Tag: Job Control in urbiscript

Jean-Christophe Baillie Akim Demaille Quentin Hocquet
Matthieu Nottale

Gostai S.A.S.

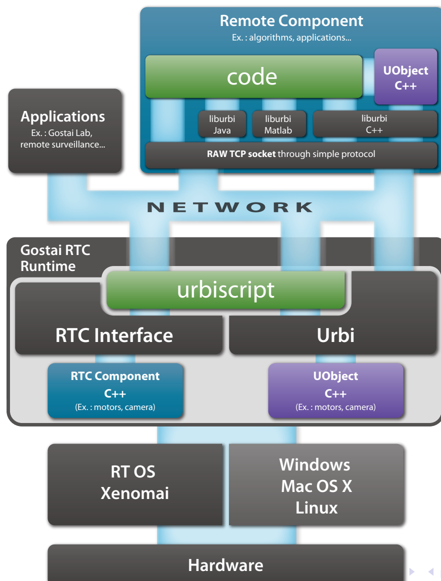
Control Architectures of Robots 2010
May 18th, 2010

1. Concurrency
   - Flow Control
   - Events
2. Tags
   - Jobs
   - Events
   - Misc

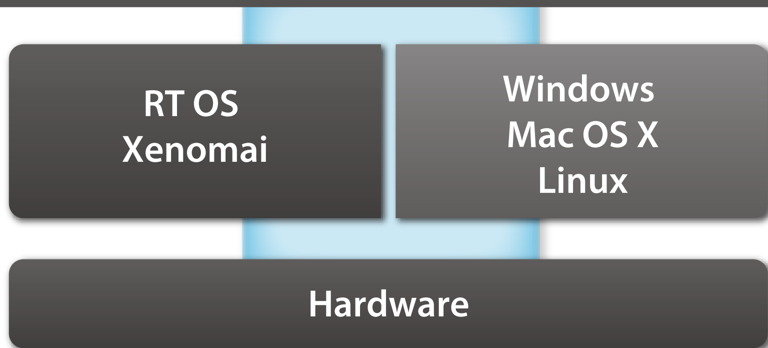
URBI
2.0

Open Source
AGPL v3

Architecture

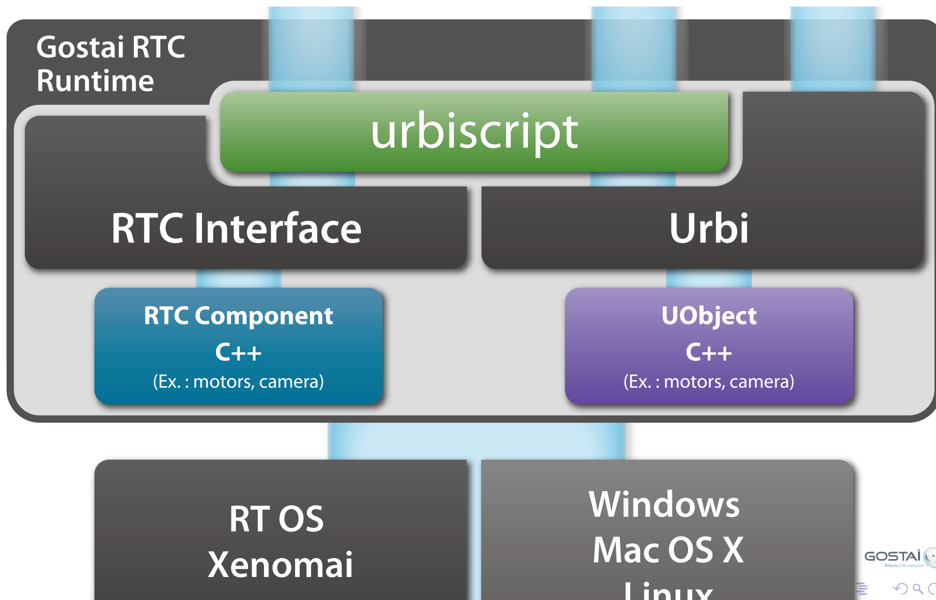


Architecture



GOSTAI
Robotics for everyone

Architecture



Architecture

Remote Component

Ex. : algorithms, applications...

code

UObject
C++

liburbi
Java

liburbi
Matlab

liburbi
C++

RAW TCP socket through simple protocol

NETWORK

GOSTAI
Robotics for everyone

Architecture

Applications

Ex. : Gostai Lab, remote surveillance...

