JCSPre: The Robot Edition To Control LEGO NXT Robots

Jon KERRIDGE,
Alex PANAYOTOPOULOS
and
Patrick LISMORE

NAPIER UNIVERSITY
EDINBURGH

Motivation

- Can JCSP be implemented on a LEGO NXT?
- Why not just use the Transterpreter approach?
- Wanted to use Java
  - Relates better to previous experience
  - More people use Java than occam
  - Influence people in the wider community
  - Might convince people that the Java thread model can be avoided
  - Demonstrate reuse in a fun environment
  - Promote plug and play capability
  - Eclipse IDE can be used directly

Questions Posed

- Does it fit on a LEGO NXT robot
- Which subset of JCSP is required
- Which Java environments are available
- How many processes can be accommodated
- Can we utilise existing JCSP design patterns
- Can we exploit further capabilities of the LEGO NXT robot, such as Bluetooth

Java Environment

- Obvious one is LeJOS
  - Already contains LEGO NXT abstractions
  - Appears to be small
  - Already contains a threading model used by JCSP
  - Takes account of the processors used in LEGO NXT
  - Still being developed
- Other JVMs
  - IBM J9 and NSIcom CrE-ME
    - Designed for embedded CDC market
    - Designed for Windows CE
    - Not available for LEGO NXT processor combination.
    - Fully functional

Choosing JCSP Package Subset

- org.jcsp.lang
  - Provides the fundamental CSP capabilities
- org.jcsp.util
  - Provides channel buffering capability
- org.jcsp.awt
  - Provides a ‘parallel’ AWT capability
  - Not required on LEGO NXT
- org.jcsp.net
  - Provides the network capability
  - Not required (yet) on LEGO NXT

LeJOS JVM Capabilities

- LeJOS JVM is **not** a full Java implementation
  - Limited garbage collection
  - Variables of type long cannot be manipulated
  - The class Class is not implemented
  - The LeJOS API is missing many classes considered fundamental to the core Java API.
  - The LeJOS classes are closer to the CLDC Java specification

Included JCSP Fundamentals

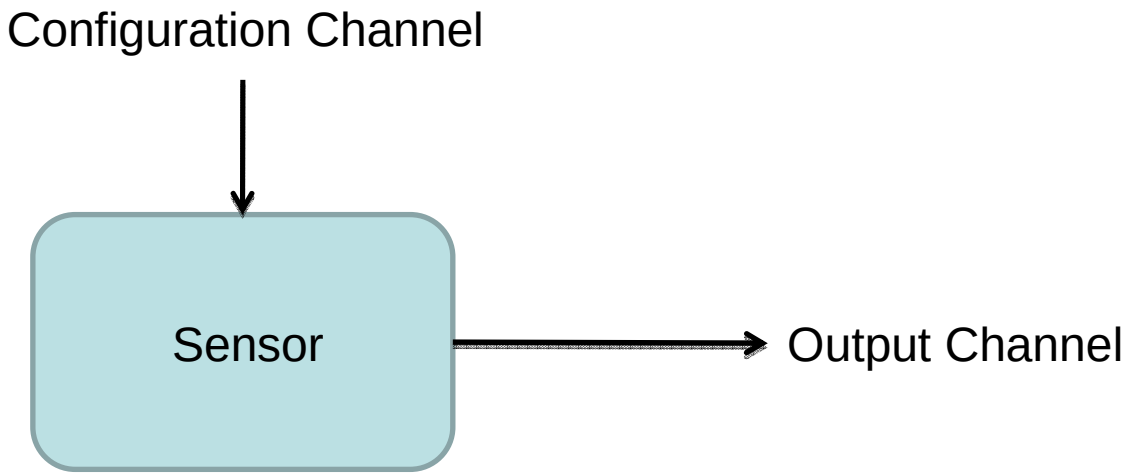
- Required system structuring features
  - CSProcess,
  - Alternative (requires Barrier and Guard),
  - Parallel (requires ParThread),
  - CSTimer,
  - Skip
- Communication (Object and int versions of each)
  - One2One Channel and One2One ChannelImpl
  - ChannelInput and ChannelOutput
  - AltingChannelInput
  - The Any, Call and Connection versions **not** included

LeJOS Abstractions

- Sensors (two types)
  - A-D sensors (based on previous RCX robot)
  - I2C sensors (newer , more esoteric sensors)
- Motors
  - Movement (angular and power)
  - Tachometer input interface
- Communications and Other devices
  - Bluetooth
  - GPS
  - Keyboards
  - Interaction with ‘host’ PC (for debugging)

Sensor Process Architecture

- Each sensor type is implemented with a channel interface, hiding the underlying LeJOS abstraction
- The associated event listener is contained within the sensor process

