Open Roberta (Blockly-based)

Andrea Sterbini – sterbini@di.uniroma1.it

Open Roberta
Simple visual robot/microcontroller programming

Built with Blockly lab.open-roberta.org

Transforms visual programs to Python/Java/C/C++ (depending on which type of robot)

Deploys the program on the robot

Runs the program on the robot (or a simulation on the PC)

Debug the program by stepping/tracing it

Visual interface to the robot configuration details

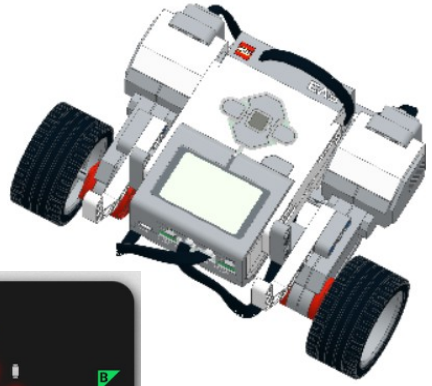
Motors, sensors, wheels geometry, LCD displays, LEDs, ports, shields

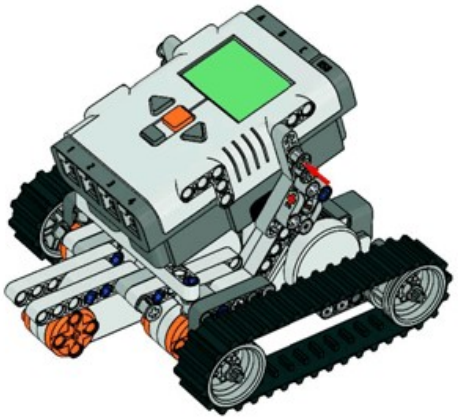
WIKI: https://jira.iais.fraunhofer.de/wiki/display/ORInfo

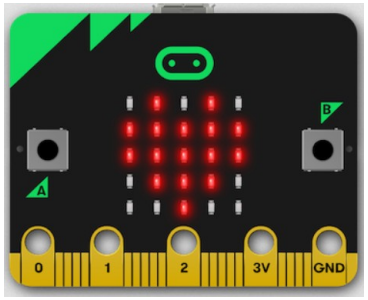
Open Roberta

Many robots and embedded systems supported

NAO, BOB3, Lego WeDo 2, Lego EV3, Lego NXT, Bot'n Roll, Calliope Mini, Micro:bit, Arduino, mBot, senseBox