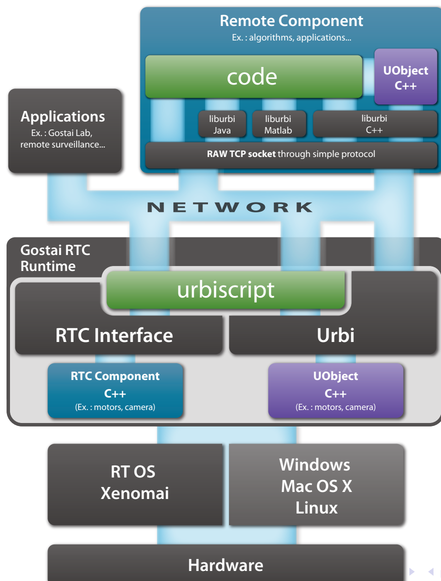
CO

liburbi Java

RAW TCP

NETW

Architecture



Concurrency

1. Concurrency
   - Flow Control
   - Events
2. Tags

Flow Control

1 Concurrency
- Flow Control
- Events

2 Tags

GOSTAI Robotics for everyone

From Sequential to Concurrent

sequentiality

$a ; b$

background concurrency

a , b

tight sequentiality

$a | b$

fair-start concurrency

$a \& b$

From Sequential to Concurrent

sequentiality

$a ; b$

tight sequentiality

$a | b$

background concurrency

a , b

fair-start concurrency

$a \& b$

From Sequential to Concurrent

sequentiality

$a;b$

tight sequentiality

$a|b$

background concurrency

a,b

fair-start concurrency

$a\&b$

GOSTAI Robotics for everyone

From Sequential to Concurrent

sequentiality

$a;b$

background concurrency

a,b

tight sequentiality

$a|b$

fair-start concurrency

$a\&b$

Backgrounding Commands

```
for (3)
{
  sleep(1s);
  echo("ping");
},
sleep(0.5s);
for (3)
{
  sleep(1s);
  echo("pong");
},
sleep(4s);
```

```
[00000316] *** ping
[00000316] *** pong
[00000316] *** ping
[00000316] *** pong
[00000316] *** ping
[00000316] *** pong
```

Backgrounding Commands

```
for (3)
{
  sleep(1s);
  echo("ping");
},
sleep(0.5s);
for (3)
{
  sleep(1s);
  echo("pong");
},
sleep(4s);
```

```
[00000316] *** ping
[00000316] *** pong
[00000316] *** ping
[00000316] *** pong
[00000316] *** ping
[00000316] *** pong
```

GOSTAI Robotics for everyone

Concurrent Flavors of Flow Control Constructs

```
for  (var i : 3.seq.reverse)
{
  echo("%s: start" % i);
  sleep(i);
  echo("%s: done" % i)
};
echo("done");
[00125189] *** 2: start
[00127190] *** 2: done
[00127190] *** 1: start
[00128192] *** 1: done
[00128192] *** 0: start
[00128193] *** 0: done
[00128194] *** done
```

```
for& (var i : 3.seq.reverse)
{
  echo("%s: start" % i);
  sleep(i);
  echo("%s: done" % i)
};
echo("done");
[00105789] *** 2: start
[00105789] *** 1: start
[00105789] *** 0: start
[00105793] *** 0: done
[00106793] *** 1: done
[00107793] *** 2: done
[00107795] *** done
```

GOSTAI Robotics for everyone

Concurrent Flavors of Flow Control Constructs

```
for  (var i : 3.seq.reverse)
{
  echo("%s: start" % i);
  sleep(i);
  echo("%s: done" % i)
};
echo("done");
[00125189] *** 2: start
[00127190] *** 2: done
[00127190] *** 1: start
[00128192] *** 1: done
[00128192] *** 0: start
[00128193] *** 0: done
[00128194] *** done
```

```
for& (var i : 3.seq.reverse)
{
  echo("%s: start" % i);
  sleep(i);
  echo("%s: done" % i)
};
echo("done");
[00105789] *** 2: start
[00105789] *** 1: start
[00105789] *** 0: start
[00105793] *** 0: done
[00106793] *** 1: done
[00107793] *** 2: done
[00107795] *** done
```