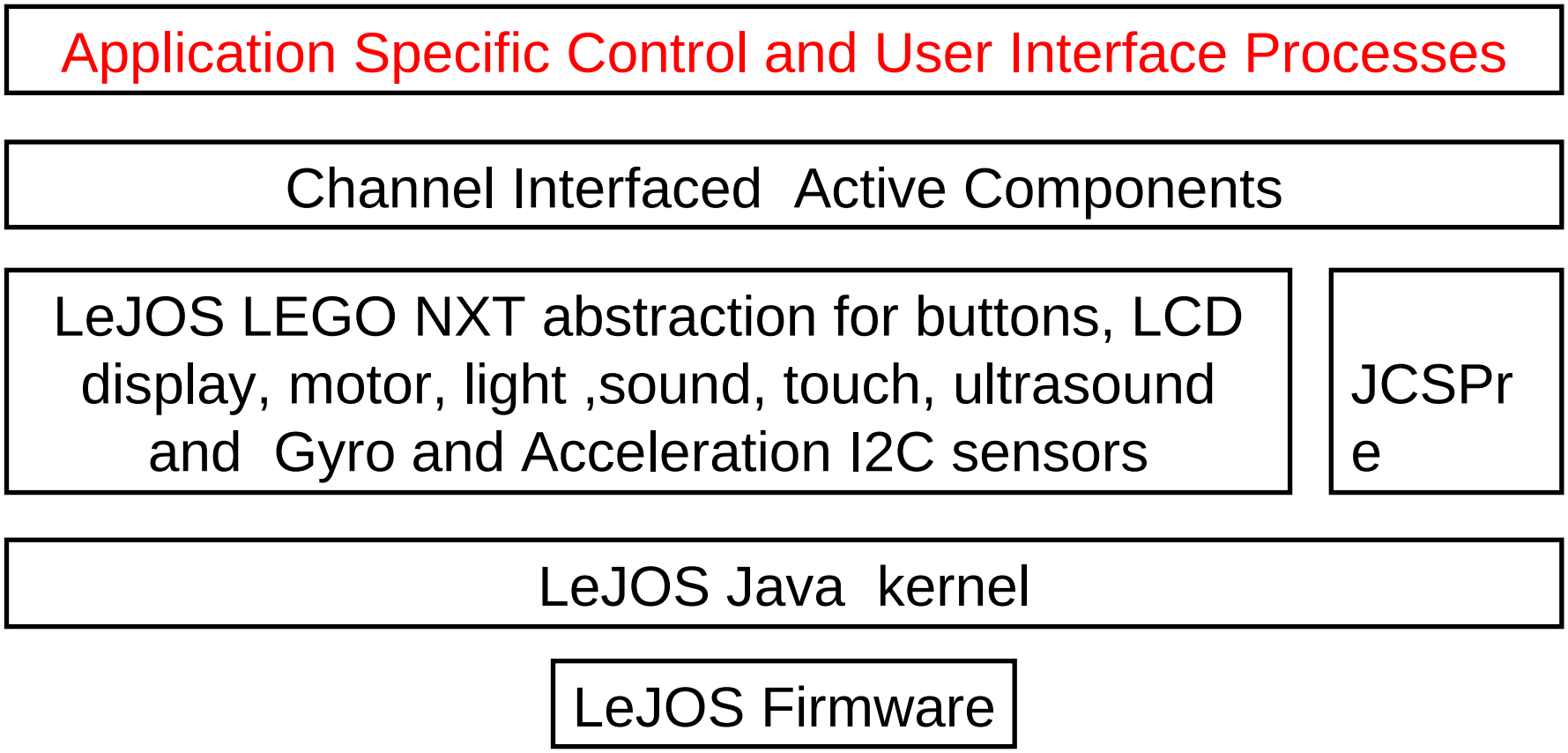


Motor Process Architecture

- Input Channel to set motor speed
- Configuration Channel to set operating parameters
  - Halting behaviour – Stop or Float
  - Speed input – rotational or power based

- Termination of all processes is achieved by sending a known value, Integer.MAXINT, to the configuration channel or input channel if available.