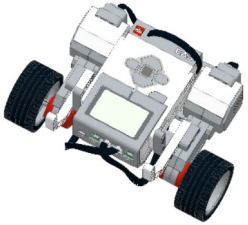


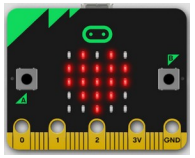




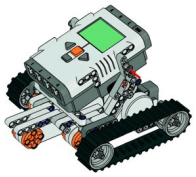
Many generated languages

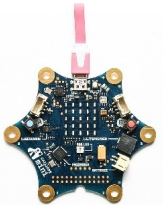
Python: Lego EV3 micro:bit NAO



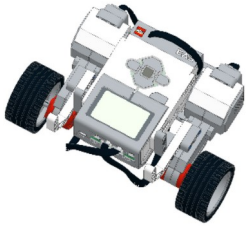


C/C++: Arduino, Bot'n roll, Lego NXT and EV3, BOB3, SenseBox, mBot, Calliope



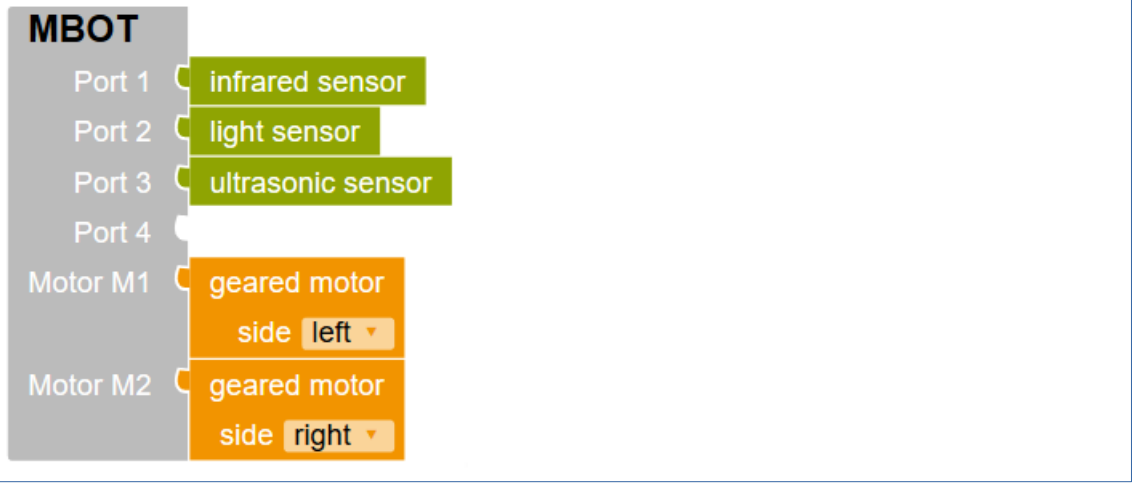


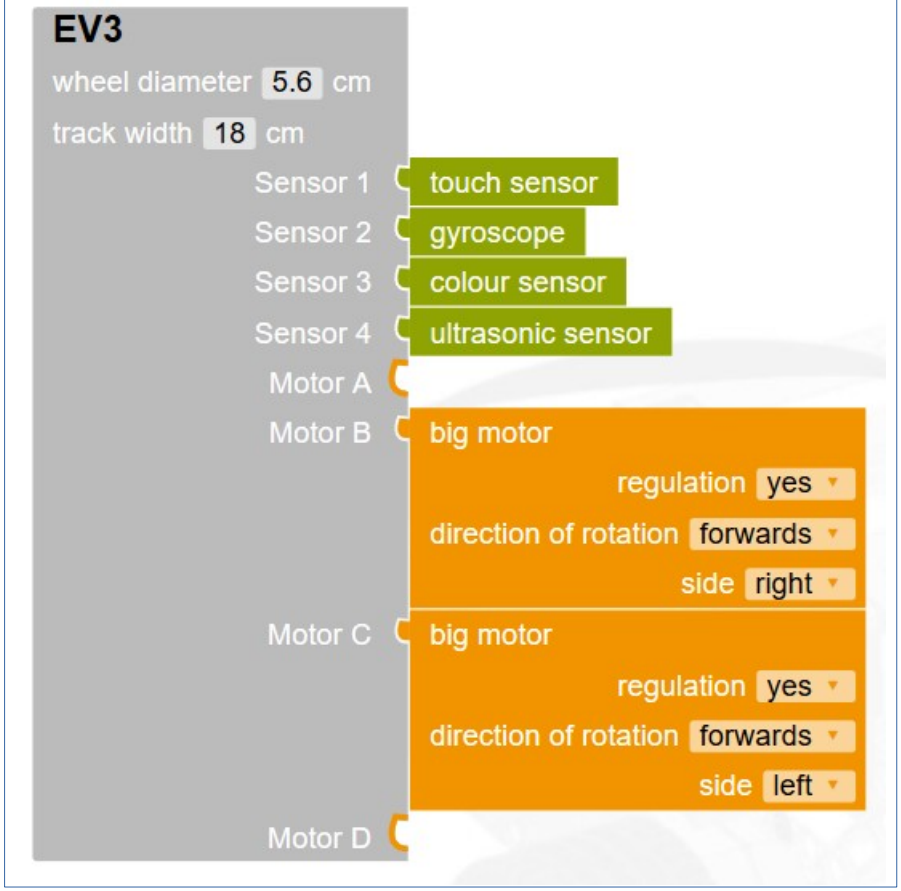
Java: Lego EV3

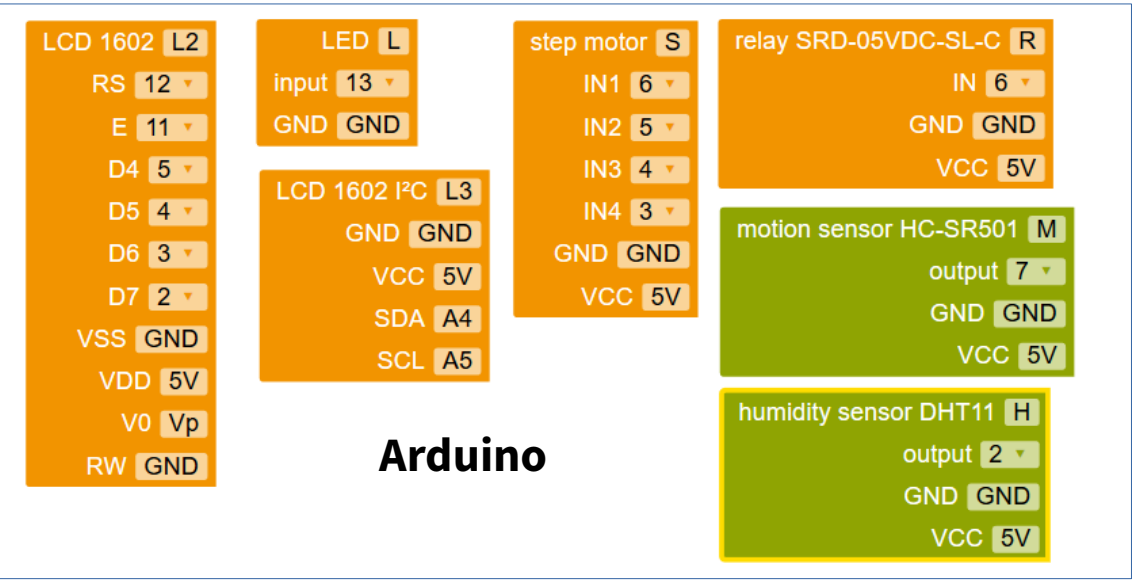


Json: Lego WeDo (runs on PC)

