.setName("Sketch");
                                sketch.setHardware(hardware);
                                rootObject.setSketch(sketch);
                            }
                            try {
                                res.save(Maps.newHashMap());
                            } catch (IOException e) {
                                ArduinoUiActivator.log(Status.ERROR,
                                        "Init semantic model failed", e);
                            }

                            session.addSemanticResource(semanticModelURI,
                                    monitor);

                            // Add ardublock kit
                            final URI defaultKitModelURI = URI
                                    .createPlatformPluginURI(
                                            "/fr.obeo.dsl.arduino.design/resources/ArdublockKit.arduino",
                                            true);
                            session.addSemanticResource(defaultKitModelURI,
                                    monitor);

                            session.save(monitor);
                        }
                    });
```



Enable viewpoints

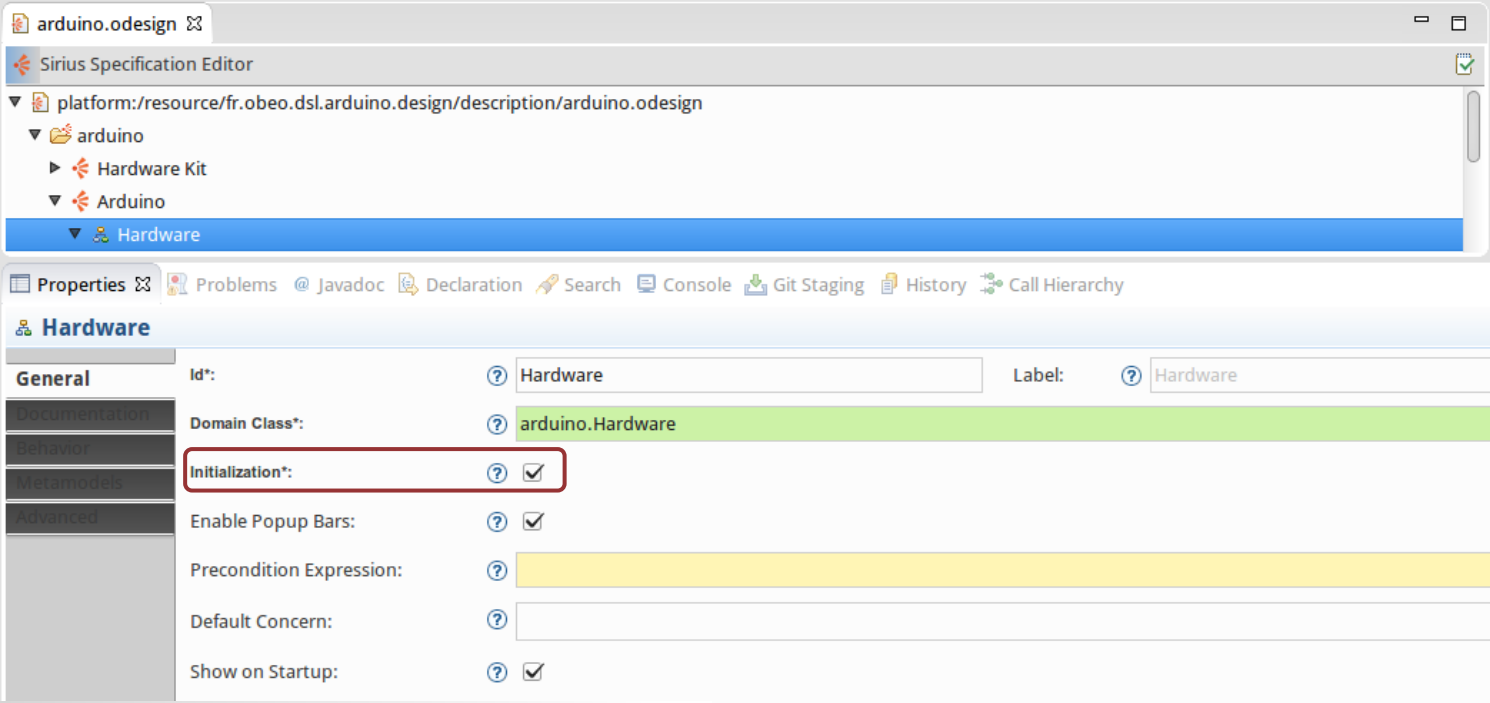
```
public void createProject(IProgressMonitor monitor, IProject project) {
    try {
        project.create(monitor);
        project.open(monitor);
        ModelingProjectManager.INSTANCE.convertToModelingProject(project, monitor);
    } catch (CoreException e) {
        ArduinoUiActivator.log(Status.ERROR, "Open project failed", e);
    }

    String modelPath = '/' + project.getName(); //$NON-NLS-1$
    final Session session = createAird(
            project,
            URI.createPlatformResourceURI(modelPath
                    + "/representations.aird", true), monitor);

    final String semanticModelPath = getSemanticModelPath(session);
    initSemanticModel(session, semanticModelPath, monitor);

    final String[] viewpointsToActivate = { ARDUINO_VP };
    enableViewpoints(session, viewpointsToActivate);

    openHardware(session);
}
```



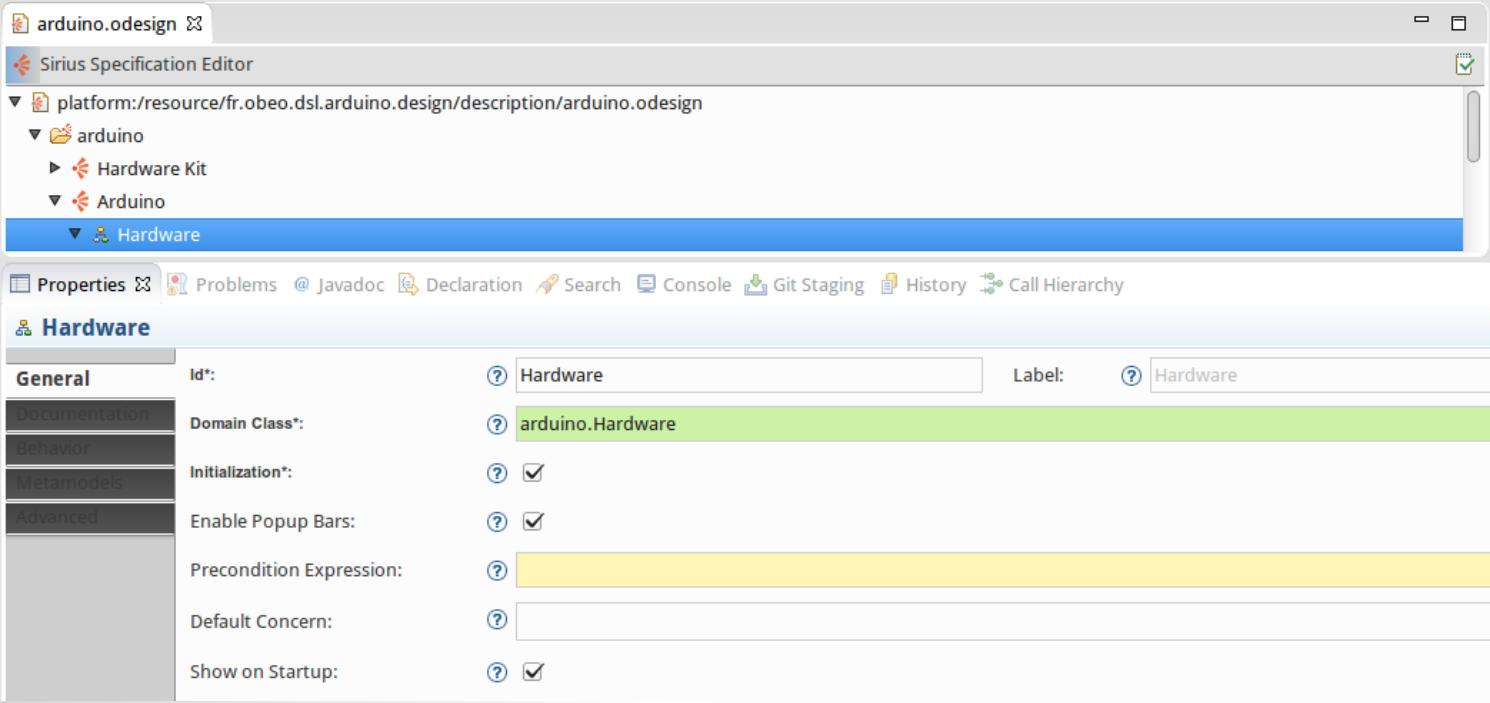
```
public static void enableViewpoints(final Session session,
        final String... viewpointsToActivate) {
    if (session != null) {
        session.getTransactionalEditingDomain()
                .getCommandStack()
                .execute(
                        new RecordingCommand(session
                                .getTransactionalEditingDomain()) {
                            @Override
                            protected void doExecute() {
                                ViewpointSelectionCallback callback = new ViewpointSelectionCallback();

                                for (Viewpoint vp : ViewpointRegistry
                                        .getInstance().getViewpoints()) {
                                    for (String viewpoint : viewpointsToActivate) {
                                        if (viewpoint.equals(vp.getName()))
                                            callback.selectViewpoint(
                                                    vp,
                                                    session,
                                                    new NullProgressMonitor());
                                    }
                                }
                            }
                        });
    }
}
```

Open a diagram

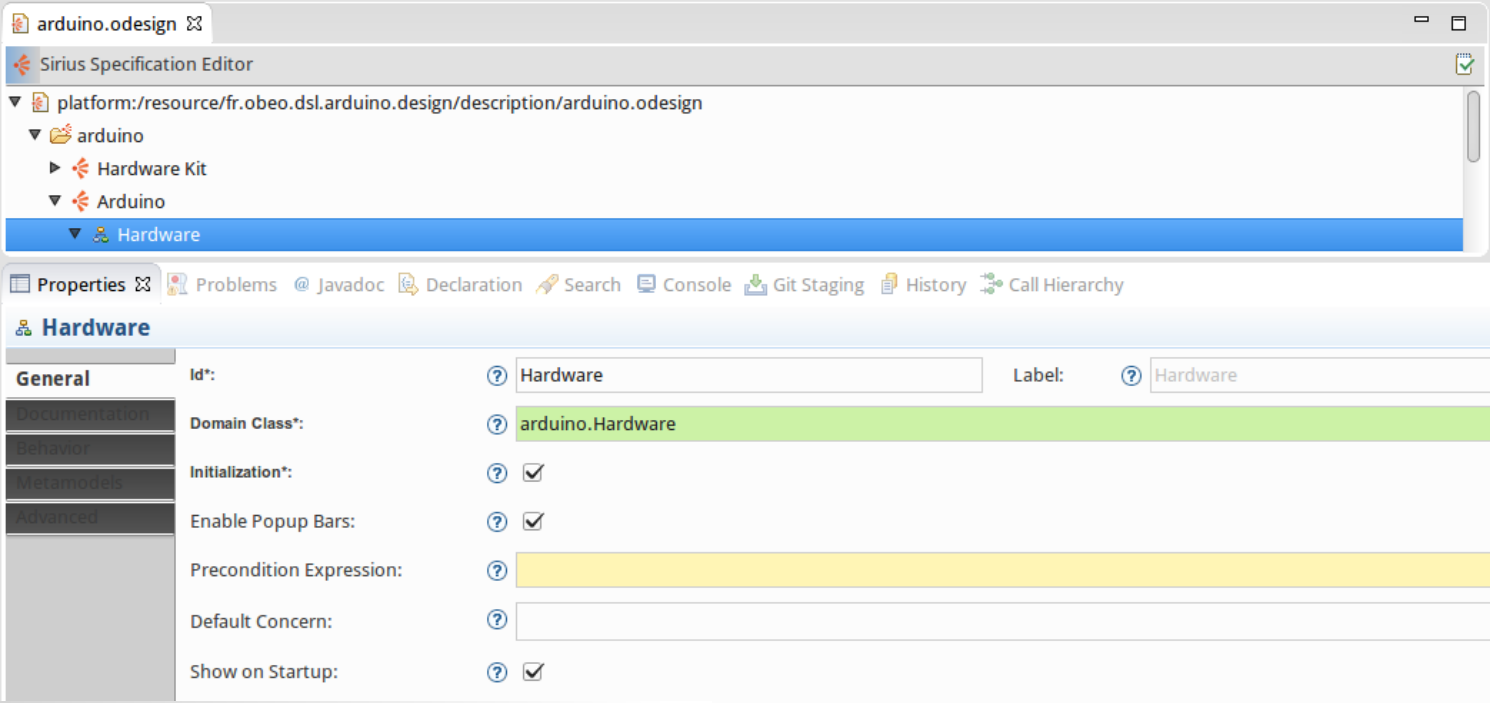
```
public void createProject(IProgressMonitor monitor, IProject project) {
    try {
        project.create(monitor);
        project.open(monitor);
        ModelingProjectManager.INSTANCE.convertToModelingProject(project, monitor);
    } catch (CoreException e) {
        ArduinoUiActivator.log(Status.ERROR, "Open project failed", e);
    }