Architectural Framework



Active Sensors

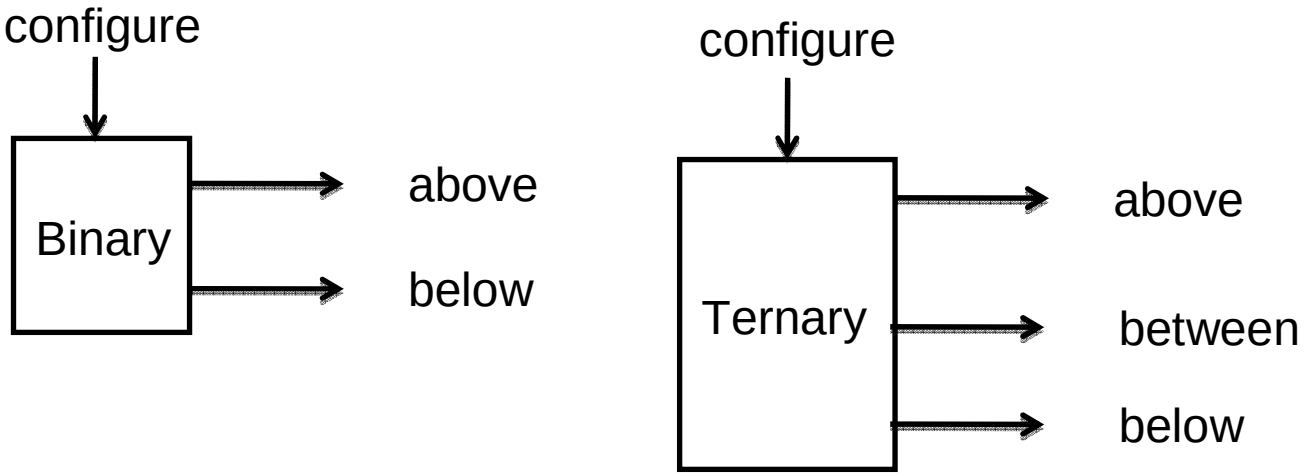
- Implement the Interfaces
  - CSProcess
    - Requires a run() method to be executed as part of a Parallel
  - SensorPortListener
    - Requires a stateChanged() method that is called whenever an A to D type sensor value changes
    - I2C sensors do not use this interface and the active sensor process has to periodically read the device ports to determine its current value
- A delta value configured into a sensor to ensure that only changes of sensor value greater than delta are written to the sensor's output channel.

The JCSPre packages

- lang
  - Provides the basic JCSP capability
- io
  - Provides the sensor and motor abstractions
  - Provides the Bluetooth receive and transmit processes
- plugnplay
  - Provides multiplexers, and other processes used to test the system's operation and which can be used in designs
- filters
  - Provides threshold filters that can be used in designs
- rconsole
  - Provides processes that communicate message strings to a PC over usb or bluetooth communication links.

Threshold Filters

- Binary and ternary filters are provided.
  - The input value is output on one of the output channels depending on its relationship to threshold values
  - Threshold values communicated on the configure channel, or in the process constructor



Feasibility Testing (LeJOS)

- Underlying LeJOS thread model could spawn 160 child processes from a single parent
- A simple variant placed a single integer in each spawned child thread.
  - The number of child processes reduced to 90.
- The number of threads is dependent on the size of each thread.

Feasibility Testing (JCSPre)

- Parallel
  - An integer Producer process, followed by N Add processes that input a number, add a constant to it and then output the new value. The final value is output to the LEGO NXT LCD screen by a final process.
  - N = 78 was the maximum number of ‘Add’ processes.
- Alternative
  - N processes each output to a multiplex process that inputs from each input channel once in each cycle.
  - N = 76 was found to be the upper limit

Feasibility Testing Implications

- System does seem capable of supporting sufficient processes provided
  - A lot of very small processes are not used
  - Especially when ‘hidden’ inside other processes
    - Delta and DynamicDelta from ‘standard’ JCSP would be inappropriate
    - use the ProcessRead and ProcessWrite capabilities of JCSP

Bluetooth Communication (NXT)

- LEGO NXT robots are provided with Bluetooth capability
- Processes in the JCSPre.io package provide Bluetooth Transmit and Receive capability that permit communication between
  - LEGO NXT and a PC
  - LEGO NXTs
- The communication is implemented using the Bluetooth Serial Port Protocol (btspp)

Bluetooth Communication (PC)