Visual Robot/Microcontroller configuration of the sensort/actuators connected (and where)





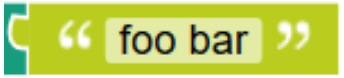


E.G. Configuration in Java (EV3 + Lejos firmware)

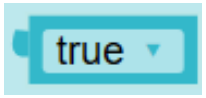
```
public class NEPOprog {
  private static Configuration brickConfiguration;
  private Set<UsedSensor> usedSensors = new LinkedHashSet<UsedSensor>();
  private Hal hal = new Hal(brickConfiguration, usedSensors);
  public static void main(String[] args) {
    try {
      brickConfiguration = new EV3Configuration.Builder()
        .setWheelDiameter(5.6)
        .setTrackWidth(18.0)
        .addActor(ActorPort.B, new Actor(ActorType.LARGE, true,
                                  DriveDirection.FOREWARD, MotorSide.RIGHT))
        .addActor(ActorPort.C, new Actor(ActorType.LARGE, true,
                                  DriveDirection.FOREWARD, MotorSide.LEFT))
        .build();
...
```

Data types

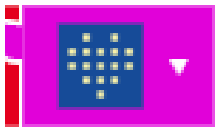
Number 

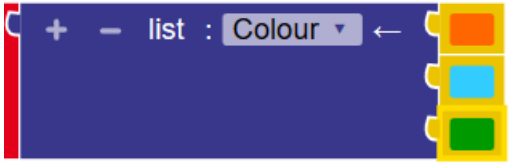
String 

Colour 

Boolean 

Connection

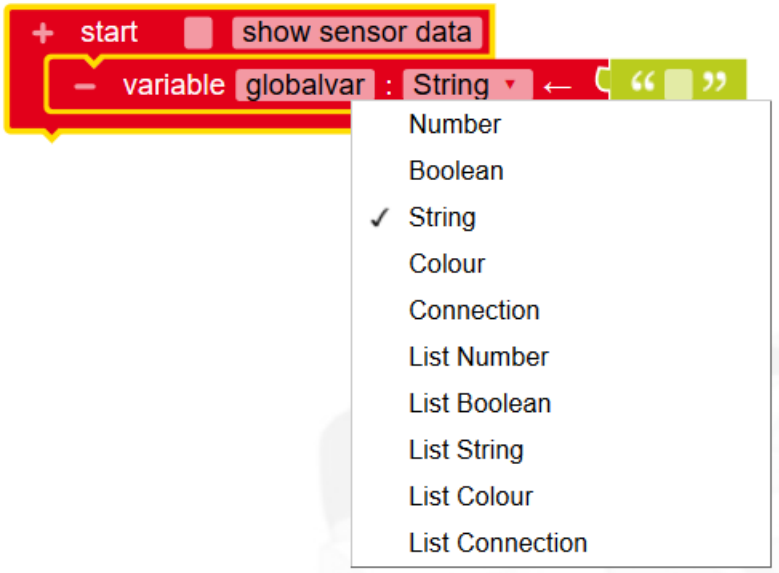
Image 

List of <T> 

(same type for all elements)

Variables and arguments are typed
(the connector is coloured)

Data types are visually enforced
(cannot join if the type is wrong)



Execution model: single thread

Single thread of execution (main program/main loop)

New Functions?	YES	
Global variables?	YES	(defined only at main level)
Local variables?	YES?	(defined as function's arguments)
Messages?	NO?	(but robots can communicate over BT)
Events?	NO	

Events simulated by polling the sensors + "when"