    String modelPath = '/' + project.getName(); //$NON-NLS-1$
    final Session session = createAird(
            project,
            URI.createPlatformResourceURI(modelPath
                    + "/representations.aird", true), monitor);

    final String semanticModelPath = getSemanticModelPath(session);
    initSemanticModel(session, semanticModelPath, monitor);

    final String[] viewpointsToActivate = { ARDUINO_VP };
    enableViewpoints(session, viewpointsToActivate);

    openHardware(session);
}
```



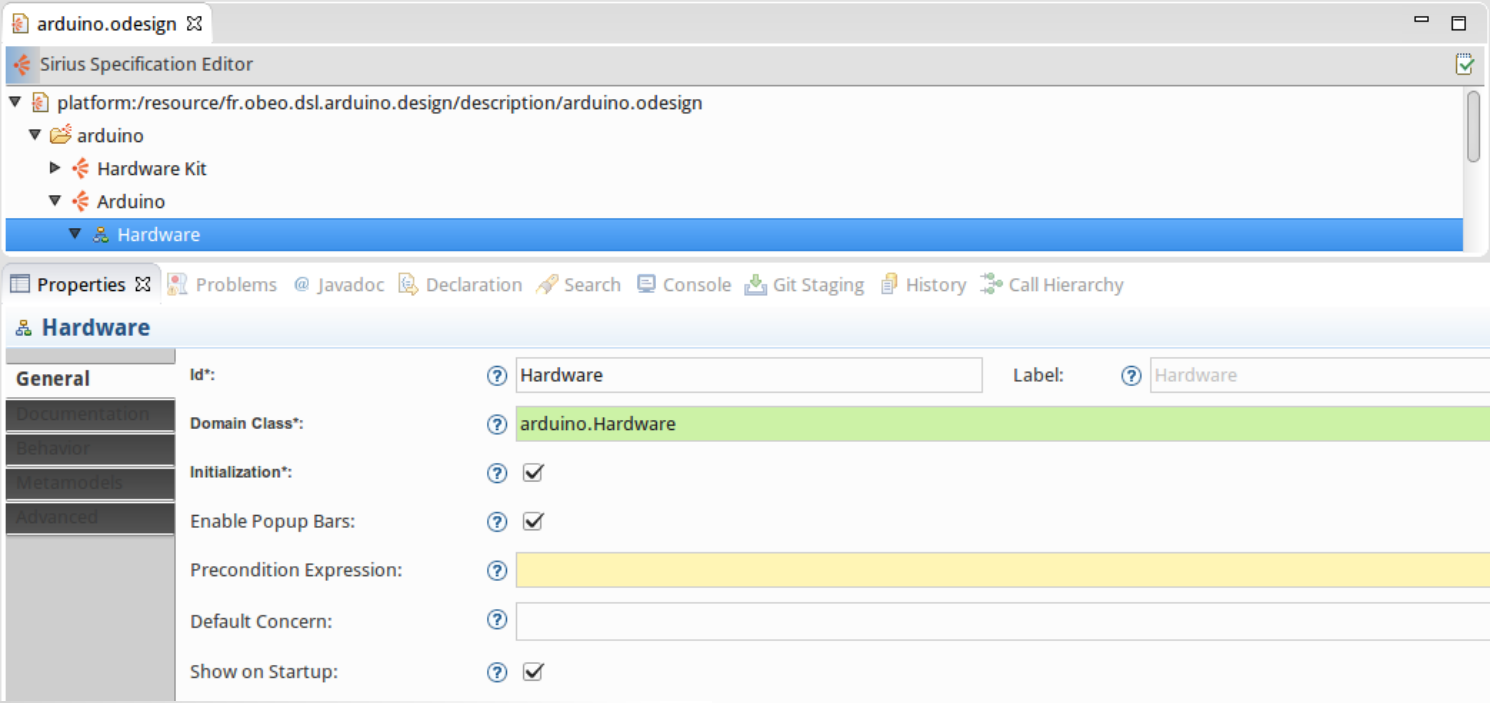

```java
public void openHardware(final Session session) {
    Collection<DRepresentation> representations = DialectManager.INSTANCE
            .getAllRepresentations(session);
    for (DRepresentation representation : representations) {
        if ("Hardware".equals(representation.getName())) {
            DialectUIManager.INSTANCE.openEditor(session, representation,
                    new NullProgressMonitor());
            return;
        }
    }
}
```




```
public void openHardware(final Session session) {
    Collection<DRepresentation> representations = DialectManager.INSTANCE
            .getAllRepresentations(session);
    for (DRepresentation representation : representations) {
        if ("Hardware".equals(representation.getName())) {
            DialectUIManager.INSTANCE.openEditor(session, representation,
                    new NullProgressMonitor());
            return;
        }
    }
}
```



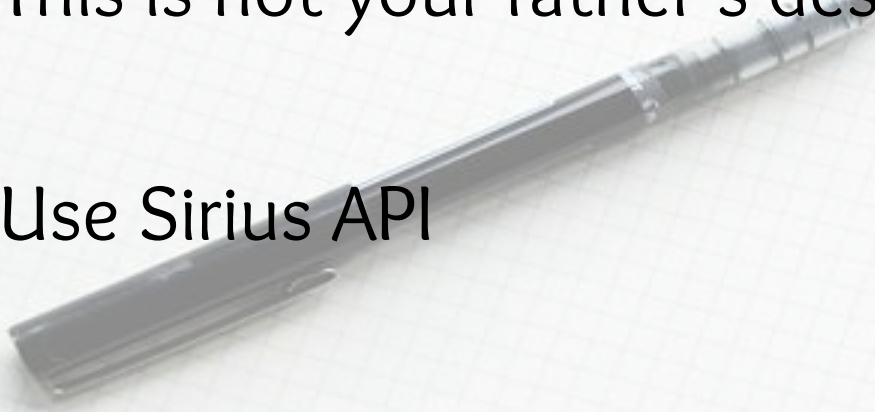

```java
public void openHardware(final Session session) {
    Collection<DRepresentation> representations = DialectManager.INSTANCE
            .getAllRepresentations(session);
    for (DRepresentation representation : representations) {
        if ("Hardware".equals(representation.getName())) {
            DialectUIManager.INSTANCE.openEditor(session, representation,
                    new NullProgressMonitor());
            return;
        }
    }
}
```




```java
public void openHardware(final Session session) {
    Collection<DRepresentation> representations = DialectManager.INSTANCE
            .getAllRepresentations(session);
    for (DRepresentation representation : representations) {
        if ("Hardware".equals(representation.getName())) {
            DialectUIManager.INSTANCE.openEditor(session, representation,
                    new NullProgressMonitor());
            return;
        }
    }
}
```

Remind #3

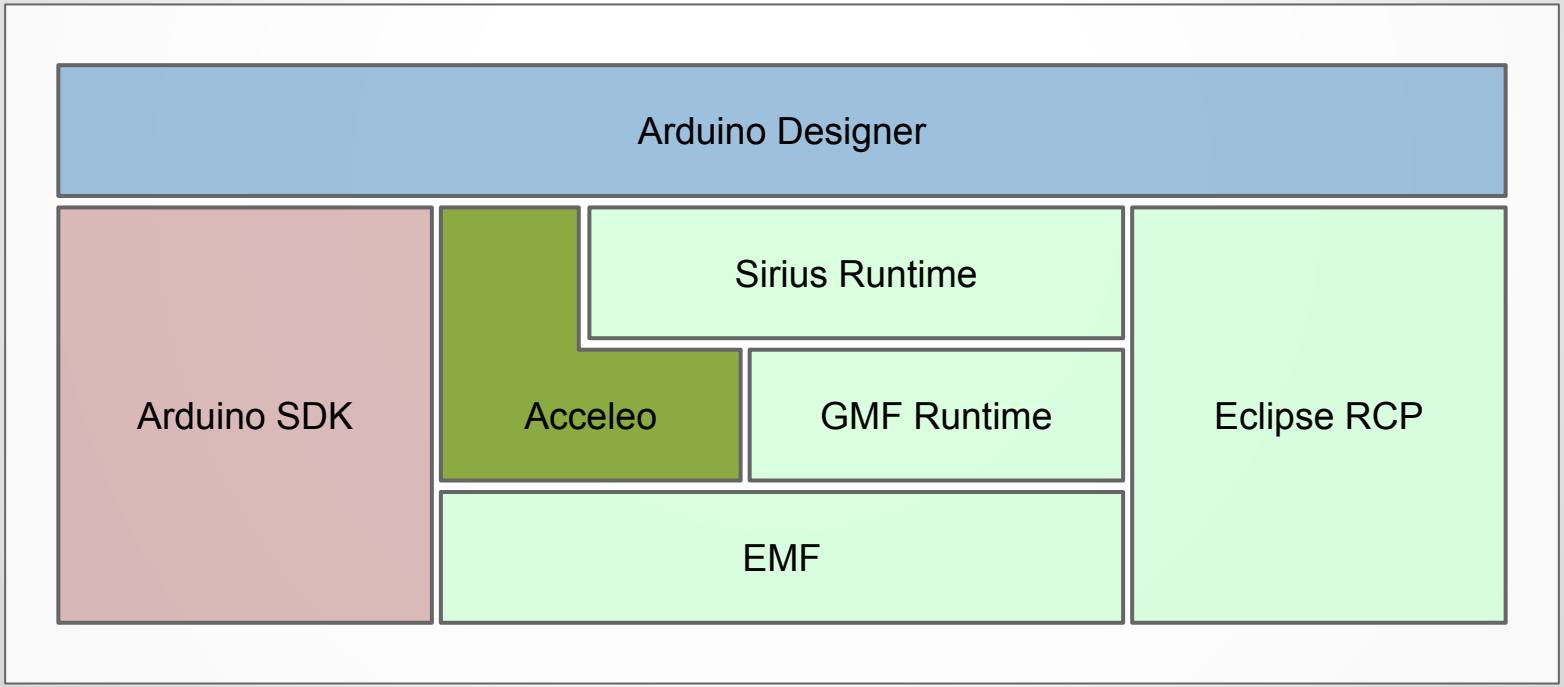
1. This is not your father's designer
2. Use Sirius API



Integrate code generator

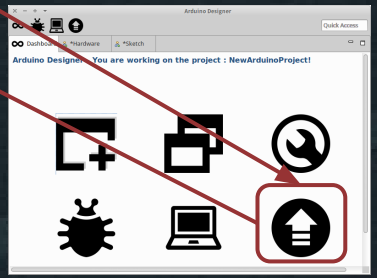
Generate ino file with Acceleo

Integrate code generator