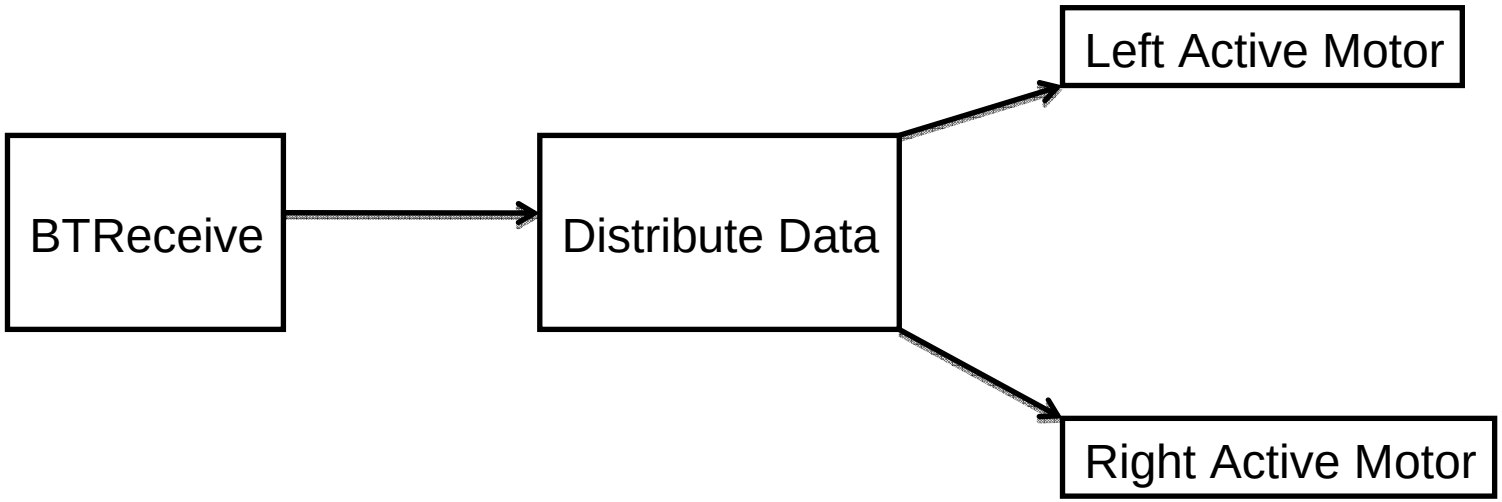
- A Bluetooth Connection process has been provided to interact with a JCSP system running on the PC
- Uses the ActiveSerialPort concept (CPA- 2004)
- Uses a javax.comm implementation appropriate to the platform
  - Implemented on PC using Windows XP
  - Implemented on PDA using Windows Mobile

Designing Robot Controllers with JCSPre

- The majority of the required processes already exist
  - Contained within the JCSPre packages
    - lang, io, plugnplay, and rconsole
  - ‘Simply’ a matter of connecting them together with channels
- Implementer designed control process
- Systems can be developed totally within an Eclipse environment
  - The code can also be uploaded into the LEGO NXT from within the Eclipse environment.

User Interface Steerable Robot

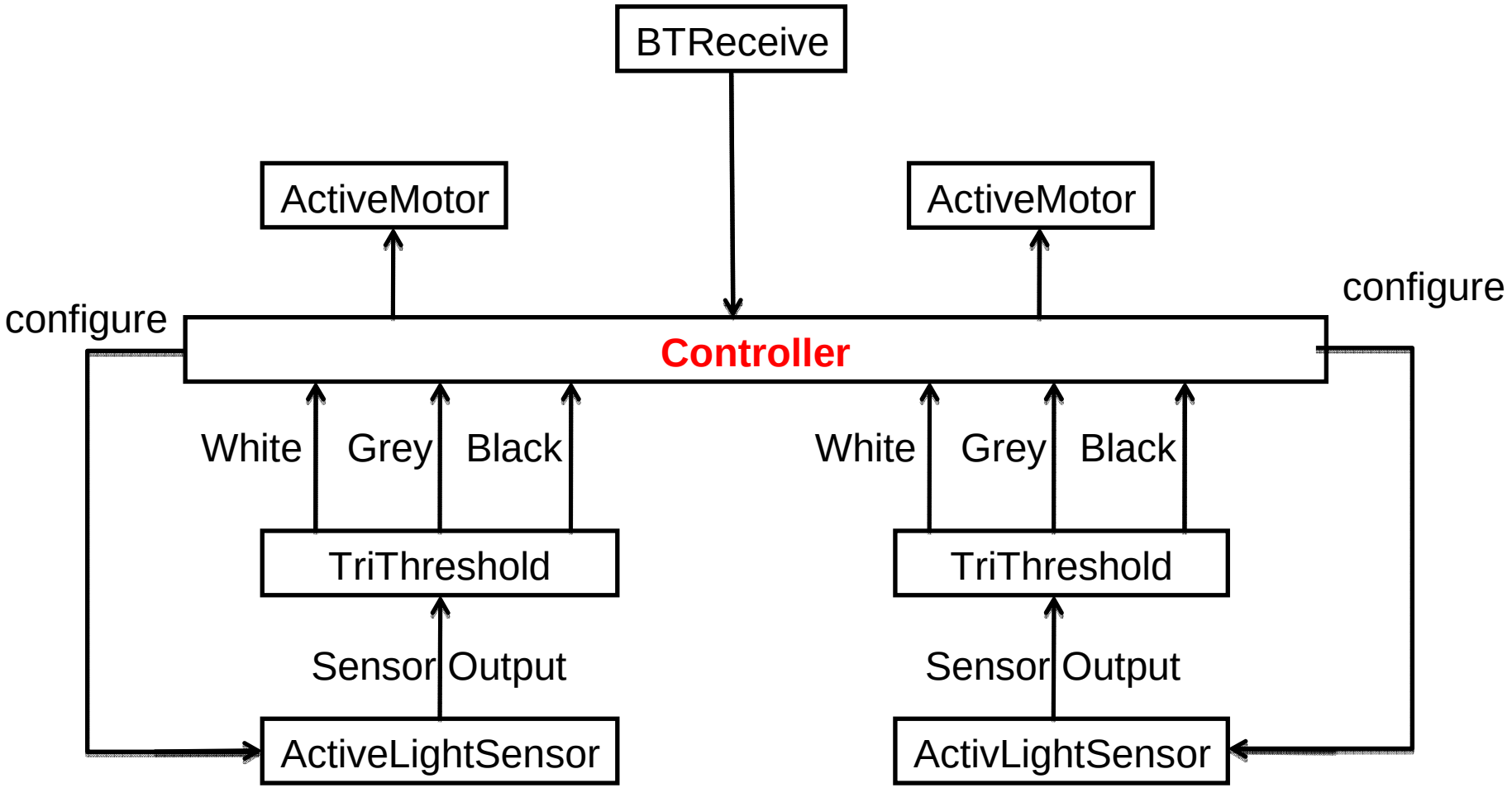
- PC or PDA based user interface sends motor movement data to robot using Bluetooth.
  - Data comprises pairs of speeds for left and right motors
  - Only the Distribute Data process was written specially