Lego EV3 robots can connect via BT and exchange text messages

Other robots can communicate over serial wires

“Advanced-enough” programming

Counted Loops, Foreach,

Repeat until, Repeat while

Continue, break

Wait N ms,

Wait until condition …

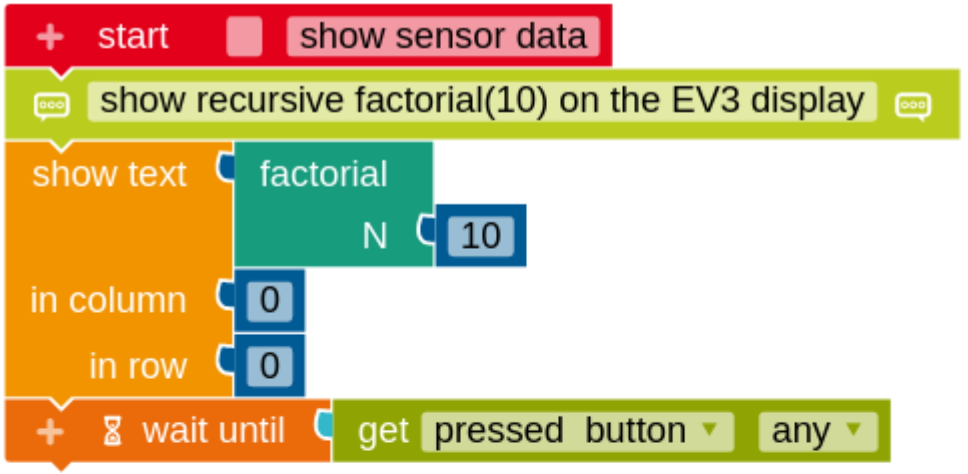
or other condition … or else

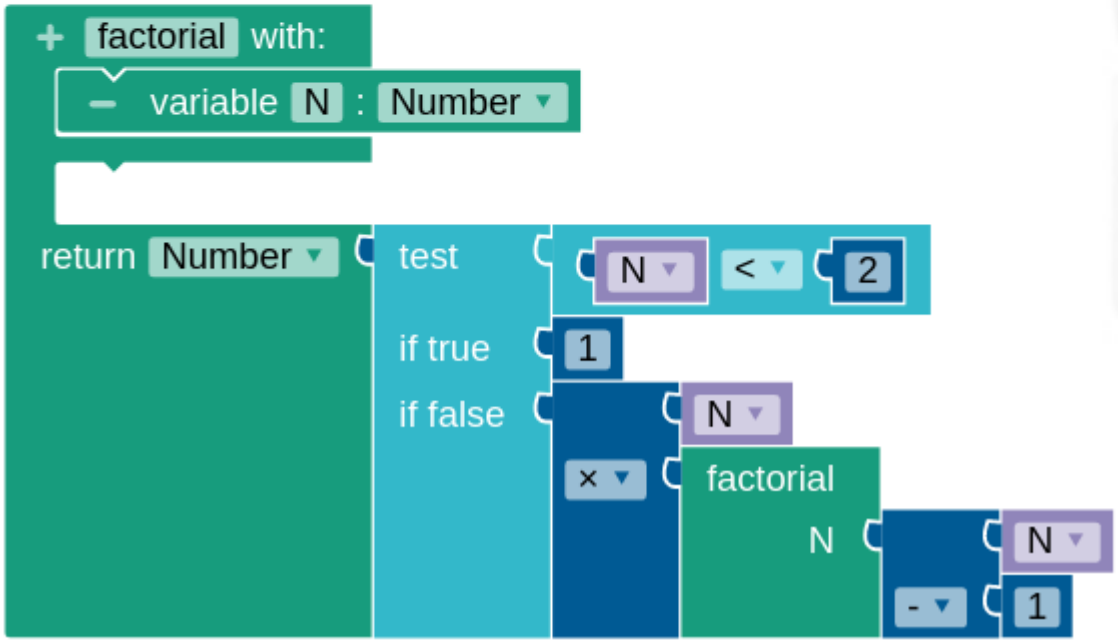
If, if-else, if-elif-…-else

Constrain value between

Recursion? YES

Local variables as arguments(!)





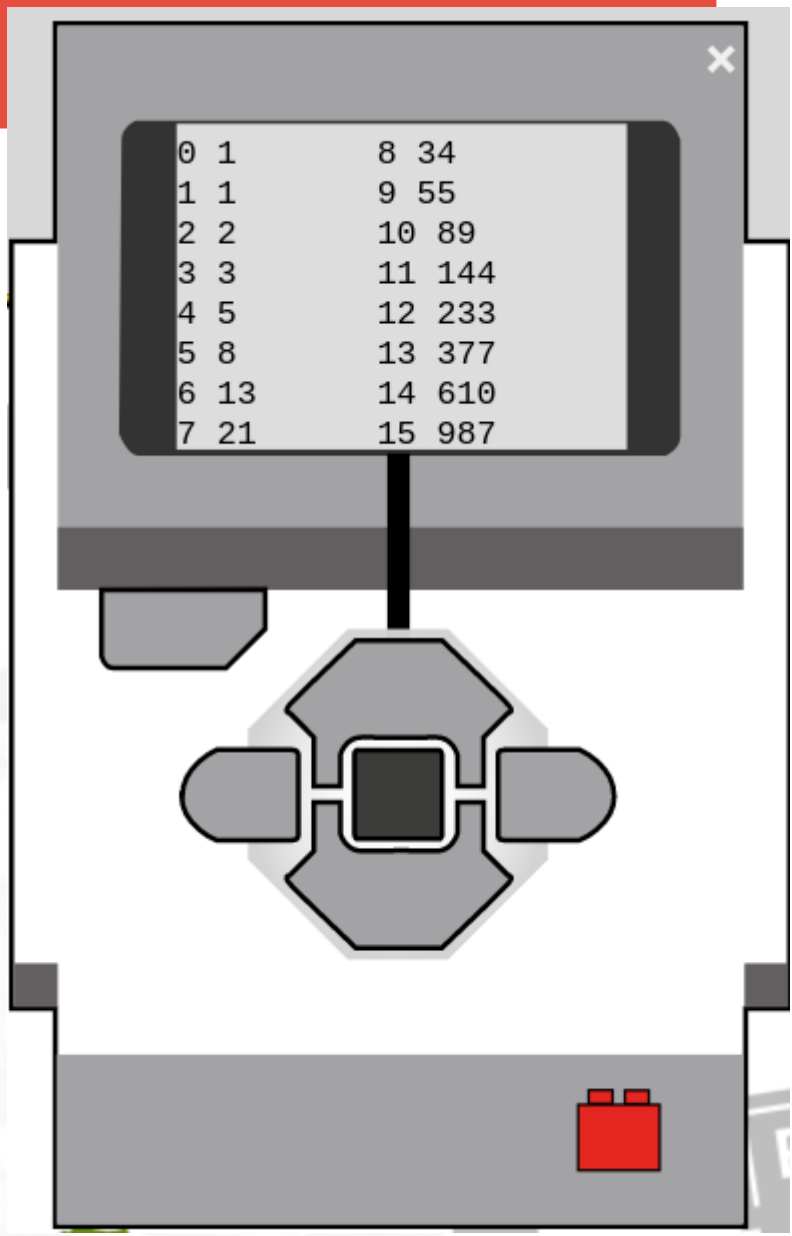
Example: efficient recursive Fibonacci (forward loop simulation)