```
string sInput;
int iLength, iN;
double dblTemp;
bool again = true

while (again) {
    iN = -1;
    again = false
    getline(cin,
    system("cls")
    stringstream(
    iLength = sIn
    if (iLength <
        again = t
        continue;
    } else if (si
        again = t
        continue;
    } while (++i
        if (isdi
            cont
    } else i
```

```
public void upload(final Sketch sketch) {
    if (preferences.getArduinoSdk() == null
            || preferences.getArduinoSdk().length() == 0) {
        askUser();
        return;
    }
    final ProgressMonitorDialog dialog = new ProgressMonitorDialog(
            PlatformUI.getWorkbench().getActiveWorkbenchWindow().getShell());
    try {
        dialog.run(true, true, new IRunnableWithProgress() {
            @Override
            public void run(IProgressMonitor monitor) {
                monitor.beginTask("Upload sketch to arduino platform...",
                        100);
                monitor.subTask("Generate code");
                File genFolder = generateCode(sketch);

                monitor.worked(33);
                monitor.subTask("Compile code");
                String arduinoSdk = preferences.getArduinoSdk();
                String serialPort = preferences.getArduinoSerialPort();
                String boardTag = sketch.getHardware().getPlatforms()
                        .get(0).getName();
                String workingDirectory = genFolder.toString();
                ArduinoBuilder builder = new ArduinoBuilder(arduinoSdk,
                        boardTag, workingDirectory, serialPort);
                List<String> libraries = getLibraries(sketch);
                final IStatus compileStatus = builder.compile("Sketch",
                        libraries);
                if (compileStatus.getSeverity() != IStatus.OK) {
                    Display.getDefault().syncExec(new Runnable() {
                        public void run() {
                            MessageDialog.openError(
                                    dialog.getShell(),
                                    "Compilation Fail",
                                    "Compilation fail : "
                                            + compileStatus.getMessage());
                        }
                    });
                    return;
                }

                monitor.worked(33);
                monitor.subTask("Upload code");
                final IStatus uploadStatus = builder.upload();
                if (uploadStatus.getSeverity() != IStatus.OK) {
                    Display.getDefault().syncExec(new Runnable() {
                        public void run() {
                            MessageDialog.openError(dialog.getShell(),
                                    "Upload Fail", "Upload fail : "
                                            + uploadStatus.getMessage());
                        }
```

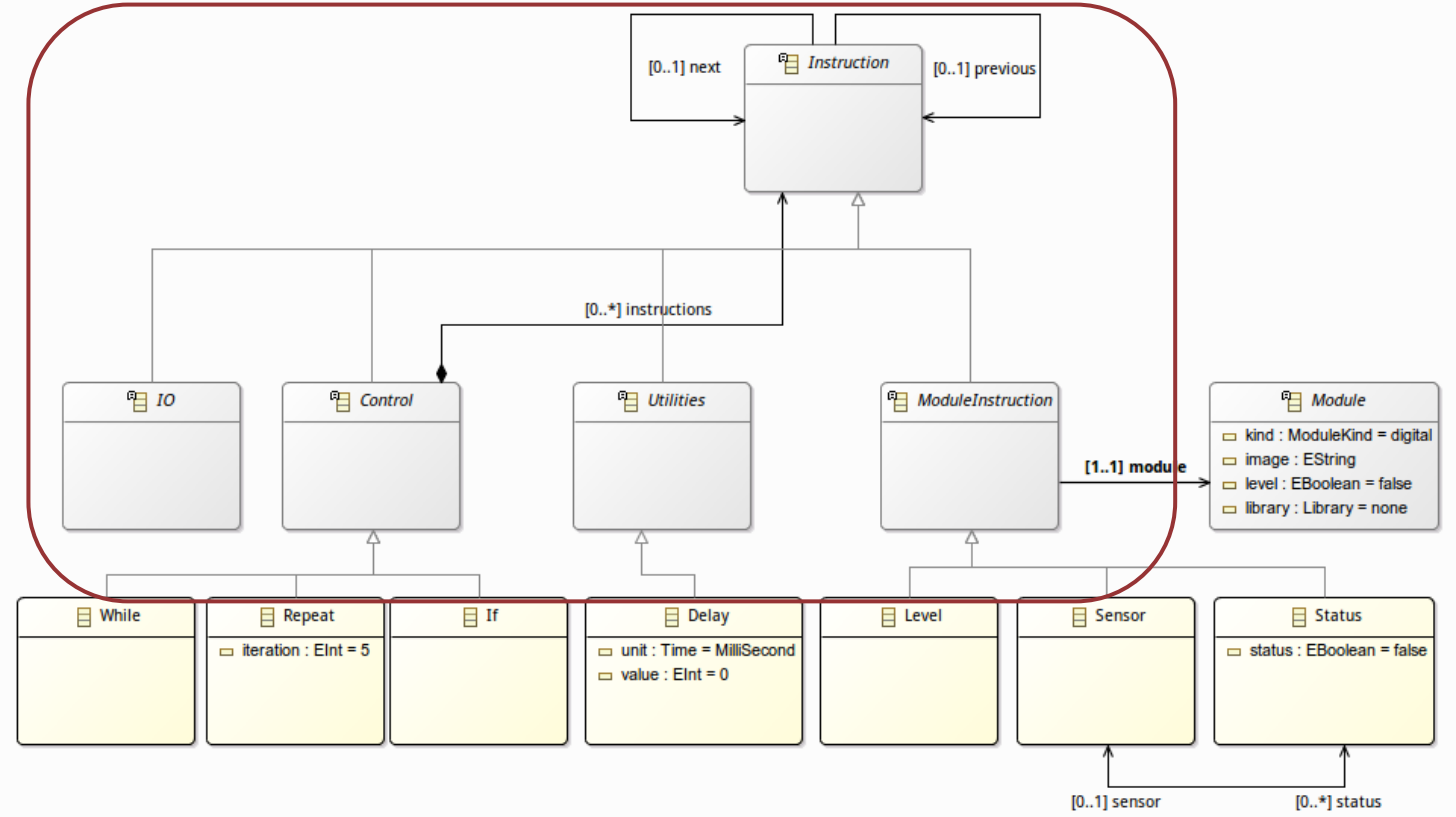


```
private File generateCode(Sketch sketch) {
    boolean oldNotificationsPref = AcceleoPreferences
            .areNotificationsForcedDisabled();
    AcceleoPreferences.switchForceDeactivationNotifications(true);
    IFile file = ResourcesPlugin
            .getWorkspace()
            .getRoot()
            .getFile(
                    new Path(sketch.eResource().getURI()
                            .toPlatformString(true)));
    IFolder folder = file.getProject().getFolder("code");
    File genFolder = folder.getRawLocation().makeAbsolute().toFile();

    try {
        Generate generator = new Generate(sketch.eResource().getURI(),
                genFolder, new ArrayList<Object>());
        generator.doGenerate(new BasicMonitor());
    } catch (IOException e) {
        ArduinoUiActivator.log(Status.ERROR, "Code generation failed", e);
    }
    AcceleoPreferences
            .switchForceDeactivationNotifications(oldNotificationsPref);

    return genFolder;
}
```

generate.mtl

```
[comment encoding = UTF-8 /]
[module generate('http://www.obeo.fr/arduino')]
[import fr::obeo::dsl::arduino::gen::main::arduinoservices /]

[template public generateSketch(sketch : Sketch)]
[comment @main/]
[genIno()/]
[genMakefile()/]
[/template]

[template public genIno (sketch : Sketch) ]
[file (sketch.getVariableName()+'.ino', false, 'UTF-8')]
[for (library : Library | sketch.eAllContents("Level").module->select(library<>arduino::Library::none).libr
#include <[library.toString().toUpperFirst()/].h>
[/for]
[for (library : Library | sketch.eAllContents("Status").module->select(library<>arduino::Library::none).lib
#include <[library.toString().toUpperFirst()/].h>
[/for]
[for (mod : Module | sketch.eAllContents("Status").module->asOrderedSet())]
[if (mod.library<>arduino::Library::music)]
int [mod.getVariableName()/] = [sketch.getPinId(mod)/];
[/if]
[/for]
[for (mod : Module | sketch.eAllContents("Level").module->select(library<>arduino::Library::none)->asOrdere
[mod.library.toString().toUpperFirst()/] [mod.getVariableName()/];
[/for]
[for (mod : Module | sketch.eAllContents("Sensor").module->asOrderedSet())]
int [mod.getVariableName()/] = [sketch.getPinId(mod)/];
[/for]
[for (instruction : Repeat | getRepeatInstructions(sketch))]
int iter_[eContainer(arduino::Sketch).getRepeatInstructionIndex(instruction)/];
[/for]
[for (instruction : Variable | sketch.eAllContents("Variable"))]
int [instruction.name/];
[/for]