Steerable Robot

- Robot has two light sensors pointing at ground
- Robot is steered by sensing either a white or black line across its path.
- The sensors align themselves with the line
  - If it is white line it just continues
  - If it is a black line the robot reverses
- Placement of the lines in the path of the robot mean that it can be made to follow a pre-determined route.
- The initialisation of the Robot is carried out using an interface on PC connected by Bluetooth

Steerable Robot Design

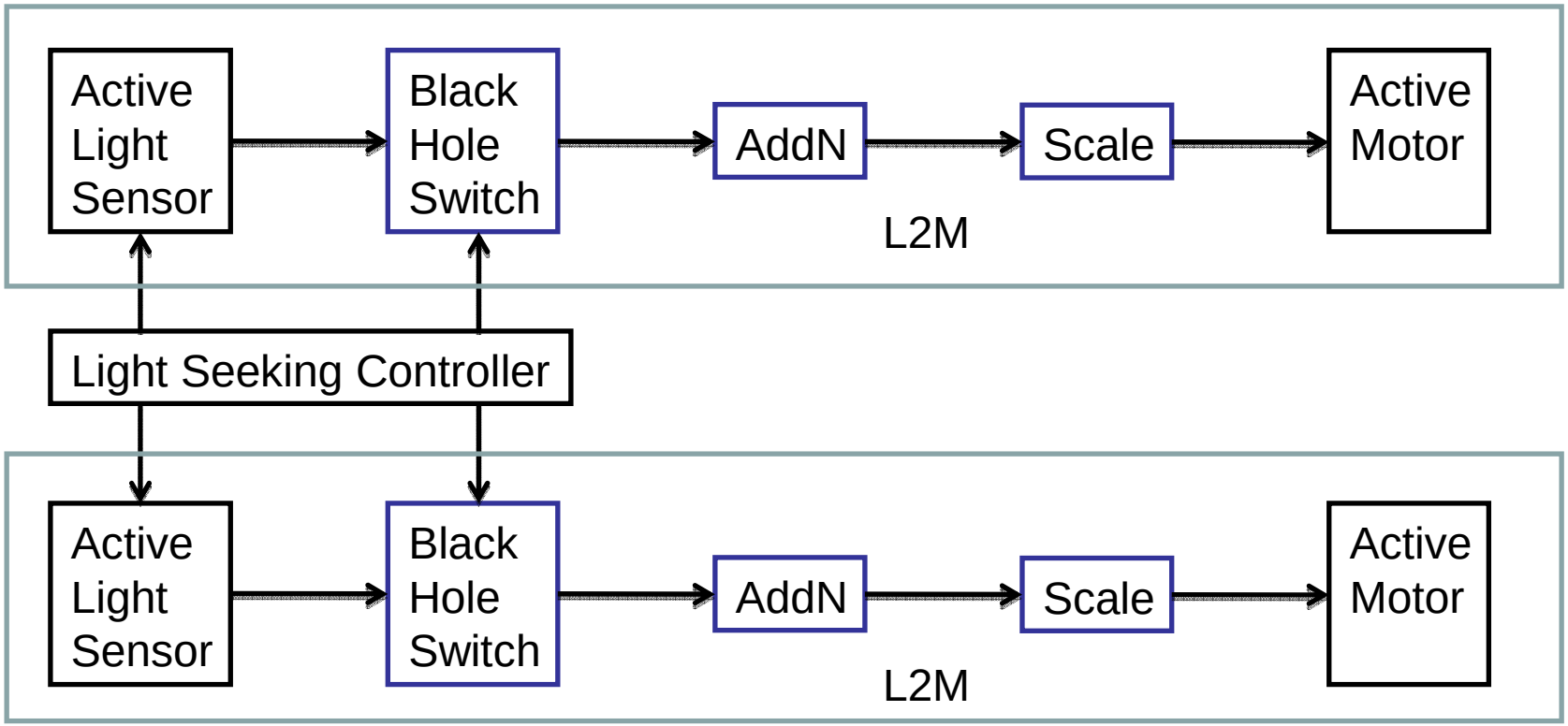


Subsumption Architectures

