

Programming using Wylidrin

SLIDES TAKEN FROM IOT SUMMER CAMP 2015, INNOVATION LABS

What Language Should I use?



Visual Programming

```
Print on screen "Led on pin 0 should blink"  
Print on screen "Press the Stop button to stop"  
repeat while true  
do  
  Set On LED on pin 0  
  delay 500 milliseconds  
  Set Off LED on pin 0  
  delay 500 milliseconds
```

The image shows a sequence of Scratch code blocks. It starts with two 'Print on screen' blocks. The first block contains the text "Led on pin 0 should blink" and the second block contains "Press the Stop button to stop". Below these is a 'repeat while' loop with the condition 'true'. Inside the loop, there is a 'do' block containing four sub-blocks: 'Set On LED on pin 0', 'delay 500 milliseconds', 'Set Off LED on pin 0', and another 'delay 500 milliseconds' block.

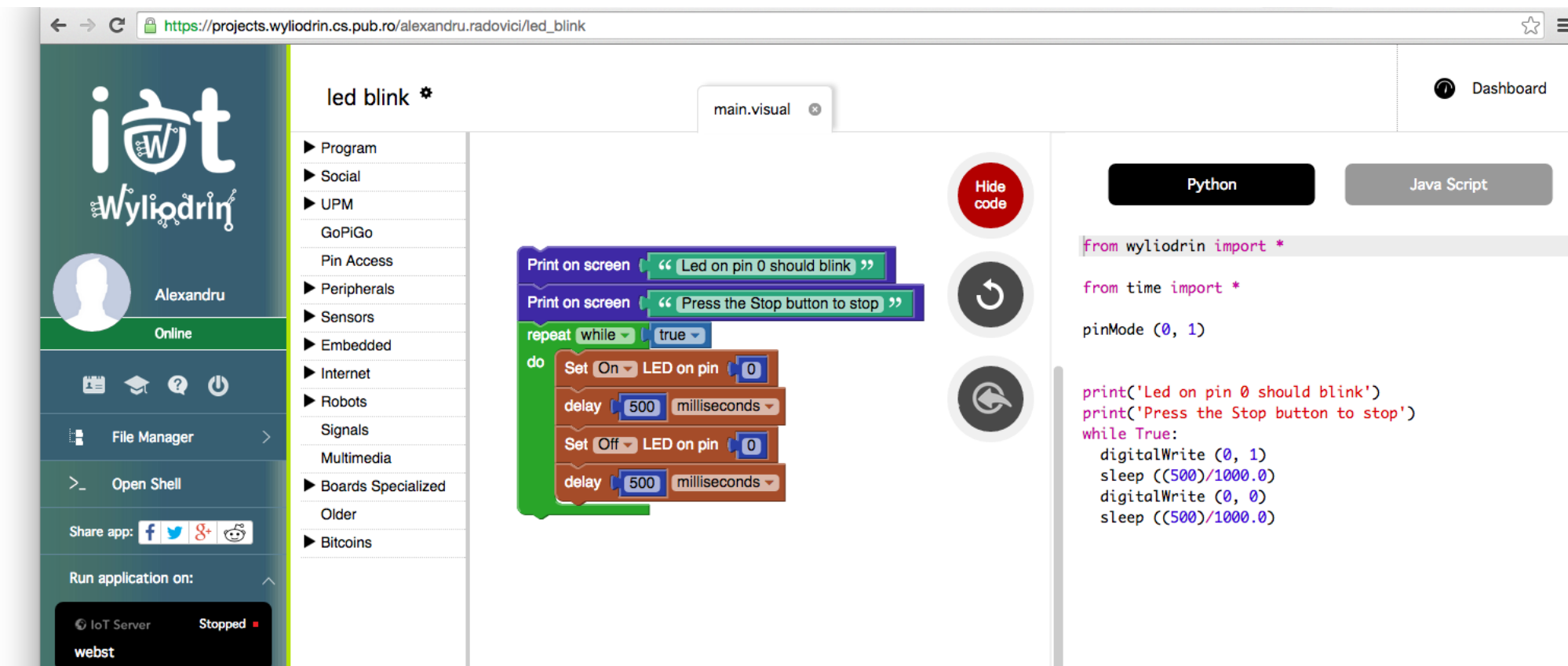
Prototyping

Drag and drop blocks

Writes code in Python or Javascript

Good for documentation

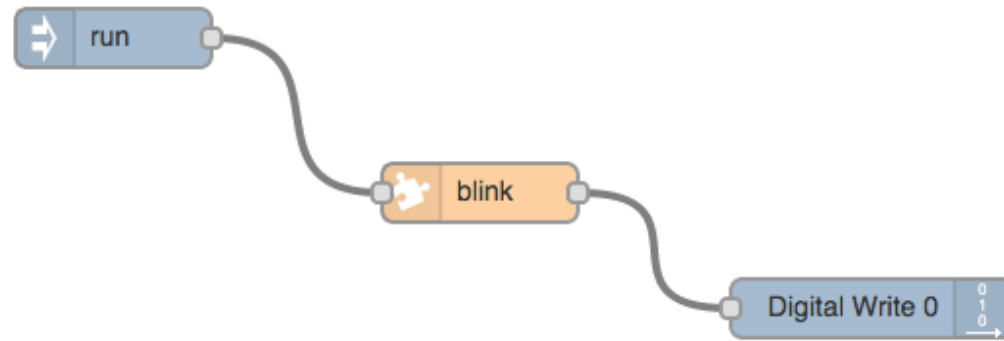
Python



The screenshot shows the Wylodrin IoT development environment. The browser address bar displays `https://projects.wylodrin.cs.pub.ro/alexandru.radovici/led_blink`. The interface includes a sidebar with the user profile 'Alexandru' and a navigation menu with categories like Program, Social, UPM, GoPiGo, Pin Access, Peripherals, Sensors, Embedded, Internet, Robots, Signals, Multimedia, Boards