F value

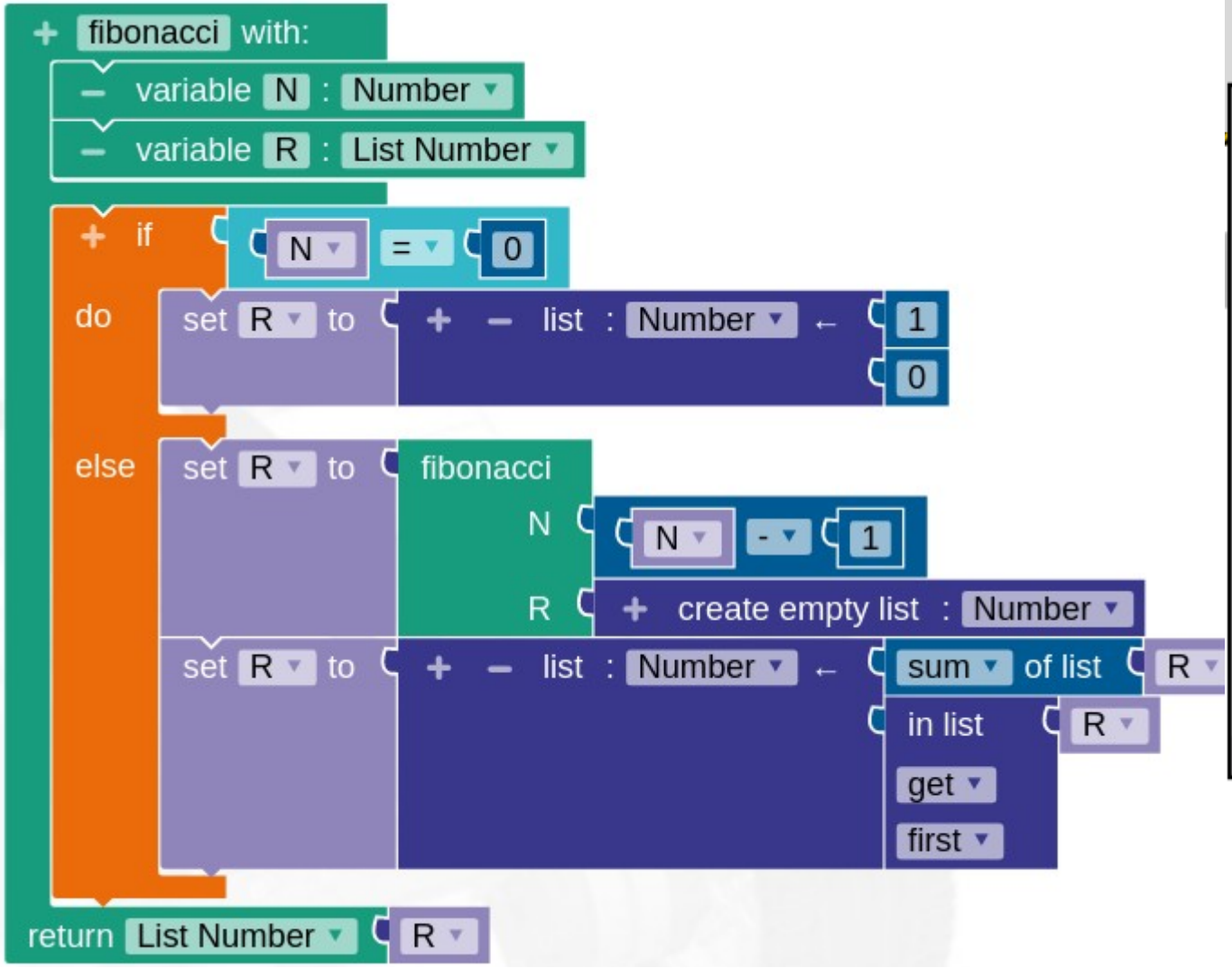
previous value

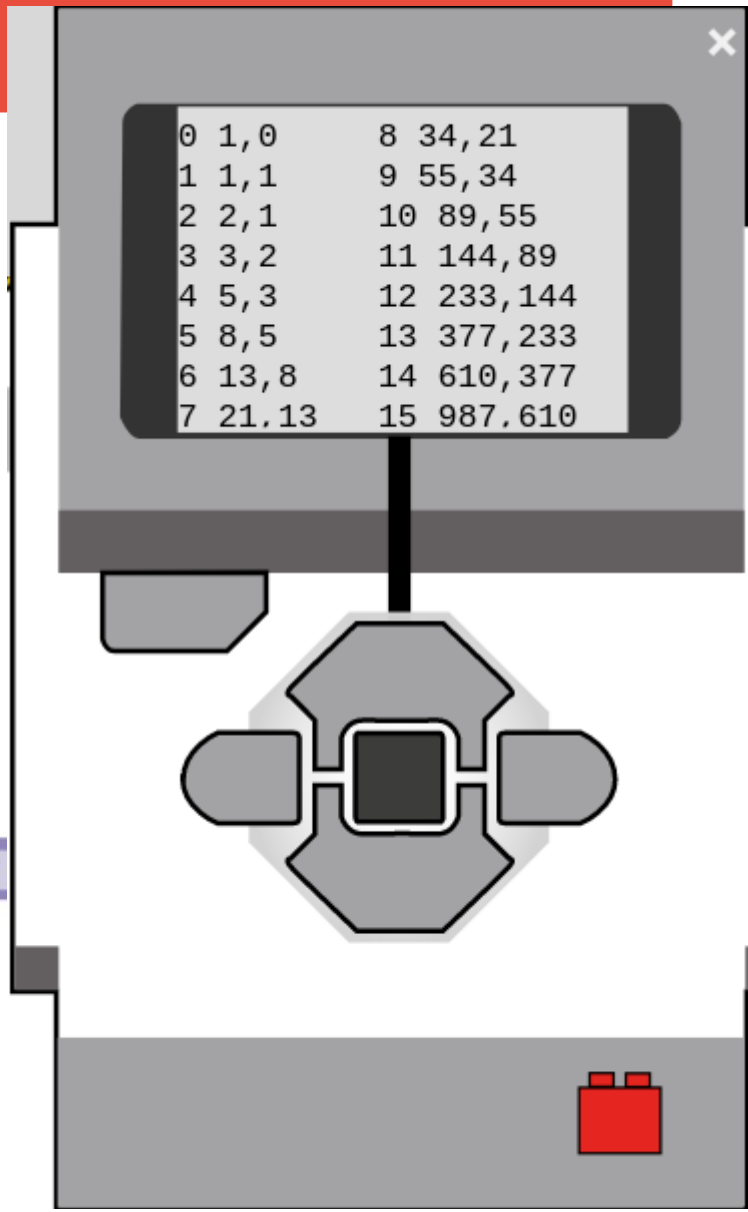
```
+ fibonacci with:
    – variable N2 : Number

  return Number   fibonacci_ N N2  F 1  Fold 0
```

```
+ fibonacci_ with:
    – variable N : Number
    – variable F : Number