- Brooks Subsumption architecture describes a robot control environment whereby the behaviour of a robot can be broken up into layers of increasing complexity.
- Lower levels can have their outputs subsumed by higher levels
- This has been achieved in a subsumption package
  - This does NOT use the subsumption package available in LeJOS, which is a very partial implementation
  - Contains a subsumption pattern that can be modified to specific requirements

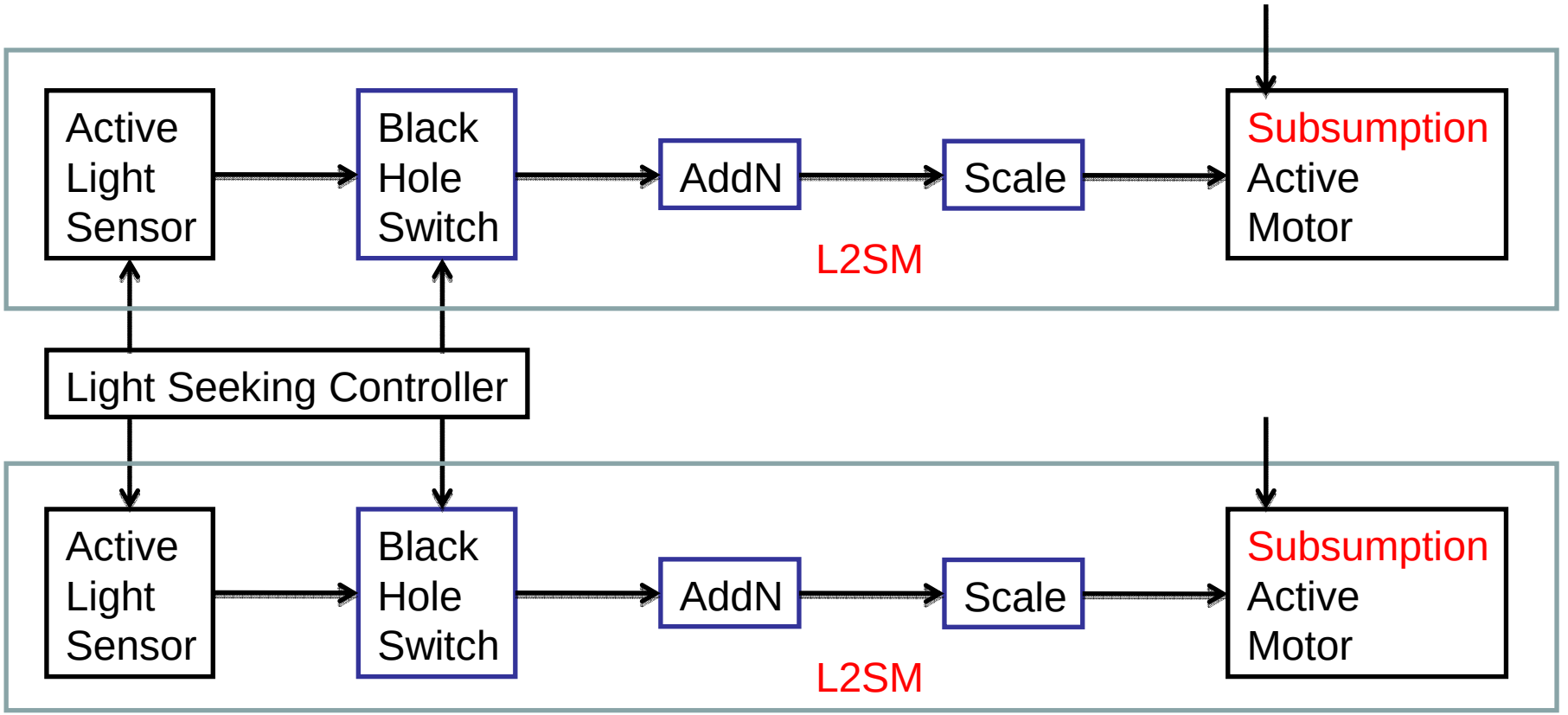
Light Seeking Robot

- Light SeekingRobot
  - Comprises two cross-coupled Light to Motor (L2M) controllers; due to Braitenburg