void setup() {
[if (sketch.eAllContents("Status").module.library->select(lib|lib = arduino::Library::music)->asOrderedSet(
 music.init();
[/if]
[for (mod : Module | sketch.eAllContents("Status").module->asOrderedSet())]
[if (mod.library<>arduino::Library::music)]
  pinMode([mod.getVariableName()/], OUTPUT);
[/if]
[/for]
[for (mod : Module | sketch.eAllContents("Level").module->select(library<>arduino::Library::none)->asOrdere
 [mod.getVariableName()/].attach([sketch.getPinId(mod)/]);
[/for]
[for (instruction : Variable | sketch.eAllContents("Variable"))]
  [instruction.name/]=0;
[/for]
}
```

All is instruction!

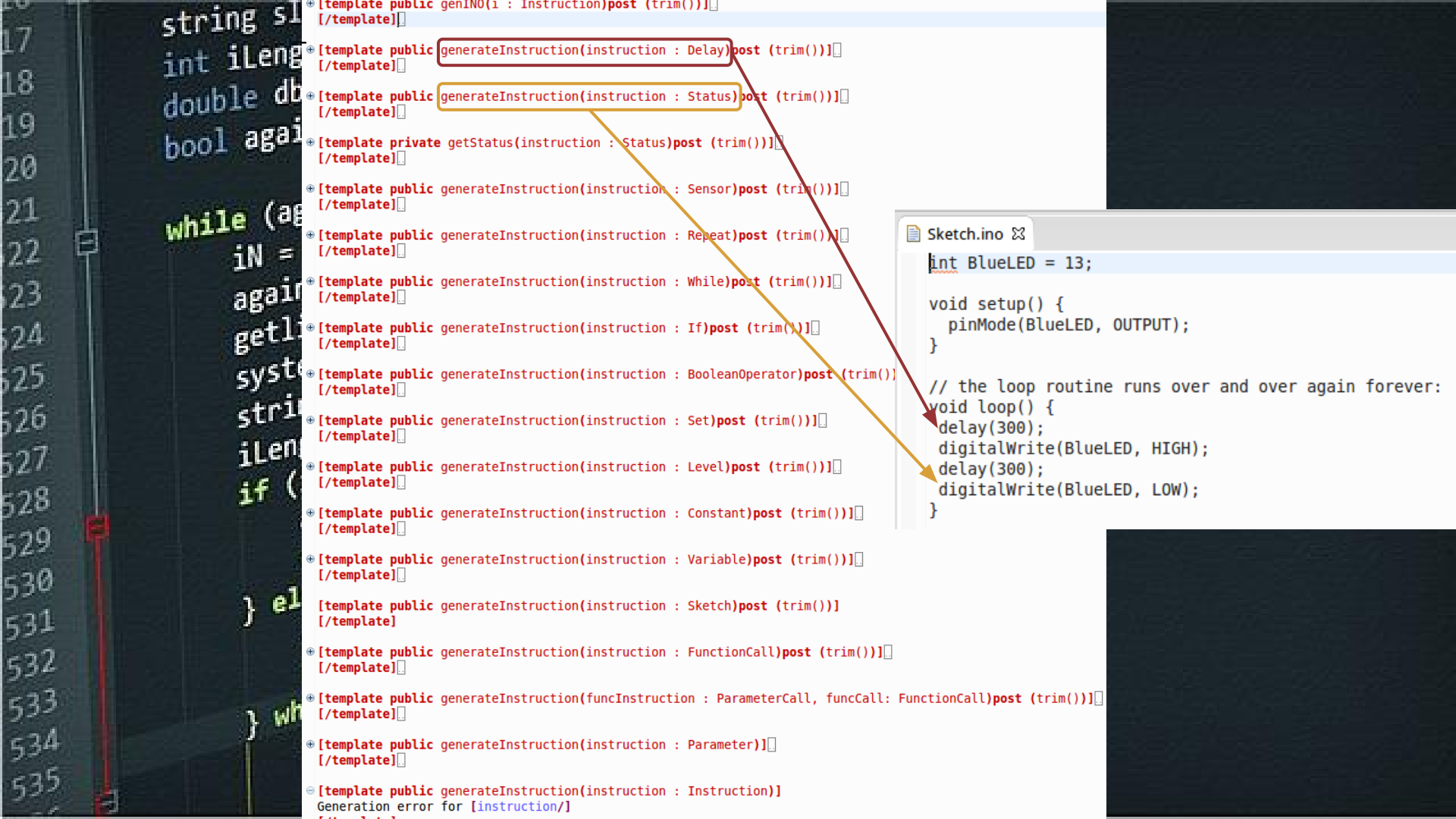


All is instruction!

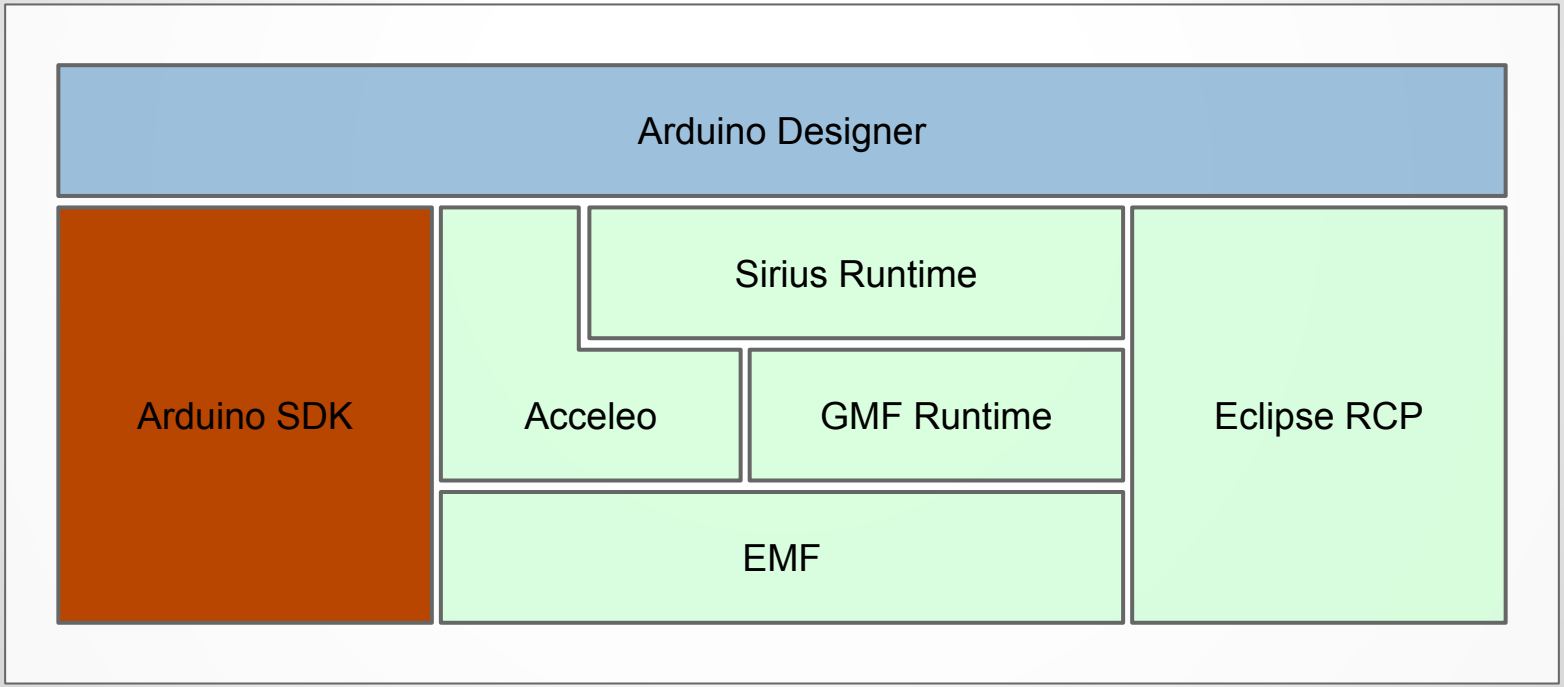
Use

inheritance

for the templates!


[template public genINO(i : Instruction)post (trim())]
[/template]
[template public generateInstruction(instruction : Delay)post (trim())]
[/template]
[template public generateInstruction(instruction : Status)post (trim())]
[/template]
[template private getStatus(instruction : Status)post (trim())]
[/template]
[template public generateInstruction(instruction : Sensor)post (trim())]
[/template]
[template public generateInstruction(instruction : Repeat)post (trim())]
[/template]
[template public generateInstruction(instruction : While)post (trim())]
[/template]
[template public generateInstruction(instruction : If)post (trim())]
[/template]
[template public generateInstruction(instruction : BooleanOperator)post (trim())]
[/template]
[template public generateInstruction(instruction : Set)post (trim())]
[/template]
[template public generateInstruction(instruction : Level)post (trim())]
[/template]
[template public generateInstruction(instruction : Constant)post (trim())]
[/template]
[template public generateInstruction(instruction : Variable)post (trim())]
[/template]
[template public generateInstruction(instruction : Sketch)post (trim())]
[/template]
[template public generateInstruction(instruction : FunctionCall)post (trim())]
[/template]
[template public generateInstruction(funcInstruction : ParameterCall, funcCall: FunctionCall)post (trim())]
[/template]
[template public generateInstruction(instruction : Parameter)]
[/template]
[template public generateInstruction(instruction : Instruction)]
Generation error for [instruction/]
Sketch.ino
int BlueLED = 13;
void setup() {
pinMode(BlueLED, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop() {
delay(300);
digitalWrite(BlueLED, HIGH);
delay(300);
digitalWrite(BlueLED, LOW);
}

Combine with Arduino tools

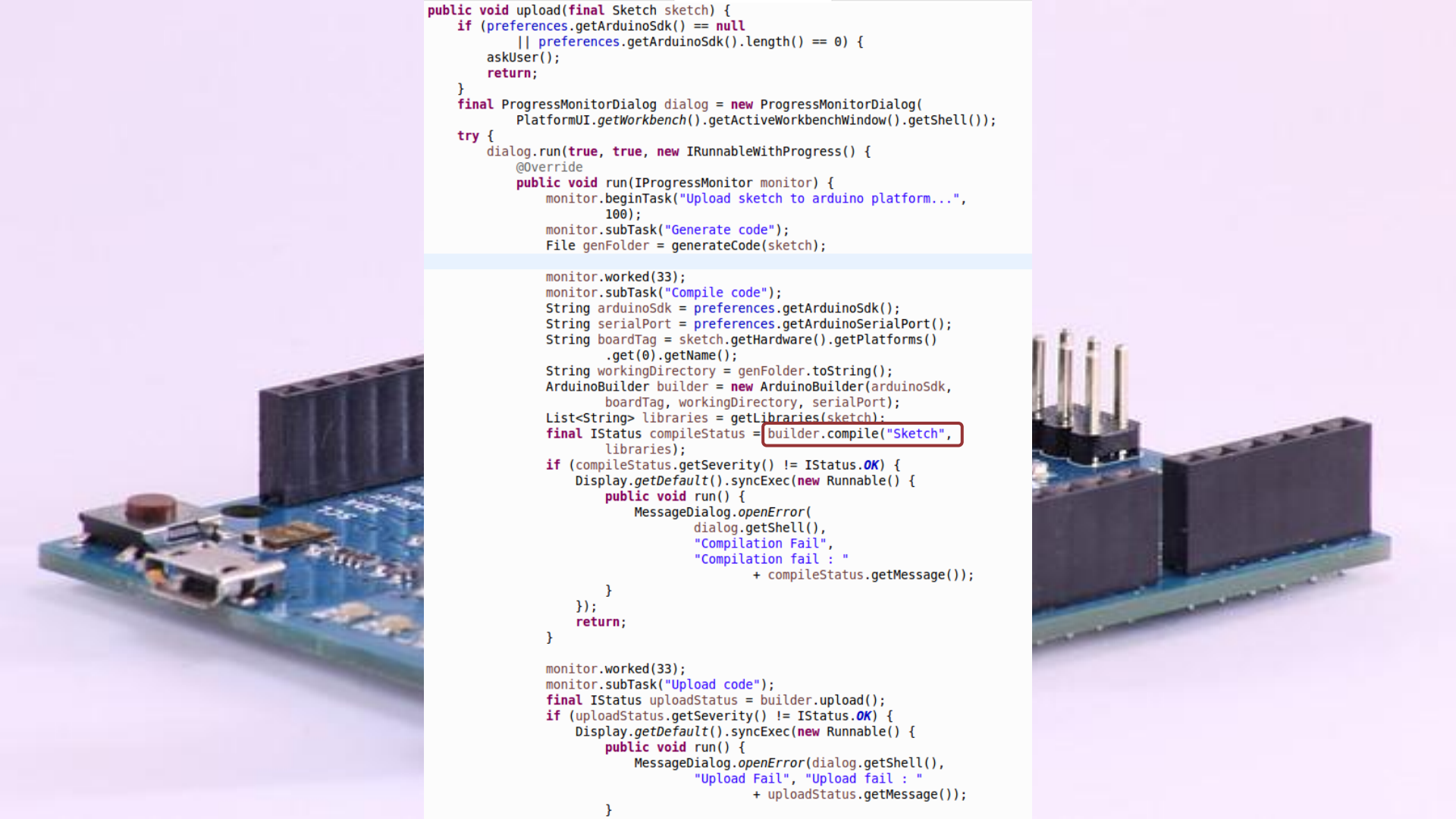


Combine with Arduino tools

Arduino compiler

Target uploader

Based on Arduino IDE to get well packaged tools : avr-gcc, avrdude

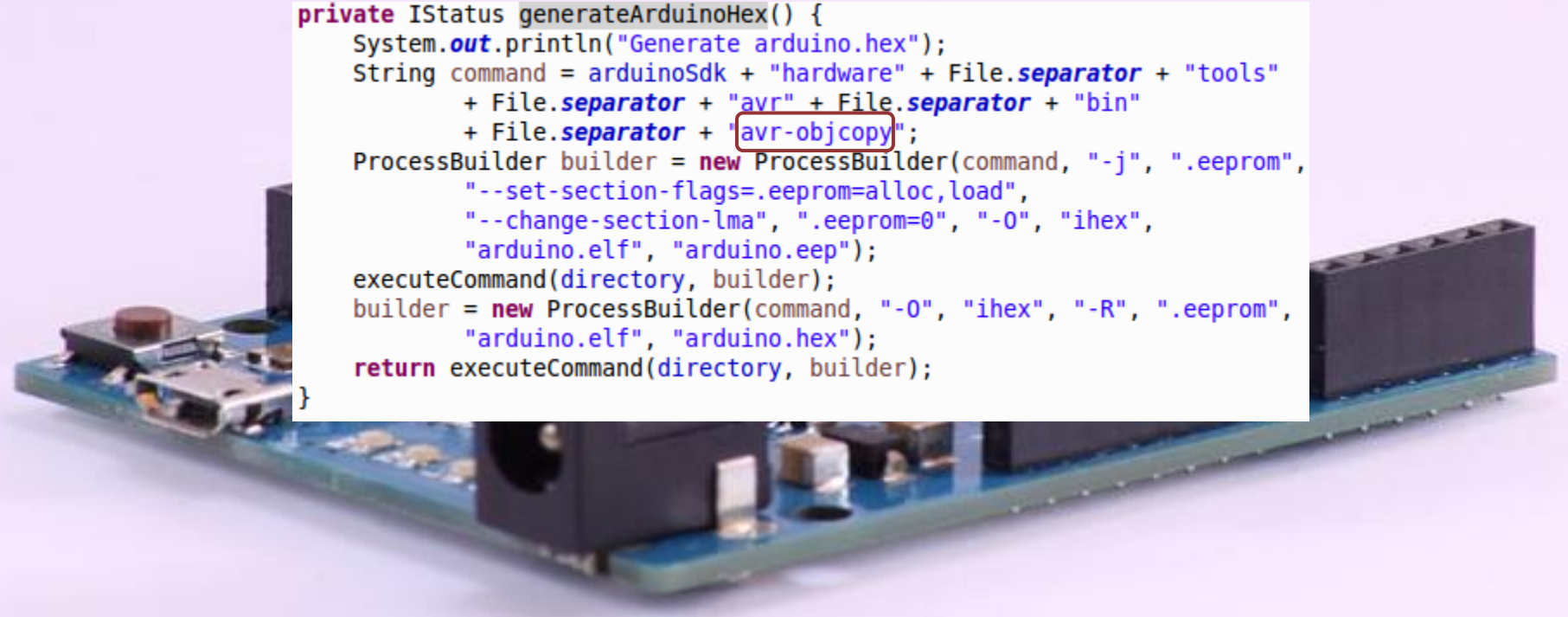


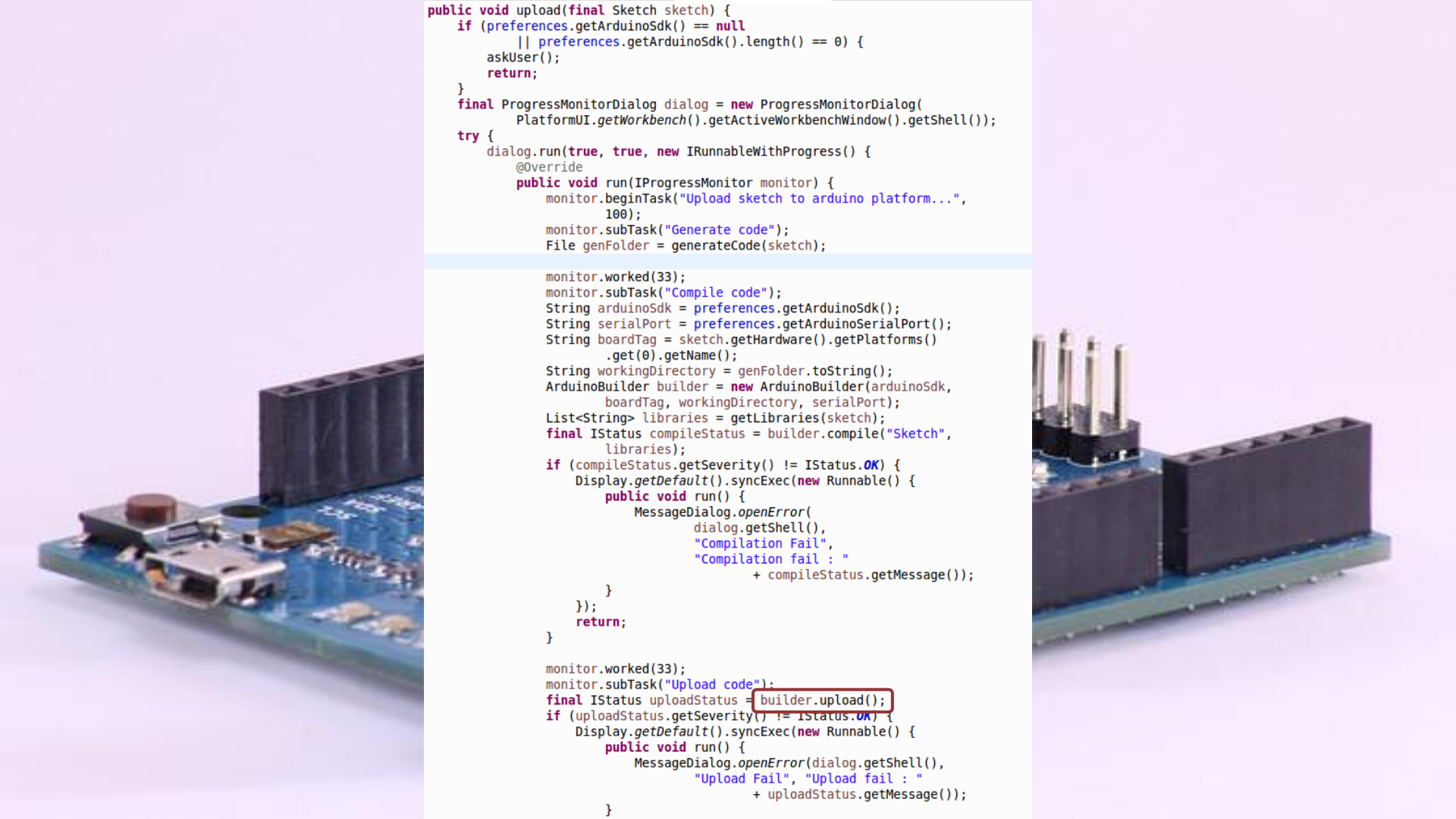
```
public void upload(final Sketch sketch) {
    if (preferences.getArduinoSdk() == null
            || preferences.getArduinoSdk().length() == 0) {
        askUser();
        return;
    }
    final ProgressMonitorDialog dialog = new ProgressMonitorDialog(
            PlatformUI.getWorkbench().getActiveWorkbenchWindow().getShell());
    try {
        dialog.run(true, true, new IRunnableWithProgress() {
            @Override
            public void run(IProgressMonitor monitor) {
                monitor.beginTask("Upload sketch to arduino platform...",
                        100);
                monitor.subTask("Generate code");
                File genFolder = generateCode(sketch);