Specialized, Older, and Bitcoins. The main workspace is titled 'led blink' and contains a 'main.visual' tab. On the left, there are two 'Print on screen' blocks with messages: 'Led on pin 0 should blink' and 'Press the Stop button to stop'. Below these is a 'repeat while' loop with 'true' selected, containing a 'do' block with four steps: 'Set On LED on pin 0', 'delay 500 milliseconds', 'Set Off LED on pin 0', and 'delay 500 milliseconds'. On the right, there are three control buttons: 'Hide code', a refresh button, and a back button. Below these are two tabs: 'Python' (selected) and 'Java Script'. The Python code editor shows the following code:

```
from wylodrin import *  
  
from time import *  
  
pinMode (0, 1)  
  
print('Led on pin 0 should blink')  
print('Press the Stop button to stop')  
while True:  
    digitalWrite (0, 1)  
    sleep ((500)/1000.0)  
    digitalWrite (0, 0)  
    sleep ((500)/1000.0)
```

Streams Programming



Data Driven Programming

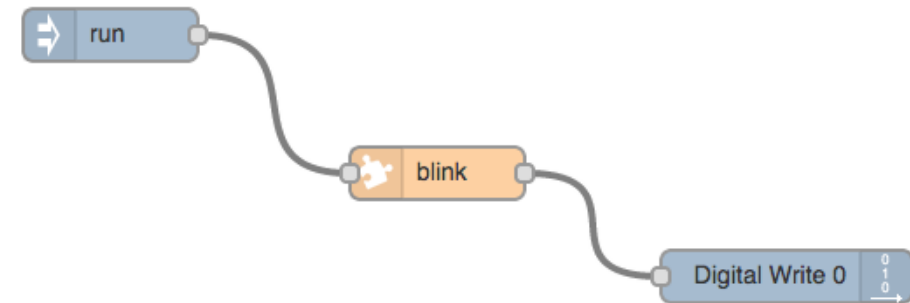
Event based

Data determines actions

The program is a graph

Elements

- Nodes
- Data routes



Messages

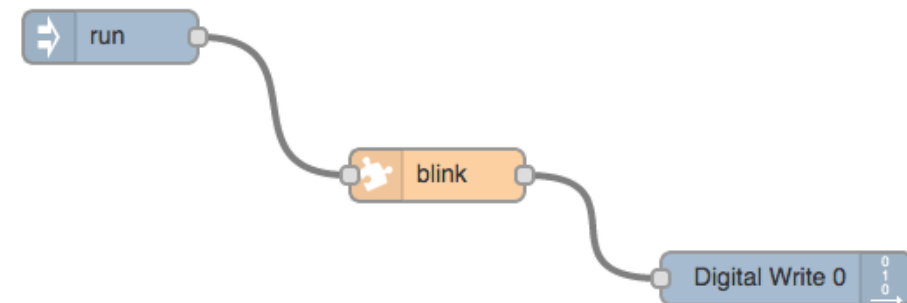
Nodes send messages to each other

- An input
- Multiple outputs

When a message arrives in the input

- Node activates
- Might send out some messages

Example



Messages

Javascript objects

{

Nodes expect to receive payload

topic:

payload:

...