GOSTAI Robotics for everyone

Factorial

```
function echoFact (var n)
{
  var res = 1;
  for (var i = 2; i <= n; i++)
    res *= i;
  echo("%2d! = %7d" % [n, res]);
}|;

echoFact(9);
[00003434] ***  9! =  362880

echoFact(10);
[00003434] *** 10! = 3628800
```

```
for& (var i : 10.seq.reverse)
  echoFact(i);
[00018878] ***  1! =       1
[00018878] ***  0! =       1
[00018879] ***  2! =       2
[00018879] ***  3! =       6
[00018879] ***  4! =      24
[00018880] ***  5! =     120
[00018880] ***  6! =     720
[00018880] ***  7! =    5040
[00018881] ***  8! =   40320
[00018881] ***  9! =  362880
```

Factorial

```
function echoFact (var n)
{
  var res = 1;
  for (var i = 2; i <= n; i++)
    res *= i;
  echo("%2d! = %7d" % [n, res]);
}|;

echoFact(9);
[00003434] ***  9! =  362880

echoFact(10);
[00003434] *** 10! = 3628800
```

```
for& (var i : 10.seq.reverse)
  echoFact(i);
[00018878] ***  1! =       1
[00018878] ***  0! =       1
[00018879] ***  2! =       2
[00018879] ***  3! =       6
[00018879] ***  4! =      24
[00018880] ***  5! =     120
[00018880] ***  6! =     720
[00018880] ***  7! =    5040
[00018881] ***  8! =   40320
[00018881] ***  9! =  362880
```

Factorial

```
echoFact = function (var n)
{

  var res = 1;
  for (var i = 2; i <= n; i++)
    res *= i;
  echo("%2d! = %7d" % [n, res])
}|;

for& (var i : 4.seq.reverse)
  echoFact(i);
[00018878] ***   1! =        1
[00018878] ***   0! =        1
[00018879] ***   2! =        2
[00018879] ***   3! =        6
```

```
echoFact = function (var n)
{
  nonInterruptible;
  var res = 1;
  for (var i = 2; i <= n; i++)
    res *= i;
  echo("%2d! = %7d" % [n, res])
}|;

for& (var i : 4.seq.reverse)
  echoFact(i);
[00000051] ***   3! =        6
[00000052] ***   2! =        2
[00000053] ***   1! =        1
[00000054] ***   0! =        1
```

GOSTAI Robotics for everyone

Factorial

```
echoFact = function (var n)
{

  var res = 1;
  for (var i = 2; i <= n; i++)
    res *= i;
  echo("%2d! = %7d" % [n, res])
}|;

for& (var i : 4.seq.reverse)
  echoFact(i);
[00018878] ***   1! =         1
[00018878] ***   0! =         1
[00018879] ***   2! =         2
[00018879] ***   3! =         6
```

```
echoFact = function (var n)
{
  nonInterruptible;
  var res = 1;
  for (var i = 2; i <= n; i++)
    res *= i;
  echo("%2d! = %7d" % [n, res])
}|;

for& (var i : 4.seq.reverse)
  echoFact(i);
[00000051] ***   3! =         6
[00000052] ***   2! =         2
[00000053] ***   1! =         1
[00000054] ***   0! =         1
```

Events

1 Concurrency
- Flow Control
- Events

2 Tags

Events are Messages

```
var e = Event.new|;
at (e?)
  echo("ping");

e!;
[00000000] *** ping

e!;
[00000000] *** ping
```

Events carry Messages

```
var e = Event.new|;

at (e?)
  echo("e");

at (e?(var x))
  echo("e(x)");

at (e?(1))
  echo("e(1)");

at (e?(var x) if x % 2)
  echo("e(odd)");
```

```
e!;
[00000845] *** e

e!(0);
[00011902] *** e
[00011902] *** e(x)

e!(1);
[00023327] *** e
[00023327] *** e(x)
[00023327] *** e(1)
[00023327] *** e(odd)
```

GOSTAI Robotics for everyone

Events carry Messages

```
var e = Event.new|;

at (e?)
  echo("e");

at (e?(var x))
  echo("e(x)");

at (e?(1))
  echo("e(1)");

at (e?(var x) if x % 2)
  echo("e(odd)");
```

```
e!;
[00000845] *** e

e!(0);
[00011902] *** e
[00011902] *** e(x)

e!(1);
[00023327] *** e
[00023327] *** e(x)
[00023327] *** e(1)
[00023327] *** e(odd)
```