    – variable Fold : Number

  return Number   test  N = 0
                  if true  F
                  if false  fibonacci_ N  N - 1   F  F + Fold   Fold  F
```

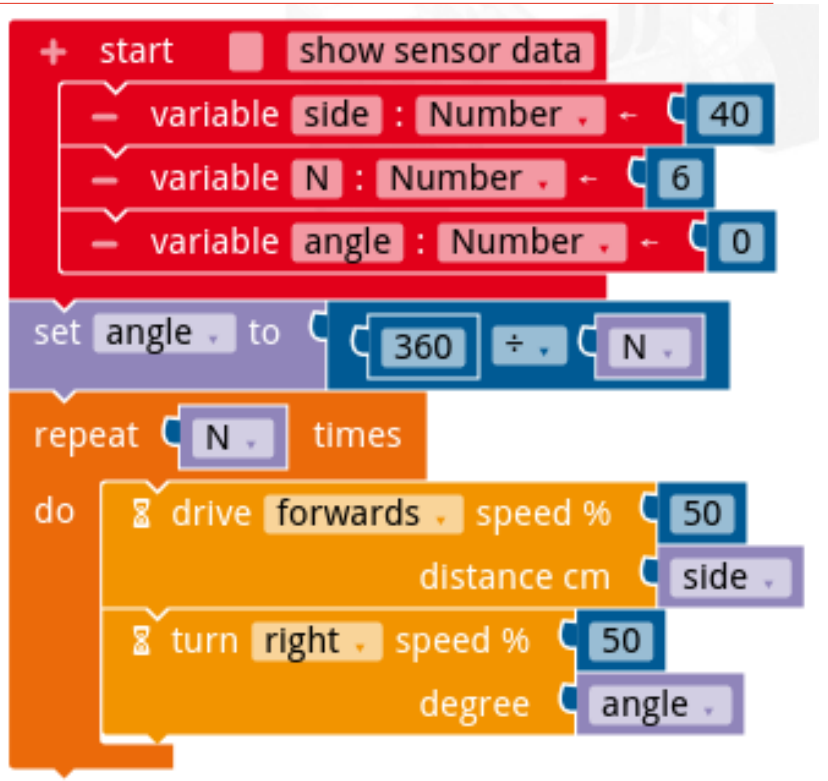


Example: efficient recursive Fibonacci (backward loop simulation)