}

Nodes

Run

Digital and analog pins access

Trigger

Delay

Value

Buffer

Switch

Change

Range

Function

Visual

Subflows

Signals

Messages

Print

Web request

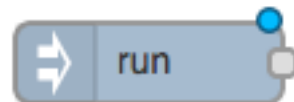
Web server

Run

Sends a message at a certain interval

Payload:

- Timestamp
- Blank
- String
- # number of message



Edit run node

✉ Payload

☰ Topic

🔄 Repeat

every

Fire once at start ?

👉 Name

Note: "interval between times" and "at a specific time" will use cron. See info box for details.

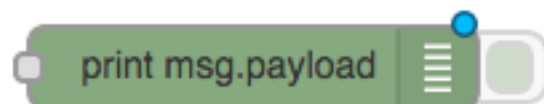
Ok Cancel

Print

Prints to the screen

- Only payload
- Whole messages
- Some field

May be stopped with the button



Edit print node

☰ Output

👉 Field

👉 Name

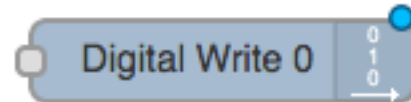
Ok

Cancel

Digital and analog pins access

Digital write

- Writes HIGH(1) or LOW (0) when it receives a message
- The value written is the payload



Edit digitalwrite node

✎ Name

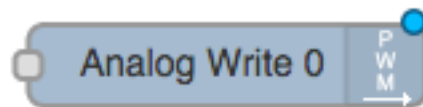
● Pin Number

Ok Cancel

Digital and analog pins access

Analog write

- Writes PWM when it receives a message
- The value written is the payload



Edit analogwrite node

Name

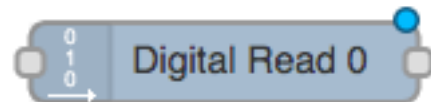
● Pin Number

Ok Cancel

Digital and analog pins access

Digital read

- Reads HIGH(1) or LOW (0) when it receives a message
- The value read is sent in the payload



Edit digitalread node

Name

● Pin Number

Ok Cancel

Digital and analog pins access

Analog read

- Reads an analog value when it receives a message
- The value read is sent in the payload



Edit analogread node

Name

● Pin Number

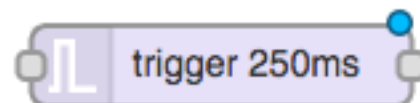
Ok

Cancel

Trigger

When it receives a message

- Sends a message with Output as payload
- Waits for some time
- Sends the second Output as a payload



Edit trigger node

↑ Output

⌚ then wait

Milliseconds

↓ output

⌚ and

don't extend the timer if retriggered

📁 Name

Name

Setting the timeout to 0 sets an infinite timeout = single shot.

Ok

Cancel

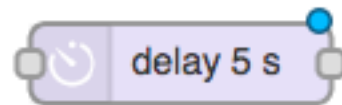
Delay

Delay

- Sends the received message with a delay

Limit

- Limits the number of messages that it sends
 - Drops or stores messages



Edit delay node

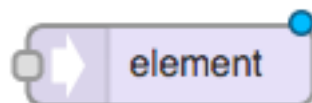
Action	Delay message
For	5 Seconds
Name	Name

Ok Cancel

Value

Value

- Stores the payload
- context.global._value_name_



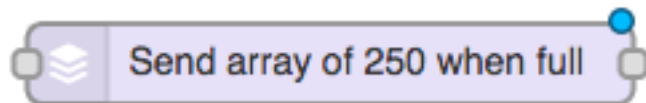
Edit value node

Name	<input type="text" value="Name"/>
Value	<input type="text" value="element"/>

Buffer

When it receives a messages

- Adds the payload and other properties to an array
- It sends the array when
 - The buffer is full
 - Receives an event



Edit buffer node

Send when buffer is full

Store array

Size 250

If buffer is full shift messages

Use a sliding window

Name Name

Ok

Cancel

Switch

When it receives a messages

- Checks some rules
- Send the message using the rule's output
- Values may be used
 - {{value}}



Edit switch node

Name

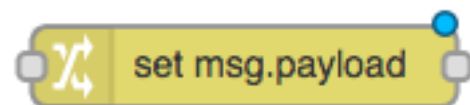
If msg.