Assignments are Events

```
var e = 0|;

at (e)
  echo("e is true!")
onleave
  echo("e is false!");

at (e == 2)
  echo("e is two!");

at (e % 2 == 0)
  echo("e is even!");
[00028013] *** e is even!
```

```
e = 0|;
// A big fat nothing.

e = 1|;
[00023327] *** e is true!

e = 2|;   // Still true.
[00028013] *** e is two!
[00028013] *** e is even!

e = 0|;   // Still even.
[00126958] *** e is false!

e = 2|;   // Still even.
[00137252] *** e is true!
[00137252] *** e is two!
```

Assignments are Events

```
var e = 0|;

at (e)
  echo("e is true!")
onleave
  echo("e is false!");

at (e == 2)
  echo("e is two!");

at (e % 2 == 0)
  echo("e is even!");
[00028013] *** e is even!
```

```
e = 0|;
// A big fat nothing.

e = 1|;
[00023327] *** e is true!

e = 2|;    // Still true.
[00028013] *** e is two!
[00028013] *** e is even!

e = 0|;    // Still even.
[00126958] *** e is false!

e = 2|;    // Still even.
[00137252] *** e is true!
[00137252] *** e is two!
```

Tags

1 Concurrency

2 Tags
- Jobs
- Events
- Misc

GOSTAI Robotics for everyone

Managing Concurrent Executions

Backgrounded code might need to be

- killed
- paused
- resumed
- more generally, managed

Tags

Managing Concurrent Executions

Backgrounded code might need to be

- killed
- paused
- resumed
- more generally, managed

Tags

Managing Concurrent Executions

Backgrounded code might need to be

- killed
- paused
- resumed
- more generally, managed

Tags

Managing Concurrent Executions

Backgrounded code might need to be

- killed
- paused
- resumed
- more generally, managed

Tags

GOSTAI Robotics for everyone

Managing Concurrent Executions

Backgrounded code might need to be

- killed
- paused
- resumed
- more generally, managed

Tags

Jobs

1 Concurrency

2 Tags
- Jobs
- Events
- Misc

GOSTAI Robotics for everyone

Stop

```
var t = Tag.new|;
var t0 = time|;
t: every(1s) echo("foo"),
sleep(2.2s);
[00000158] *** foo
[00001159] *** foo
[00002159] *** foo

t.stop;
// Nothing runs.
sleep(2.2s);
```

```
t: every(1s) echo("bar"),
sleep(2.2s);
[00000158] *** bar
[00001159] *** bar
[00002159] *** bar

t.stop;
```

Stop

```
var t = Tag.new|;
var t0 = time|;
t: every(1s) echo("foo"),
sleep(2.2s);
[00000158] *** foo
[00001159] *** foo
[00002159] *** foo

t.stop;
// Nothing runs.
sleep(2.2s);
```

```
t: every(1s) echo("bar"),
sleep(2.2s);
[00000158] *** bar
[00001159] *** bar
[00002159] *** bar

t.stop;
```

GOSTAI Robotics for everyone

Stop with a Value

```
var t = Tag.new|;
var res;
detach(res = { t: every(1s) echo("computing") })|;
sleep(2.2s);
[00000001] *** computing
[00000002] *** computing
[00000003] *** computing

t.stop("result");
assert(res == "result");
```

Block

```
var ping = Tag.new("ping")|;
ping:
  every (1s)
    echo("ping"),
assert(!ping.blocked);
sleep(2.1s);
[00000000] *** ping
[00002000] *** ping
[00002000] *** ping

ping.block;
assert(ping.blocked);

ping:
  every (1s)
    echo("pong"),
// Neither new nor old code runs.
```

```
ping.unblock;
assert(!ping.blocked);
sleep(2.1s);

// But we can use the tag again.
ping:
  every (1s)
    echo("ping again"),
sleep(2.1s);
[00004000] *** ping again
[00005000] *** ping again
[00006000] *** ping again
```