                monitor.worked(33);
                monitor.subTask("Compile code");
                String arduinoSdk = preferences.getArduinoSdk();
                String serialPort = preferences.getArduinoSerialPort();
                String boardTag = sketch.getHardware().getPlatforms()
                        .get(0).getName();
                String workingDirectory = genFolder.toString();
                ArduinoBuilder builder = new ArduinoBuilder(arduinoSdk,
                        boardTag, workingDirectory, serialPort);
                List<String> libraries = getLibraries(sketch);
                final IStatus compileStatus = builder.compile("Sketch",
                        libraries);
                if (compileStatus.getSeverity() != IStatus.OK) {
                    Display.getDefault().syncExec(new Runnable() {
                        public void run() {
                            MessageDialog.openError(
                                    dialog.getShell(),
                                    "Compilation Fail",
                                    "Compilation fail : "
                                            + compileStatus.getMessage());
                        }
                    });
                    return;
                }

                monitor.worked(33);
                monitor.subTask("Upload code");
                final IStatus uploadStatus = builder.upload();
                if (uploadStatus.getSeverity() != IStatus.OK) {
                    Display.getDefault().syncExec(new Runnable() {
                        public void run() {
                            MessageDialog.openError(dialog.getShell(),
                                    "Upload Fail", "Upload fail : "
                                            + uploadStatus.getMessage());
                        }
```

```
public IStatus compile(String sketchName, List<String> libraries) {

    // Compile sketch
    IStatus sketchStatus = compileSketch(sketchName, libraries);
    if (sketchStatus.getSeverity() != IStatus.OK) {
        return sketchStatus;
    }

    // Compile main libraries
    IStatus mainLibrariesStatus = compileMainLibraries();
    if (mainLibrariesStatus.getSeverity() != IStatus.OK) {
        return mainLibrariesStatus;
    }

    // Compile specific libraries
    if (libraries != null) {
        for (String library : libraries) {
            String libraryName = Character.toUpperCase(library.charAt(0))
                    + library.substring(1);
            IStatus specificLibraryStatus = compileSpecificLibrary(
                    arduinoSdk + "libraries" + File.separator + libraryName
                            + File.separator, libraryName);
            if (specificLibraryStatus.getSeverity() != IStatus.OK) {
                return specificLibraryStatus;
            }
        }
    }

    // Link all
    IStatus linkStatus = link(sketchName, libraries);
    if (linkStatus.getSeverity() != IStatus.OK) {
        return linkStatus;
    }

    // Generate arduino.hex
    return generateArduinoHex();
}
```

```
private IStatus compileSketch(String fileName, List<String> libraries) {
    System.out.println("Compile Sketch");
    String command = arduinoSdk + "hardware" + File.separator + "tools"
            + File.separator + "avr" + File.separator + "bin"
            + File.separator + "avr-g++";

    if (os.contains(WINDOWS)) {
        command += ".exe";
    }
    List<String> commands = Lists.newArrayList();
    commands.add(command);
    commands.add("-x");
    commands.add("c++");
    commands.add("-include");
    commands.add("Arduino.h");
    commands.add("-c");
    commands.add("-mmcu=" + getMMCU());
    commands.add("-DF_CPU=" + getDFCPU());
    commands.add("-DARDUINO=" + getDArduino());
    commands.add("-I.");
    commands.add("-I" + arduinoSdk + "hardware" + File.separator
            + "arduino" + File.separator + "cores" + File.separator
            + "arduino");
    commands.add("-I" + arduinoSdk + "hardware" + File.separator
            + "arduino" + File.separator + "variants" + File.separator
            + "standard");
    for (String library : libraries) {
        String libraryName = Character.toUpperCase(library.charAt(0))
                + library.substring(1);
        commands.add("-I" + arduinoSdk + "libraries" + File.separator
                + libraryName);
    }
    commands.add("-g");
    commands.add("-Os");
    commands.add("-Wall");
    commands.add("-ffunction-sections");
    commands.add("-fdata-sections");
    commands.add("-fno-exceptions");
    commands.add(fileName + ".ino");
    commands.add("-o");
    commands.add(fileName + ".o");