== ▾	<input type="text"/>	send to 1 <input type="button" value="x"/>
== ▾	<input type="text"/>	send to 2 <input type="button" value="x"/>
== ▾	<input type="text"/>	send to 3 <input type="button" value="x"/>
== ▾	<input type="text"/>	send to 4 <input type="button" value="x"/>
== ▾	<input type="text"/>	send to 5 <input type="button" value="x"/>

checking all rules ▾

Change

Modifies the properties of a message



Edit change node

Set the value of the message property ▾

called

to

Tip: expects a new property name and either a fixed value OR the full name of another message property eg: msg.sentiment.score

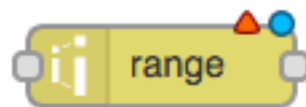
Name

Ok

Cancel

Range

Scales the payload



Edit range node

⊙ Action

➔ Map the input range:

from: to:

➔ to the result range:

from: to:

Round result to the nearest integer?

👉 Name

Tip: This node ONLY works with numbers.

Ok

Cancel

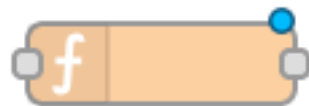
Function

Javascript function

- msg the message received
- context.global values stored with the value node
- return sends the message

May have multiple outputs

- return [o1, o2, o3]



Edit function node

Name

Function

```
1  
2 return msg;
```

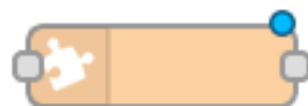
Outputs

See the Info tab for help writing functions.

Ok Cancel

Visual

Visual Programming Function



Edit visual node

Name

Function

- Function
- ▶ Program
- ▶ UPM
- GoPiGo
- Pin Access
- ▶ Peripherals
- ▶ Sensors
- ▶ Robots
- Signals
- ▶ Boards Specialized
- Older

```
set new message to new message
set message. " payload " value " value "
return new message
```

Outputs 1

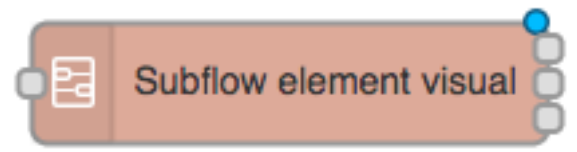
See the Info tab for help writing functions.

OK Cancel

Subflows

Subflows

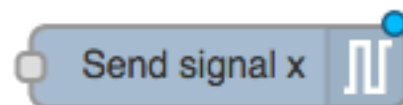
edit name + input + output delete subflow



Send signal

Sends a signal

- To the dashboard
- To another board



Edit sendsignal node

Name

To

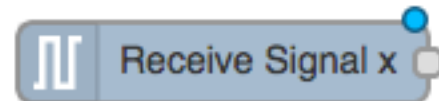
Signal

Ok Cancel

Receive signal

Receives a signal

- Sends a message with the signal received
- Payload is the value
- Topic is the name
- Sender
 - User id if it comes from the dashboard
 - Board id if it comes from a board



Edit receivesignal node

Name

Signal

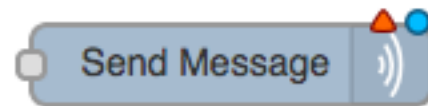
Ok

Cancel

Send message

Sends a message to another board

- The message is the payload
- Uses label instead of port



Edit send node

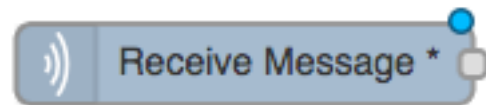
Name	<input type="text" value="Name"/>
Label	<input type="text" value="label"/>
Board ID	<input type="text" value="boardid@wylidrin.com"/>

Ok Cancel

Receive message

Receives a message from another board

- The message is the payload
- Sender is the board's id
- Uses label instead of port



Edit receive node

Name

Label

Ok

Cancel