Block

```
var ping = Tag.new("ping")|;
ping:
  every (1s)
    echo("ping"),
assert(!ping.blocked);
sleep(2.1s);
[00000000] *** ping
[00002000] *** ping
[00002000] *** ping

ping.block;
assert(ping.blocked);

ping:
  every (1s)
    echo("pong"),
// Neither new nor old code runs.
```

```
ping.unblock;
assert(!ping.blocked);
sleep(2.1s);

// But we can use the tag again.
ping:
  every (1s)
    echo("ping again"),
sleep(2.1s);
[00004000] *** ping again
[00005000] *** ping again
[00006000] *** ping again
```

Events

1 Concurrency

2 Tags
- Jobs
- Events
- Misc

Tagging Event Catchers

```
var e = true|;
var t = Tag.new|;

t: at (e)
  echo("e is true!")
onleave
  echo("e is false!");
[00000173] *** e is true!

e = false|;
[00000185] *** e is false!

e = true|;
[00000218] *** e is true!
```

```
t.freeze;
e = false|;
e = true|;
t.unfreeze;
e = true|;
e = false|;
[00000967] *** e is false!
```

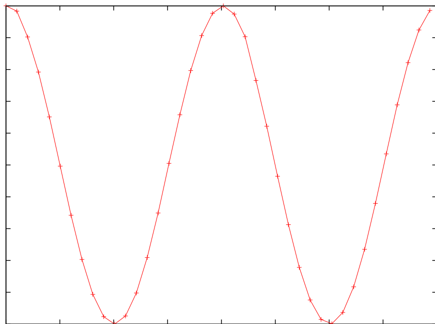
Tagging Event Catchers

```
var e = true|;
var t = Tag.new|;

t: at (e)
  echo("e is true!")
onleave
  echo("e is false!");
[00000173] *** e is true!

e = false|;
[00000185] *** e is false!

e = true|;
[00000218] *** e is true!
```

```
t.freeze;
e = false|;
e = true|;
t.unfreeze;
e = true|;
e = false|;
[00000967] *** e is false!
```

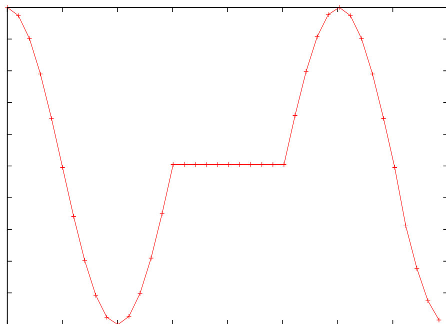
Trajectories

```
var y = 0;
  y = 0 cos:2s ampli:10,
```



Trajectories

```
var y = 0;
t: y = 0 cos:2s ampli:10,
{
  sleep(1.5s);
  t.freeze;
  sleep(1s);
  t.unfreeze;
},
```



Misc

1 Concurrency

2 Tags
- Jobs
- Events
- Misc

Hierarchical Tags

```
var demo = Tag.new;
demo.addChild("dance");
demo.addChild("sing");

function robot.demo()
{
  demo.dance: { /* dance behavior */ }
  &
  demo.sing: { /* sing behavior */ }
};
```

```
demo.dance.stop; // Stop dancing
demo.sing.stop;  // Stop singing
```

```
demo.stop;          // Stop both
```

GOSTAI Robotics for everyone

Hierarchical Tags

```
var demo = Tag.new;
demo.addChild("dance");
demo.addChild("sing");

function robot.demo()
{
  demo.dance: { /* dance behavior */ }
  &
  demo.sing: { /* sing behavior */ }
};
```

```
demo.dance.stop; // Stop dancing
demo.sing.stop;  // Stop singing
```

```
demo.stop;          // Stop both
```

GOSTAI Robotics for everyone

Hierarchical Tags

```
var demo = Tag.new;
demo.addChild("dance");
demo.addChild("sing");

function robot.demo()
{
  demo.dance: { /* dance behavior */ }
  &
  demo.sing: { /* sing behavior */ }
};
```

```
demo.dance.stop; // Stop dancing
demo.sing.stop;  // Stop singing
```

```
demo.stop;          // Stop both
```

GOSTAI Robotics for everyone

Enter/Leave events

```
var tag = Tag.new|;
at (tag.enter?)
  echo("pre");
at (tag.leave?)
  echo("post");
```

```
tag: echo("in");
[00000000] *** pre
[00000000] *** in
[00000000] *** post
```

