An Overview Of The Microsoft Robotics Developer Studio

Martin Rupp
SCIENTIFIC AND COMPUTER DEVELOPMENT SCD LTD

Goals of the software

MRDS provides a platform for developing applications involved with robotics and basic service components in order to allow every designer with an interest in developing robots to easily program the necessary procedures to run robot hardware, without being restricted by the underlying complexity of the hardware system.

Microsoft Robotics Studio (MRDS) is an Integrated Development Environment (IDE) developed by Microsoft for the development of robotics projects. It is based partly on CCR (Concurrency and Coordination Runtime), a .NET framework for managing concurrent parallel tasks,

The software is compatible with several robots including:

- ABB Group Robotics
- Adept Mobile Robots
- Aldebaran Robotics
- Arieh Robotics
- CoroWare CoroBot
- KUKA Robotics Educational Framework
- Parallax Robots
- Robosoft's robots
- Kondo KHR-1
- RoboticsConnection Traxster
- uBot-5
- Amherst
- Vex Robotics
- CNRobot
- Robotino

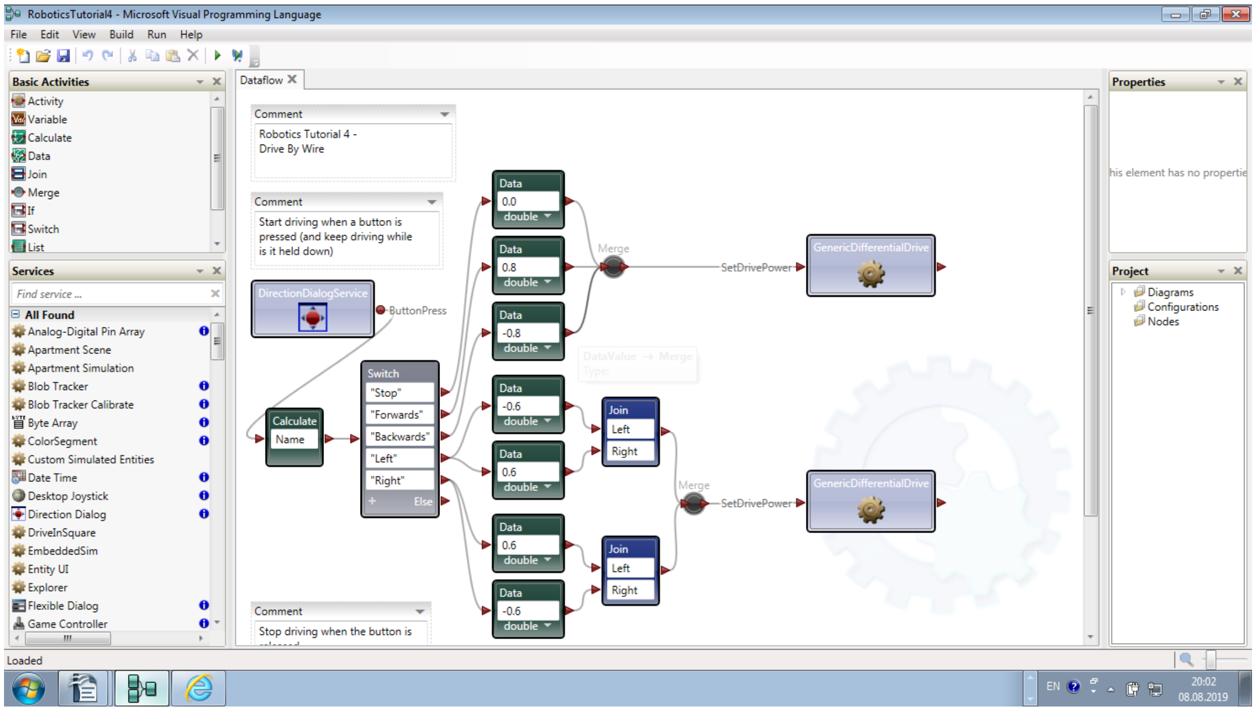
The main components

The Microsoft Visual Programing Language

The Robotics studio is based on the Microsoft visual programming language or MVPL. It is a “true” *visual* programming language because “logical” items can be dragged and dropped in such a way that a dataflow is graphically created.

As part of MRDS, Microsoft supplies different Robotics projects samples that support existing robots (see above list) .