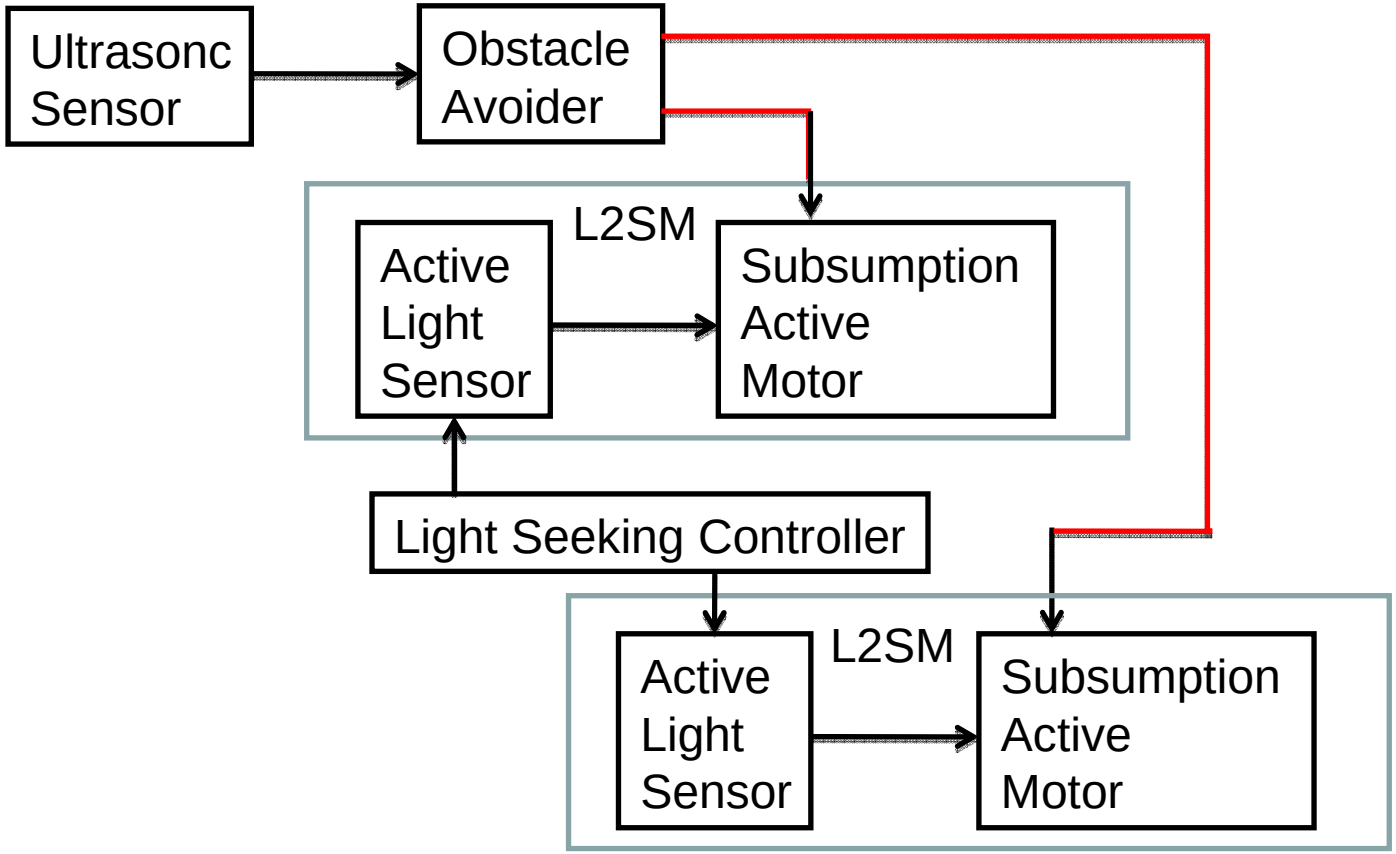
Adding Obstacle Avoidance

- The Active Motor behaviour has to be subsumed by a higher level behaviour: Obstacle Avoidance



Obstacle Avoiding Architecture

- Ultrasonic Sensor added to detect obstacles and impose an avoidance behaviour: unequal reversing