GOSTAI Robotics for everyone

Enter/Leave events

```
var tag = Tag.new|;
at (tag.enter?)
  echo("pre");
at (tag.leave?)
  echo("post");
```

```
tag: echo("in");
[00000000] *** pre
[00000000] *** in
[00000000] *** post
```

Enter/Leave events

Safety

```
function cook() {
  turnGasOn();
  {
    if (!recipe)
      throw Exception.new("ENORCP");
    if (alreadyCooked)
      return false;
    doStuff();
  };
  turnGasOff();
  return true;
}|;
```

```
var withGas = Tag.new|;
at (withGas.enter?)
  turnGasOn();
at (withGas.leave?)
  turnGasOff();

function cookSafely() {
  withGas:
  {
    if (!recipe)
      throw Exception.new("ENORCP");
    if (alreadyCooked)
      return false;
    doStuff();
    return true;
  };
}|;
```

Enter/Leave events

Safety

```
function cook() {
  turnGasOn();
  {
    if (!recipe)
      throw Exception.new("ENORCP");
    if (alreadyCooked)
      return false;
    doStuff();
  };
  turnGasOff();
  return true;
}|;
```

```
var withGas = Tag.new|;
at (withGas.enter?)
  turnGasOn();
at (withGas.leave?)
  turnGasOff();

function cookSafely() {
  withGas:
  {
    if (!recipe)
      throw Exception.new("ENORCP");
    if (alreadyCooked)
      return false;
    doStuff();
    return true;
  };
}|;
```

Mutual Exclusion

```
var x;
function inc(var who) {
  var from = x;  sleep(500ms);  var to = x += 1;
  echo("%s: %s -> %s" % [who, from, to]);
}|;
```

```
timeout (2s) {
  x = 0;

  every (1s)   inc("1"),
  sleep(0.3s);
  every (1s)   inc("2"),
};
[00000658] *** 1: 0 -> 1
[00000967] *** 2: 0 -> 2
[00001658] *** 1: 2 -> 3
[00001966] *** 2: 2 -> 4
```

```
timeout (2s) {
  x = 0;
  var m = Mutex.new;
  every (1s)   m: inc("1"),
  sleep(0.3s);
  every (1s)   m: inc("2"),
};
[00000657] *** 1: 0 -> 1
[00001158] *** 2: 1 -> 2
[00001659] *** 1: 2 -> 3
```

GOSTAI Robotics for everyone

Mutual Exclusion

```
var x;
function inc(var who) {
  var from = x;  sleep(500ms);  var to = x += 1;
  echo("%s: %s -> %s" % [who, from, to]);
}|;
```

```
timeout (2s) {
  x = 0;

  every (1s)   inc("1"),
  sleep(0.3s);
  every (1s)   inc("2"),
};
[00000658] *** 1: 0 -> 1
[00000967] *** 2: 0 -> 2
[00001658] *** 1: 2 -> 3
[00001966] *** 2: 2 -> 4
```

```
timeout (2s) {
  x = 0;
  var m = Mutex.new;
  every (1s)   m: inc("1"),
  sleep(0.3s);
  every (1s)   m: inc("2"),
};
[00000657] *** 1: 0 -> 1
[00001158] *** 2: 1 -> 2
[00001659] *** 1: 2 -> 3
```

Mutual Exclusion

```
var x;
function inc(var who) {
  var from = x;  sleep(500ms);  var to = x += 1;
  echo("%s: %s -> %s" % [who, from, to]);
}|;
```

```
timeout (2s) {
  x = 0;

  every (1s)   inc("1"),
  sleep(0.3s);
  every (1s)   inc("2"),
};
[00000658] *** 1: 0 -> 1
[00000967] *** 2: 0 -> 2
[00001658] *** 1: 2 -> 3
[00001966] *** 2: 2 -> 4
```

```
timeout (2s) {
  x = 0;
  var m = Mutex.new;
  every (1s)   m: inc("1"),
  sleep(0.3s);
  every (1s)   m: inc("2"),
};
[00000657] *** 1: 0 -> 1
[00001158] *** 2: 1 -> 2
[00001659] *** 1: 2 -> 3
```

GOSTAI Robotics for everyone

Tag: Job Control in urbiscript

1. Concurrency
   - Flow Control
   - Events
2. Tags
   - Jobs
   - Events
   - Misc

GOSTAI Robotics for everyone