Example: polygon movement




```
// MAIN code

  float ___side = 40;
  float ___N = 6;
  float ___angle = 0;

  public void run() throws Exception {
    ___angle = 360 / ((float) ___N);
    for ( float ___k0 = 0; ___k0< ___N; ___k0+= 1 ) {
      hal.driveDistance(DriveDirection.FOREWARD, 50, ___side);
      hal.rotateDirectionAngle(TurnDirection.RIGHT, 50, ___angle);
    }
  }
}
```

Our experience: 9 and 10 y/o students in K4 nd K5

1) Role play on a grid + instructions with arrows repetitions and conditions

2) small programs on Scratch with turtle graphics

3) small programs with Lego EV3 robots in Open Roberta

Pay attention to:

- Network connectivity (if possible install the software locally)
- loose wires in the robot that raise strange exceptions
- Bluetooth (use wifi, it’s more stable and supported)
- local teachers that don’t know how to help

Local install (for a better network access)

Open source

Available on https://github.com/OpenRoberta/openroberta-lab

Java based, built with Maven

You can enable/disable separately each module/Robot

You can run the server on your laptop in class and share your wifi

Robots and PC browsers in the class connect by wifi to your laptop

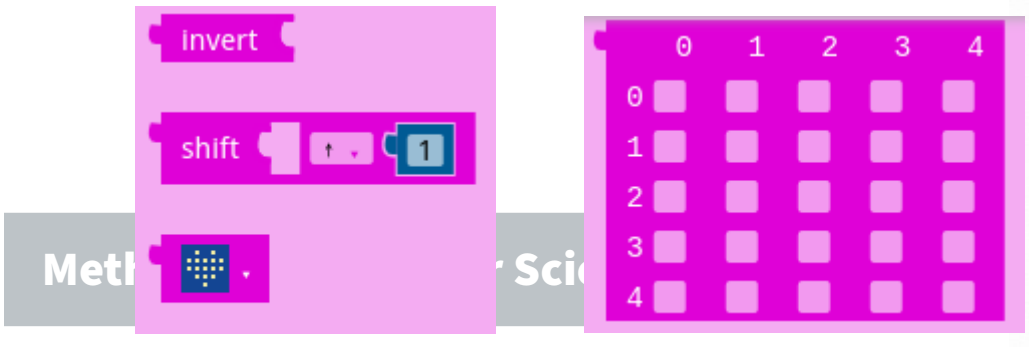
Available also for Android

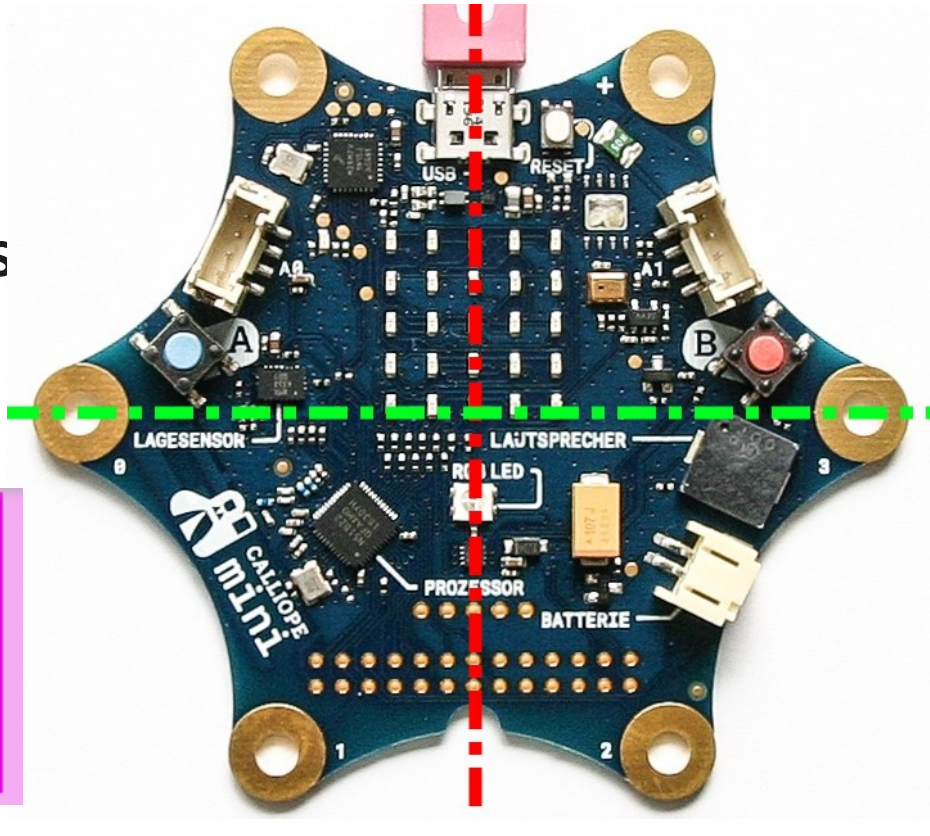
Calliope mini microcontroller: a lot of sensors