Subsumption Design Pattern

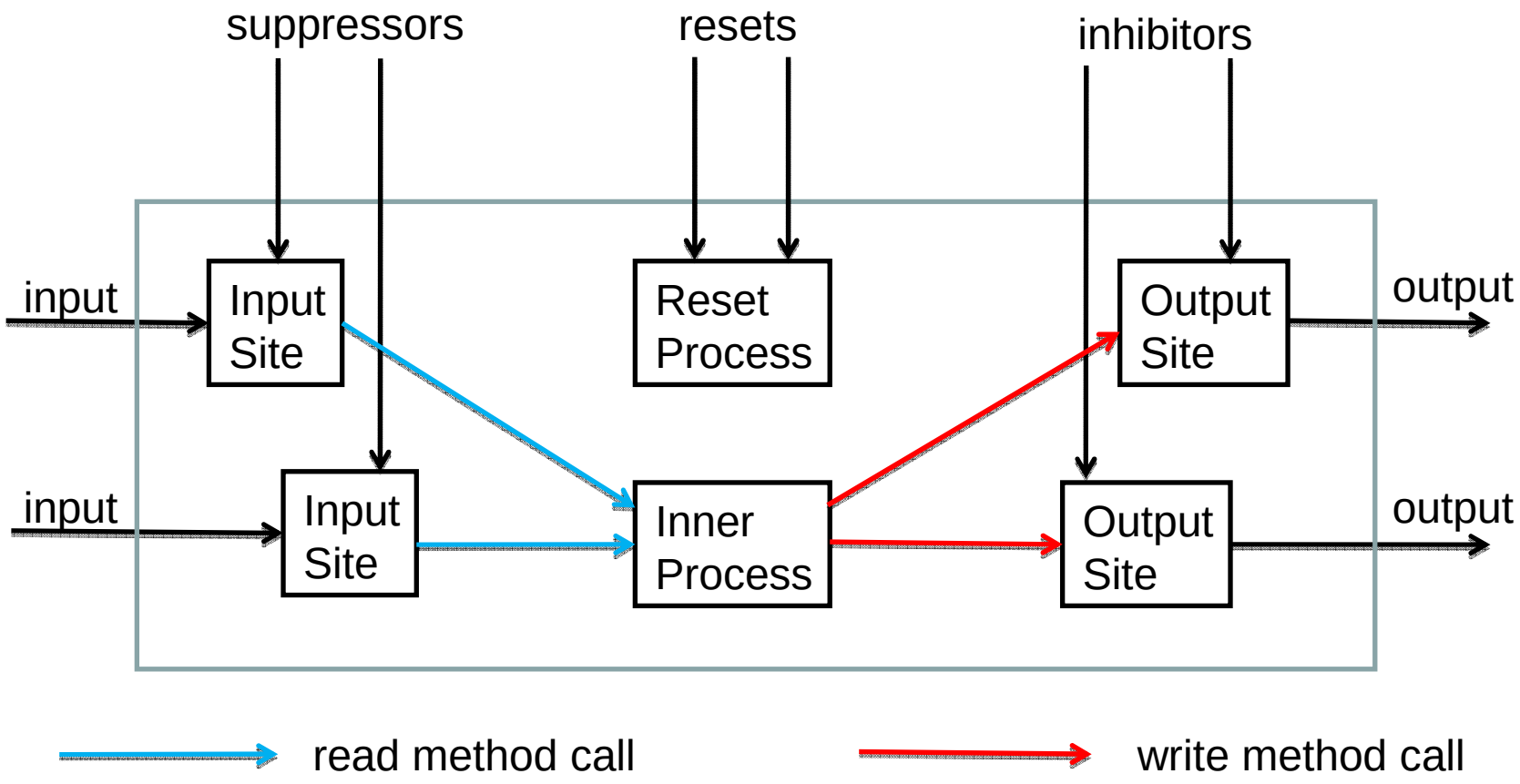
- The Subsumption Process architecture is provided as a design pattern that has to be specialised to each specific application
- Comprising
  - Input Sites with an input and several suppressor channels
  - Output Sites with an output and several inhibitor channels
  - An optional Reset Process with reset channels
  - An Inner Process that implements the control aspect

Input and Output Sites

- The number of Input and Output Sites depends upon the application.
- Each site holds a single data value
- The Inner Process can access the site data value by means of *read* and *write* methods.

- The Reset Process is able to return the data values in the Input and Output sites to a predefined value, by using the site data value access methods.

Subsumption Pattern Architecture



Conclusions

- An implementation of JCSP on a LEGO NXT has been achieved
- Sufficient processes can be utilised to enable the construction of relatively complex control systems
- It could (will) provide a vibrant way of teaching Communicating Process Architecture concepts to students
- It is fully integrated into the Eclipse IDE
- **The use of Java makes it more accessible to more people**

Future Work

- Produce a publicly available library
- Develop the teaching notes
- Implement a networking capability based upon Bluetooth so that several robots can communicate through a Bluetooth Server with each other
  - Using Kevin Chalmer's lightweight network protocol
- Implement a graphical design tool for the Eclipse environment that allows graphical system design
- Determine the limit to control system complexity under more complex operating conditions