    ProcessBuilder builder = new ProcessBuilder(commands);

    return executeCommand(directory, builder);
}
```

```
private IStatus link(String sketchName, List<String> libraries) {
    System.out.println("Link all");
    String command = arduinoSdk + "hardware" + File.separator + "tools"
            + File.separator + "avr" + File.separator + "bin"
            + File.separator + "avr-gcc";
    List<String> commands = Lists.newArrayList();
    ProcessBuilder builder = new ProcessBuilder(commands);
    commands.add(command);
    commands.add("-mmcu=" + getMMCU());
    commands.add("-Wl,--gc-sections");
    commands.add("-Os");
    commands.add("-o");
    commands.add("arduino.elf");
    commands.add(sketchName + ".o");
    commands.add("WInterrupts.o");
    commands.add("wiring_analog.o");
    commands.add("wiring.o");
    commands.add("wiring_digital.o");
    commands.add("wiring_pulse.o");
    commands.add("wiring_shift.o");
    commands.add("CDC.o");
    commands.add("HardwareSerial.o");
    commands.add("HID.o");
    commands.add("IPAddress.o");
    commands.add("main.o");
    commands.add("new.o");
    commands.add("Print.o");
    commands.add("Stream.o");
    commands.add("Tone.o");
    commands.add("USBCore.o");
    commands.add("WMath.o");
    commands.add("WString.o");
    for (String library : libraries) {
        String libraryName = Character.toUpperCase(library.charAt(0))
                + library.substring(1);
        commands.add(libraryName + ".o");
    }
    commands.add("-lc");
    commands.add("-lm");
    return executeCommand(directory, builder);
}
```


private IStatus generateArduinoHex() {
System.out.println("Generate arduino.hex");
String command = arduinoSdk + "hardware" + File.separator + "tools"
+ File.separator + "avr" + File.separator + "bin"
+ File.separator + "avr-objcopy";
ProcessBuilder builder = new ProcessBuilder(command, "-j", ".eeprom",
"--set-section-flags=.eeprom=alloc,load",
"--change-section-lma", ".eeprom=0", "-O", "ihex",
"arduino.elf", "arduino.eep");
executeCommand(directory, builder);
builder = new ProcessBuilder(command, "-O", "ihex", "-R", ".eeprom",
"arduino.elf", "arduino.hex");
return executeCommand(directory, builder);
}



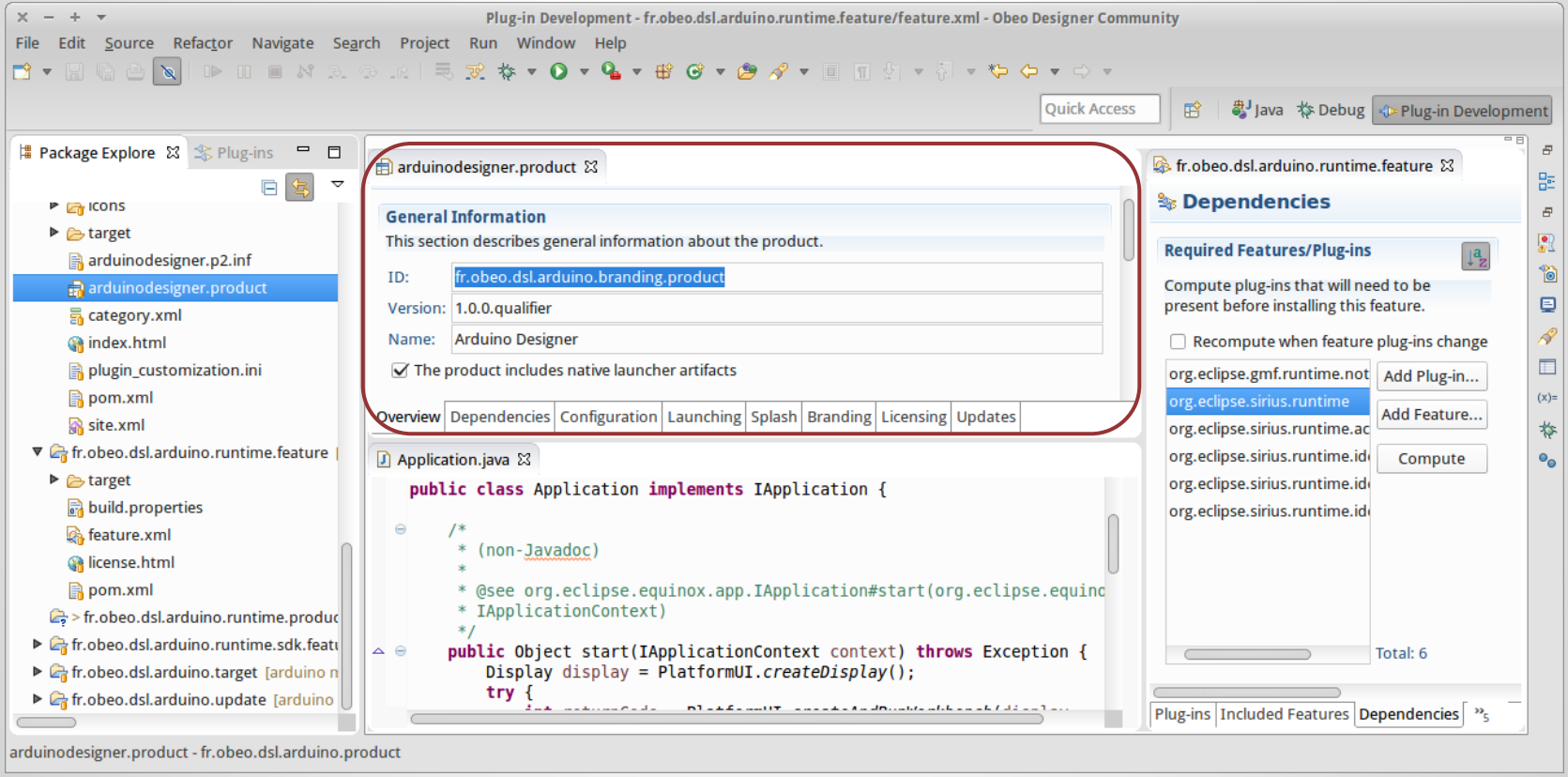
```
public void upload(final Sketch sketch) {
    if (preferences.getArduinoSdk() == null
            || preferences.getArduinoSdk().length() == 0) {
        askUser();
        return;
    }
    final ProgressMonitorDialog dialog = new ProgressMonitorDialog(
            PlatformUI.getWorkbench().getActiveWorkbenchWindow().getShell());
    try {
        dialog.run(true, true, new IRunnableWithProgress() {
            @Override
            public void run(IProgressMonitor monitor) {
                monitor.beginTask("Upload sketch to arduino platform...",
                        100);
                monitor.subTask("Generate code");
                File genFolder = generateCode(sketch);

                monitor.worked(33);
                monitor.subTask("Compile code");
                String arduinoSdk = preferences.getArduinoSdk();
                String serialPort = preferences.getArduinoSerialPort();
                String boardTag = sketch.getHardware().getPlatforms()
                        .get(0).getName();
                String workingDirectory = genFolder.toString();
                ArduinoBuilder builder = new ArduinoBuilder(arduinoSdk,
                        boardTag, workingDirectory, serialPort);
                List<String> libraries = getLibraries(sketch);
                final IStatus compileStatus = builder.compile("Sketch",
                        libraries);
                if (compileStatus.getSeverity() != IStatus.OK) {
                    Display.getDefault().syncExec(new Runnable() {
                        public void run() {
                            MessageDialog.openError(
                                    dialog.getShell(),
                                    "Compilation Fail",
                                    "Compilation fail : "
                                            + compileStatus.getMessage());
                        }
                    });
                    return;
                }

                monitor.worked(33);
                monitor.subTask("Upload code");
                final IStatus uploadStatus = builder.upload();
                if (uploadStatus.getSeverity() != IStatus.OK) {
                    Display.getDefault().syncExec(new Runnable() {
                        public void run() {
                            MessageDialog.openError(dialog.getShell(),
                                    "Upload Fail", "Upload fail : "
                                            + uploadStatus.getMessage());
                        }
```