Sensors: buttons, tilt, compass, temperature, light, sound intensity, gyroscope, accelerometer, humidity, ultrasound, external analogue sensors (e.g. colour)

Actuators: 5 x 5 LED matrix
external 4-digits display
serial port to terminal
external motor controllers

Special blocks for 5x5 LED matrix





NAO: a small “dancing” robot

Predefined movements (tai chi, wave, blink

Walk to, hand movements in space, …

Record video or picture

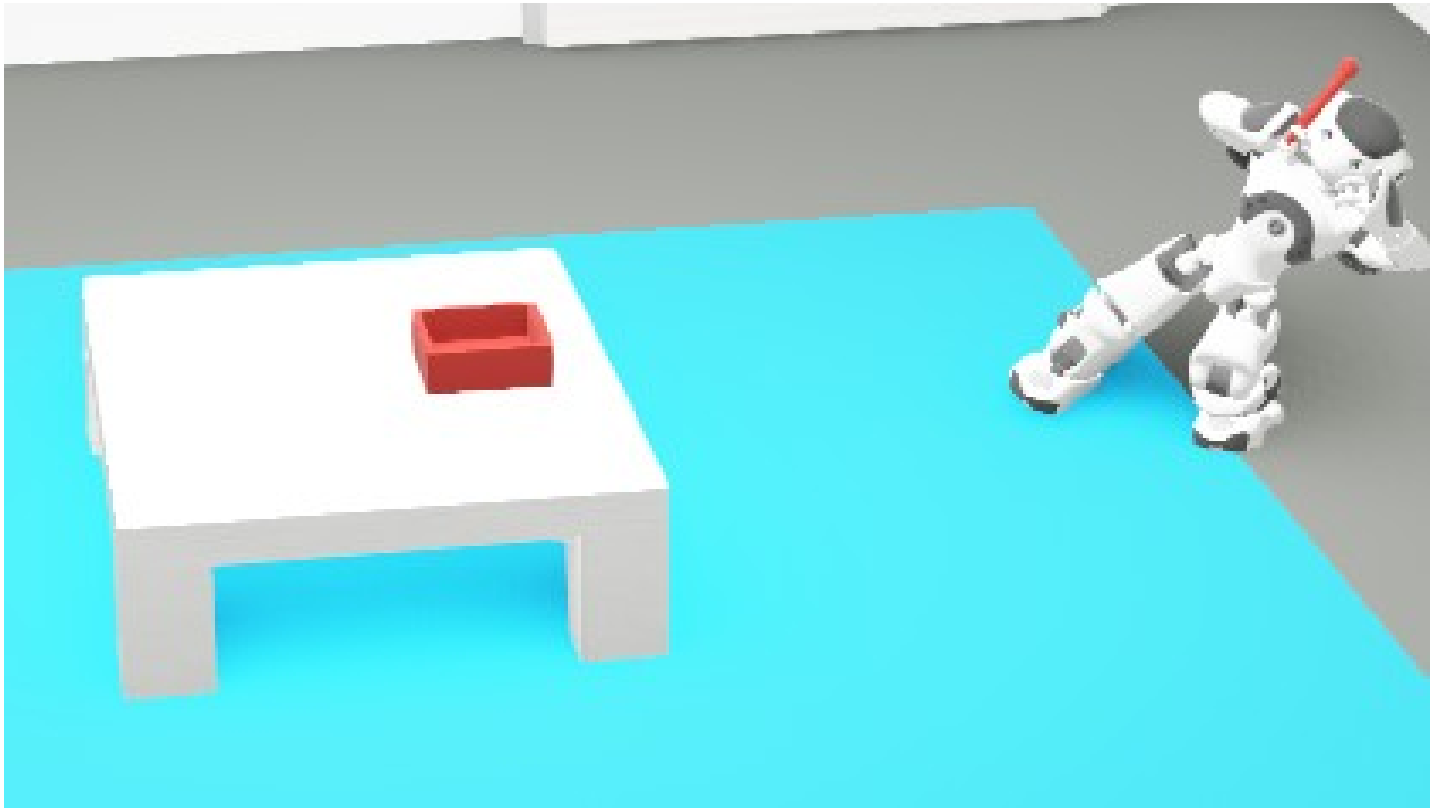
Remember/recognize face

Play sounds, speak (text to speech)

Programmed in Python

3D simulation in browser

E.G. making a Tai chi move



Demo

Demo