```
public IStatus upload() {
    System.out.println("Upload arduino.hex");
    String command = arduinoSdk + "hardware" + File.separator + "tools"
            + File.separator;
    ProcessBuilder builder = null;
    if (os.contains(WINDOWS)) {
        command += "avr" + File.separator + "bin" + File.separator
                + "avrdude.exe";
        builder = new ProcessBuilder(command, "-q", "-V", "-p", getMMCU(),
                "-C", arduinoSdk + "hardware" + File.separator + "tools"
                        + File.separator + "avr" + File.separator + "etc"
                        + File.separator + "avrdude.conf", "-c", "arduino",
                "-b", "115200", "-P", serialPort, "-U", "flash:w:arduino.hex:i");
    } else if (os.contains(MACOS)) {
        command += "avr" + File.separator + "bin" + File.separator
                + "avrdude";
        builder = new ProcessBuilder(command, "-q", "-V", "-p", getMMCU(),
                "-C", arduinoSdk + "hardware" + File.separator + "tools"
                        + File.separator + "avr" + File.separator + "etc"
                        + File.separator + "avrdude.conf", "-c", "arduino",
                "-b", "115200", "-P", serialPort, "-U",
                "flash:w:arduino.hex:i");
    } else {
        command += "avrdude";
        builder = new ProcessBuilder(command, "-q", "-V", "-p", getMMCU(),
                "-C", arduinoSdk + "hardware" + File.separator + "tools"
                        + File.separator + "avrdude.conf", "-c", "arduino",
                "-b", "115200", "-P", serialPort, "-U",
                "flash:w:arduino.hex:i");
    }
    return executeCommand(directory, builder);
}
```

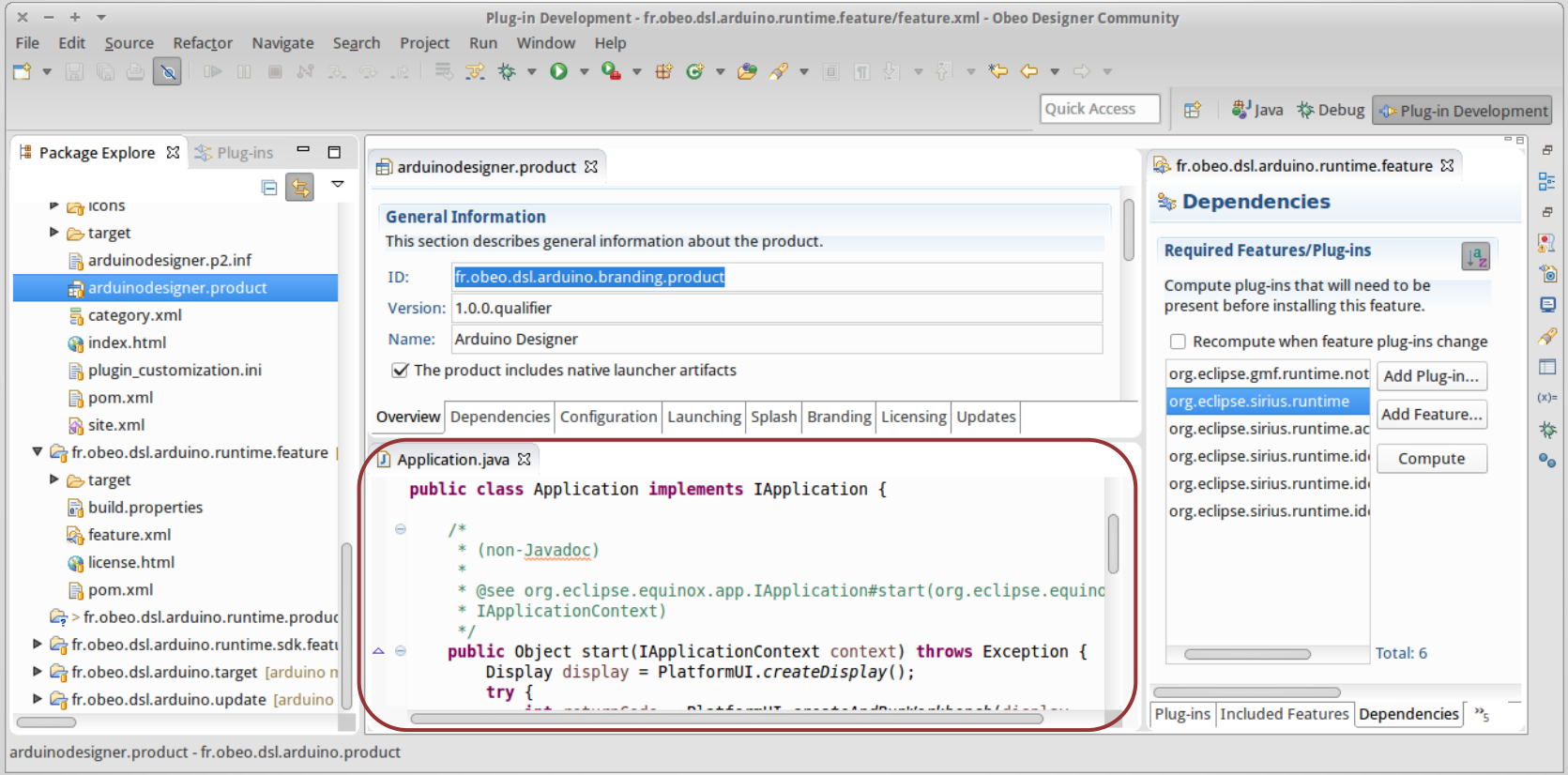
Build a product

Classical RCP product

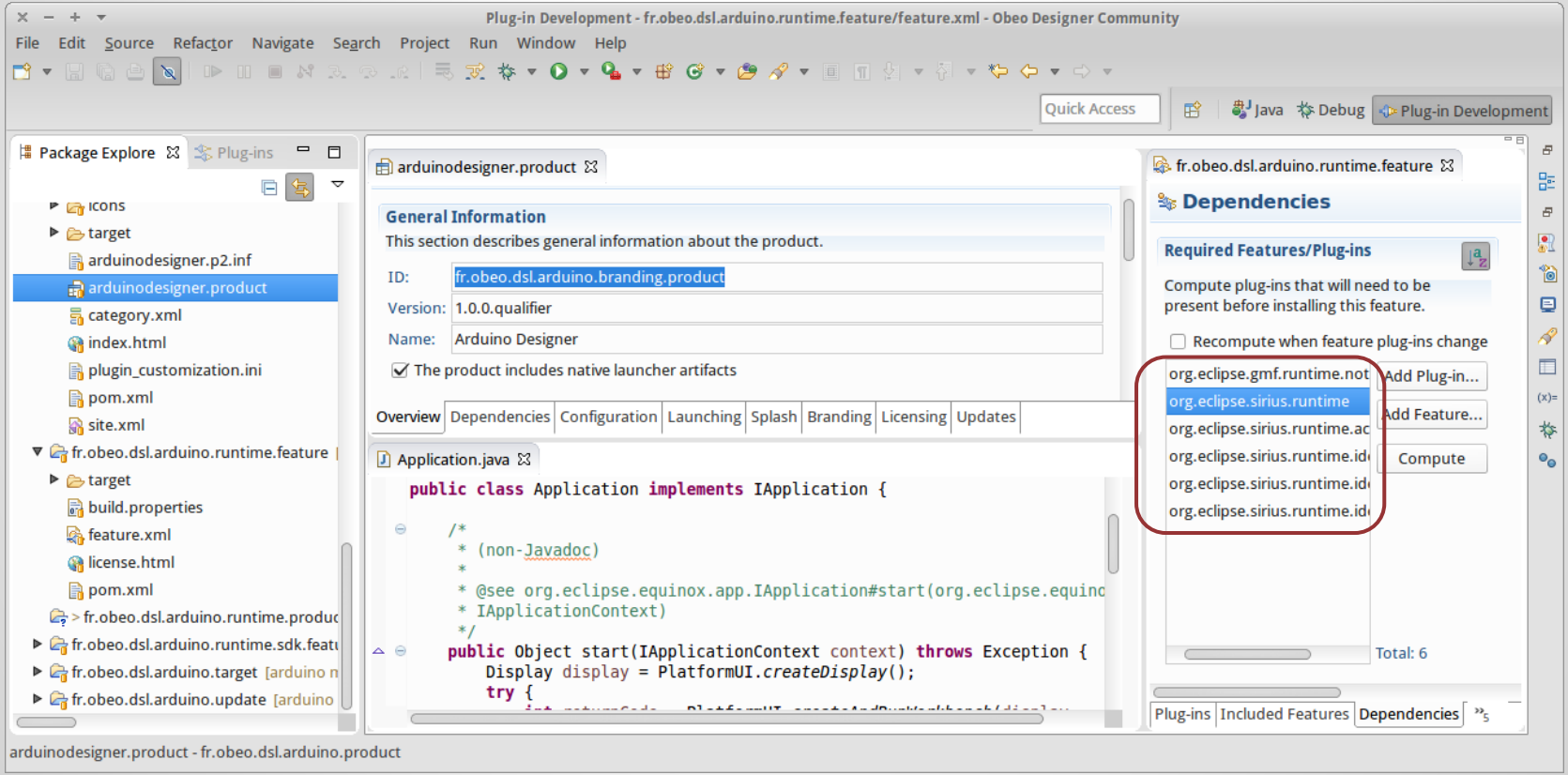
Build a product



Build a product



Build a product



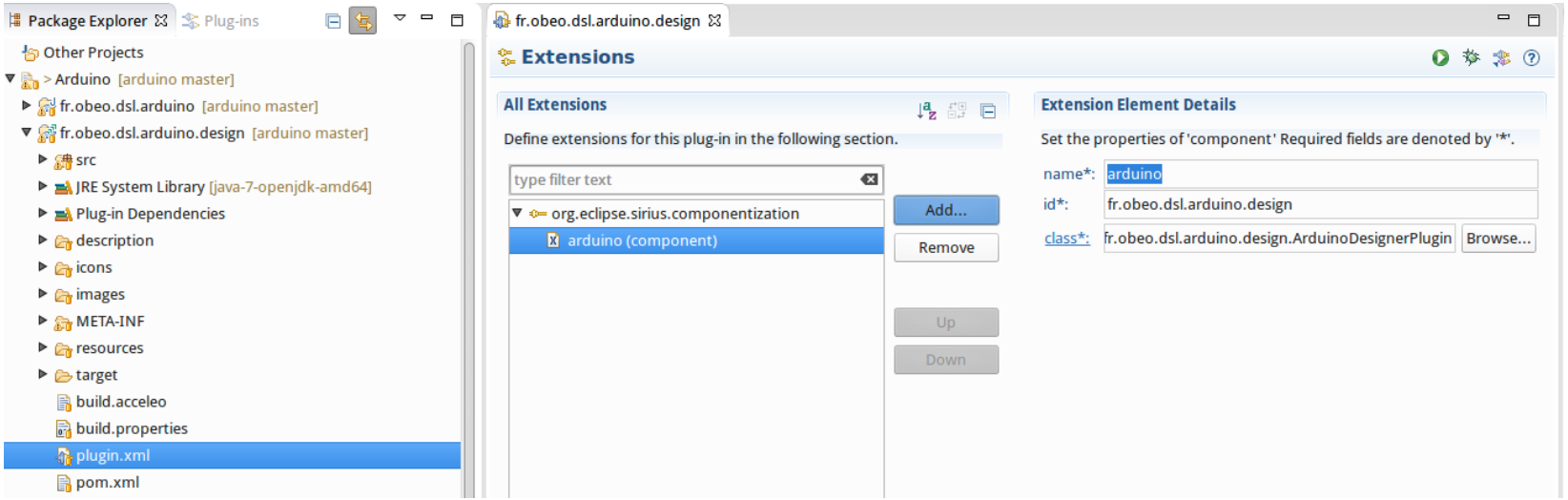
Build a product

Have a look to Sirius Developer Manual :

Deploy a Modeler description file

Build a product

Contribute to org.eclipse.sirius.componentization



Build a product

Delete your viewpoints in the stop of the plugin

```
public void start(BundleContext context) throws Exception {
  super.start(context);
  plugin = this;
  viewpoints = new HashSet<Viewpoint>();
  viewpoints.addAll(ViewpointRegistry.getInstance().registerFromPlugin(PLUGIN_ID + "/description/arduino.odesign"));
}
```

Register your viewpoints at the start of the plugin

```
public void stop(BundleContext context) throws Exception {
plugin = null;
if (viewpoints != null) {
    for (final Viewpoint viewpoint: viewpoints) {
    ViewpointRegistry.getInstance().disposeFromPlugin(viewpoint);
    }
    viewpoints.clear();
    viewpoints = null;
}
super.stop(context);
}
```

Target Platform

Use target platform definition DSL and generator : https://github.com/mbarbero/fr.obeo.releng.targetplatform

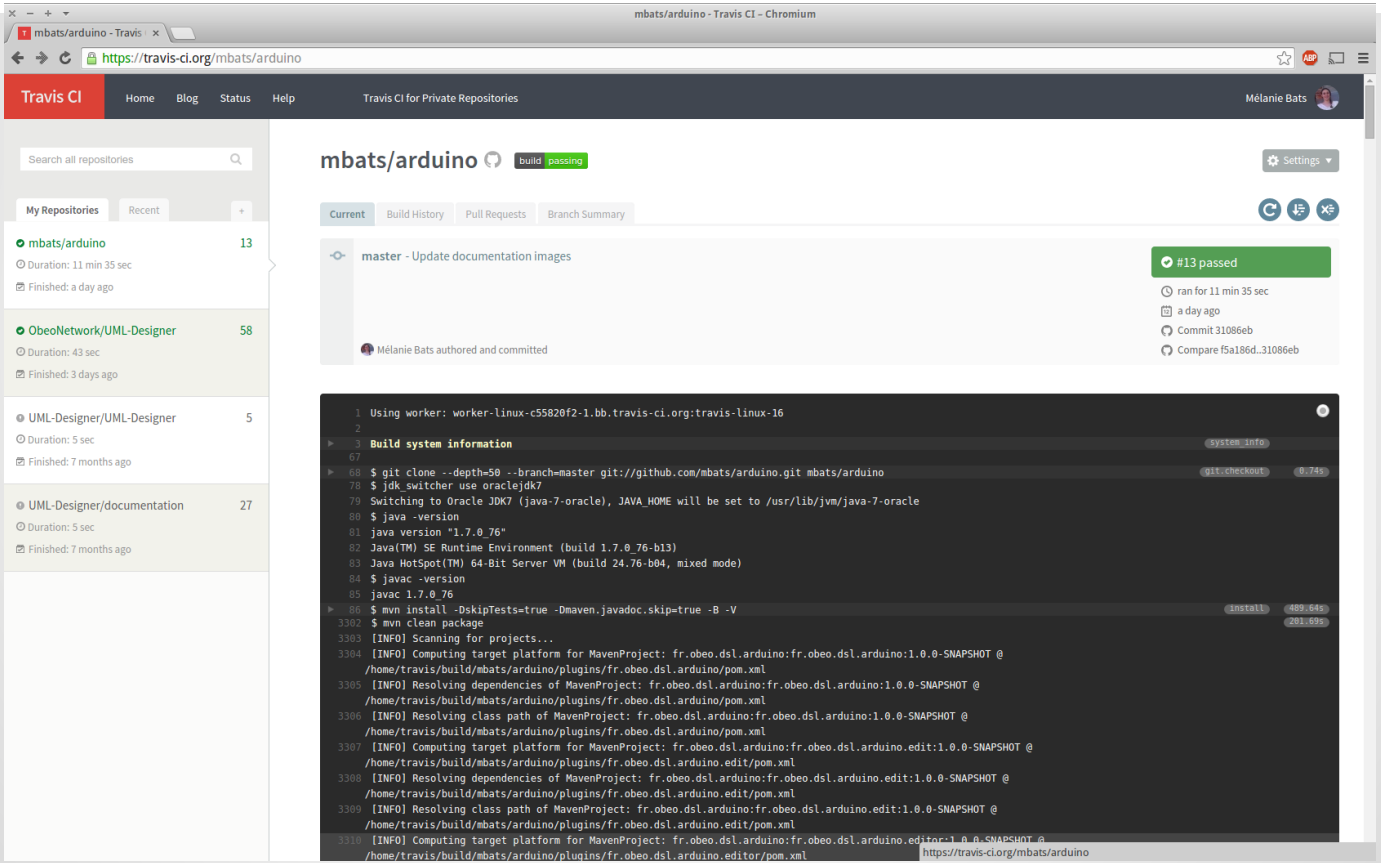
arduinodesigner-luna.tpd

```
target "ArduinoDesigner-luna"

with source, requirements

// Eclipse release
location "http://download.eclipse.org/releases/luna" {
    org.eclipse.platform.sdk lazy
    org.eclipse.equinox.executable.feature.group lazy
}

// Acceleo 3
location "http://download.eclipse.org/acceleo/updates/milestones/3.5/S201406101309/" {
    org.eclipse.acceleo.feature.group lazy
    org.eclipse.acceleo.runtime.feature.group lazy
}

// Sirius
location "http://download.eclipse.org/sirius/updates/releases/2.0.3/luna" {
    //runtime
    org.eclipse.sirius.runtime.feature.group
    org.eclipse.sirius.runtime.acceleo.feature.group
    //IDE
    org.eclipse.sirius.runtime.ide.ui.acceleo.feature.group
    org.eclipse.sirius.runtime.ide.ui.feature.group
}
```

Tycho

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Copyright (c) 2013 Obeo.
All rights reserved. This program and the accompanying materials
are made available under the terms of the Eclipse Public License v1.0
which accompanies this distribution, and is available at
http://www.eclipse.org/legal/epl-v10.html

Contributors:
    Obeo - initial API and implementation
-->
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <parent>
    <groupId>fr.obeo.dsl.arduino</groupId>
    <artifactId>common.parent</artifactId>
    <version>1.0.0-SNAPSHOT</version>
    <relativePath>../../releng/fr.obeo.dsl.arduino.common.parent</relativePath>
  </parent>
  <groupId>fr.obeo.dsl.arduino</groupId>
  <artifactId>parent</artifactId>
  <packaging>pom</packaging>

  <name>Arduino Designer Parent</name>

  <modules>
    <!-- Build plugins -->
    <module>${root-path}/plugins/fr.obeo.dsl.arduino</module>
    <module>${root-path}/plugins/fr.obeo.dsl.arduino.edit</module>
    <module>${root-path}/plugins/fr.obeo.dsl.arduino.editor</module>
    <module>${root-path}/plugins/fr.obeo.dsl.arduino.design</module>
    <module>${root-path}/plugins/fr.obeo.dsl.arduino.branding</module>
    <module>${root-path}/plugins/fr.obeo.dsl.arduino.gen</module>
    <module>${root-path}/plugins/fr.obeo.dsl.arduino.preferences</module>
    <module>${root-path}/plugins/fr.obeo.dsl.arduino.ui</module>
    <module>${root-path}/plugins/fr.obeo.dsl.arduino.build</module>

    <!-- Build test plugins -->

    <!-- Build features -->
    <module>${root-path}/packaging/fr.obeo.dsl.arduino.runtime.feature</module>

    <!-- Build update-sites -->
    <module>${root-path}/packaging/fr.obeo.dsl.arduino.update</module>

    <!-- Build products -->
    <module>${root-path}/packaging/fr.obeo.dsl.arduino.product</module>
  </modules>
</project>
```

Travis


mbats/arduino - Travis CI – Chromium
mbats/arduino - Travis
https://travis-ci.org/mbats/arduino
Travis CI
Home
Blog
Status
Help
Travis CI for Private Repositories
Mélanie Bats
Search all repositories
My Repositories
Recent
mbats/arduino 13
Duration: 11 min 35 sec
Finished: a day ago
ObeoNetwork/UML-Designer 58
Duration: 43 sec
Finished: 3 days ago
UML-Designer/UML-Designer 5
Duration: 5 sec
Finished: 7 months ago
UML-Designer/documentation 27
Duration: 5 sec
Finished: 7 months ago
mbats/arduino
build passing
Settings
Current
Build History
Pull Requests
Branch Summary
master - Update documentation images
Mélanie Bats authored and committed
#13 passed
ran for 11 min 35 sec
a day ago
Commit 31086eb
Compare f5a186d..31086eb
Using worker: worker-linux-c55820f2-1.bb.travis-ci.org:travis-linux-16
Build system information
system_info
$ git clone --depth=50 --branch=master git://github.com/mbats/arduino.git mbats/arduino
git.checkout
0.74s
$ jdk_switcher use oraclejdk7
Switching to Oracle JDK7 (java-7-oracle), JAVA_HOME will be set to /usr/lib/jvm/java-7-oracle
$ java -version
java version "1.7.0_76"
Java(TM) SE Runtime Environment (build 1.7.0_76-b13)
Java HotSpot(TM) 64-Bit Server VM (build 24.76-b04, mixed mode)
$ javac -version
javac 1.7.0_76
$ mvn install -DskipTests=true -Dmaven.javadoc.skip=true -B -V
install
489.64s
$ mvn clean package
201.69s
[INFO] Scanning for projects...
[INFO] Computing target platform for MavenProject: fr.obeo.dsl.arduino:fr.obeo.dsl.arduino:1.0.0-SNAPSHOT @ /home/travis/build/mbats/arduino/plugins/fr.obeo.dsl.arduino/pom.xml
[INFO] Resolving dependencies of MavenProject: fr.obeo.dsl.arduino:fr.obeo.dsl.arduino:1.0.0-SNAPSHOT @ /home/travis/build/mbats/arduino/plugins/fr.obeo.dsl.arduino/pom.xml
[INFO] Resolving class path of MavenProject: fr.obeo.dsl.arduino:fr.obeo.dsl.arduino:1.0.0-SNAPSHOT @ /home/travis/build/mbats/arduino/plugins/fr.obeo.dsl.arduino/pom.xml
[INFO] Computing target platform for MavenProject: fr.obeo.dsl.arduino:fr.obeo.dsl.arduino.edit:1.0.0-SNAPSHOT @ /home/travis/build/mbats/arduino/plugins/fr.obeo.dsl.arduino.edit/pom.xml
[INFO] Resolving dependencies of MavenProject: fr.obeo.dsl.arduino:fr.obeo.dsl.arduino.edit:1.0.0-SNAPSHOT @ /home/travis/build/mbats/arduino/plugins/fr.obeo.dsl.arduino.edit/pom.xml
[INFO] Resolving class path of MavenProject: fr.obeo.dsl.arduino:fr.obeo.dsl.arduino.edit:1.0.0-SNAPSHOT @ /home/travis/build/mbats/arduino/plugins/fr.obeo.dsl.arduino.edit/pom.xml
[INFO] Computing target platform for MavenProject: fr.obeo.dsl.arduino:fr.obeo.dsl.arduino.editor:1.0.0-SNAPSHOT @ /home/travis/build/mbats/arduino/plugins/fr.obeo.dsl.arduino.editor/pom.xml
https://travis-ci.org/mbats/arduino

.travis.yml

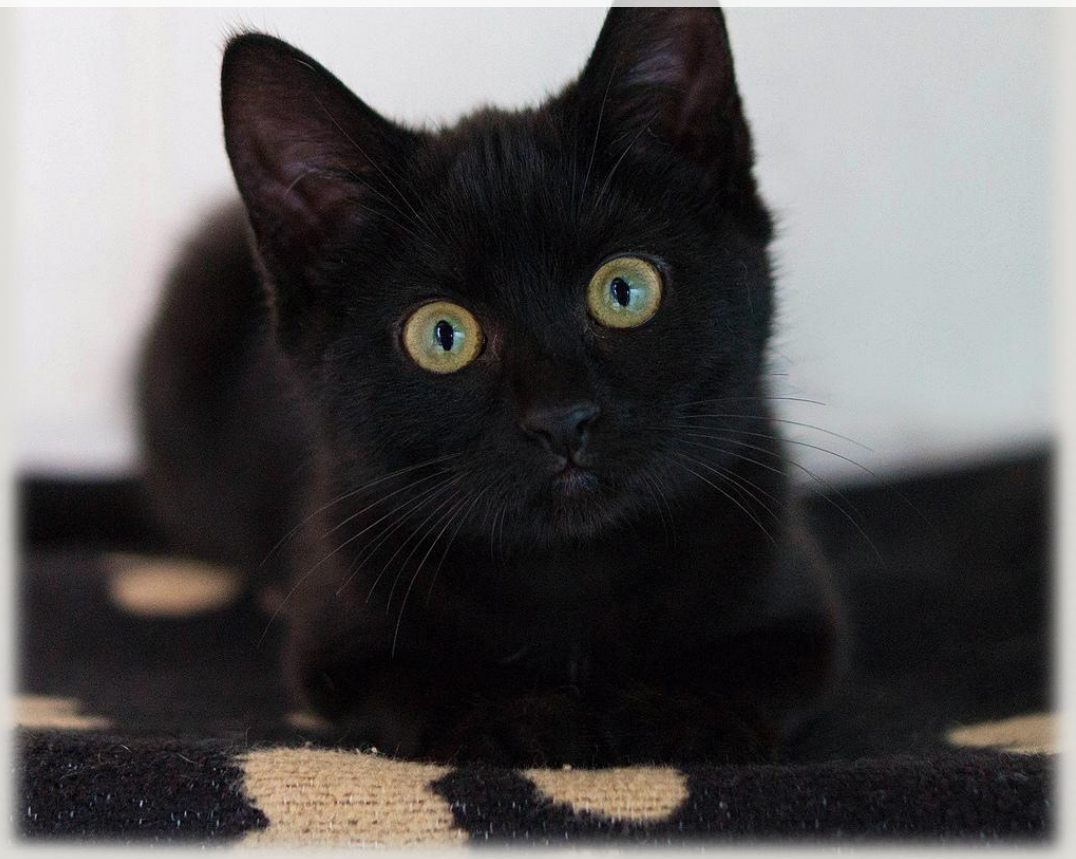
```
language: java
script: mvn clean package
jdk: oraclejdk7
deploy:
  provider: releases
  api_key:
    secure: IQP5qqJ6lZtZYU6814OVyVcVLsTXlNzd4yRBF/KSIzABgJ6jtbgFthAub3nMKWGPXLZZYzGJzChU+mcvXCaOAh3DCcTSPun404oP2ORdYpSoexios+wIqLRz6XordpeMNCPURM2XVcRAR8w+HSGlFtW/rA5GkbnK+L155/K5bRg=
  file:
  - $TRAVIS_BUILD_DIR/packaging/fr.obeo.dsl.arduino.product/target/products/ArduinoDesigner-linux.gtk.x86.zip
  - $TRAVIS_BUILD_DIR/packaging/fr.obeo.dsl.arduino.product/target/products/ArduinoDesigner-linux.gtk.x86_64.zip
  - $TRAVIS_BUILD_DIR/packaging/fr.obeo.dsl.arduino.product/target/products/ArduinoDesigner-macosx.cocoa.x86_64.zip
  - $TRAVIS_BUILD_DIR/packaging/fr.obeo.dsl.arduino.product/target/products/ArduinoDesigner-win32.win32.x86.zip
  - $TRAVIS_BUILD_DIR/packaging/fr.obeo.dsl.arduino.product/target/products/ArduinoDesigner-win32.win32.x86_64.zip
  on:
    repo: mbats/arduino
    tags: true
    all_branches: true
```

Products available

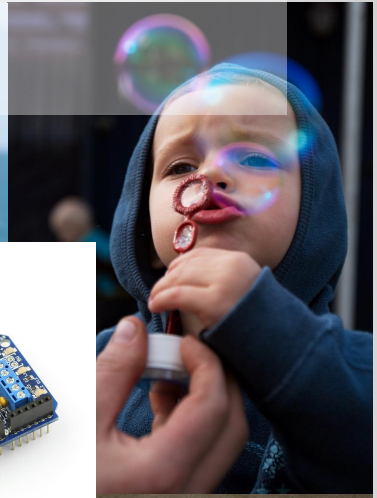
On github : https://github.com/mbats/arduino/releases


Model + Microcontroller = ♥

And where is the cat in this story?



The cat making-of ?



Inspire and create yours!

Code available on :

https://github.com/mbats/arduino

Ask questions about Sirius on the forum

Do not miss!

Wednesday

13:30 - **Sirius + Xtext = ♥**

18:00 - **Sirius Poster**

Thursday

14:30 - **Time-lapse**

15:15 - **Viewpoint: the making of**

Give your feedback !

Mélanie Bats
melanie.bats@obeo.fr
@melaniebats

Obeo
Model Driven Company