MRDS code cannot be directly compiled and loaded to a robot. There must be a Windows device in the chain between the code and the robot. Either in the robot or between the robot and the code.



Note that C# can be used as the development language - in fact it is the primary target of the IDE but for many users it will be far more difficult to use than the MVPL.

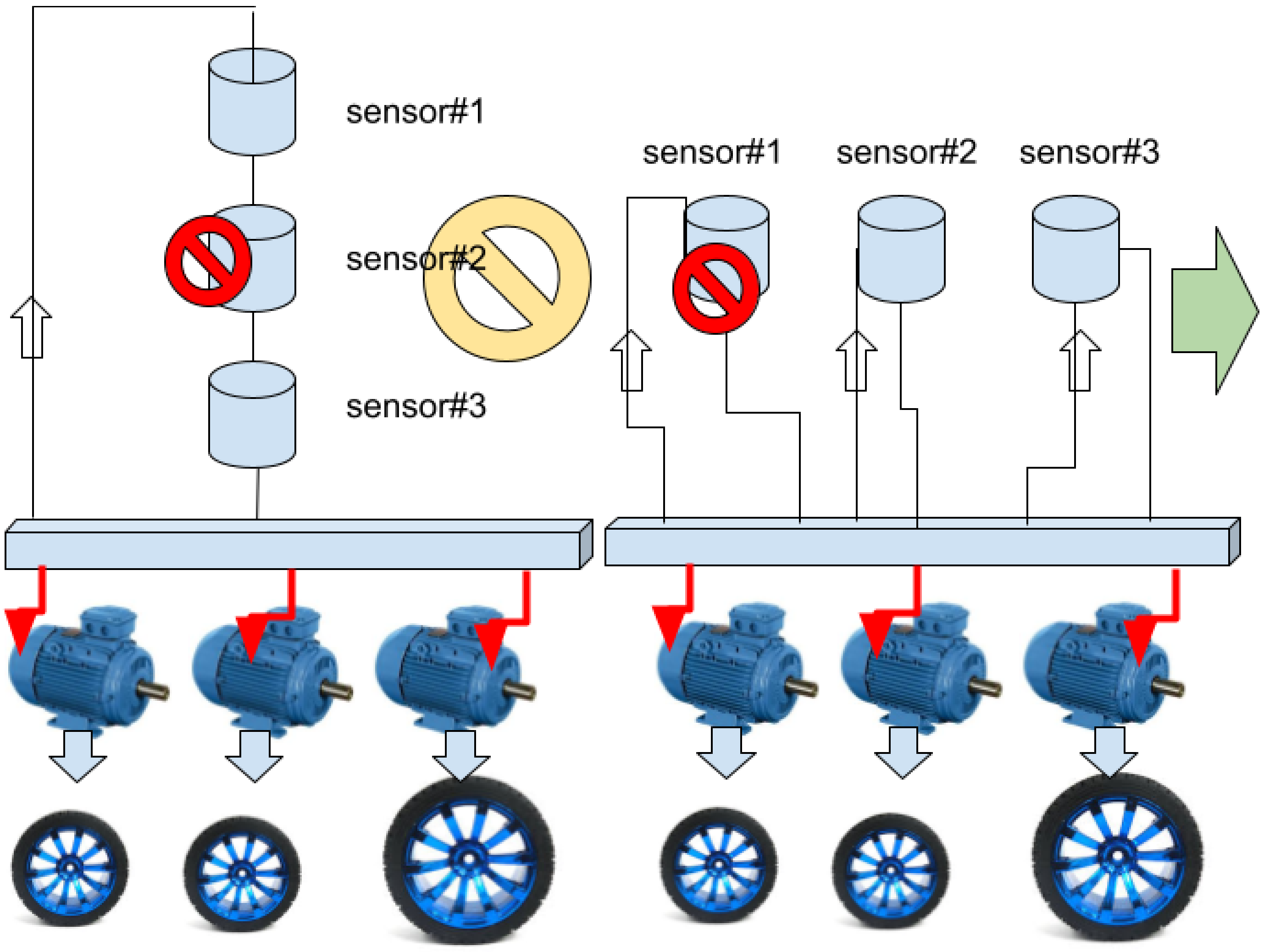
A combination of C# and MVPL can be used by developers. For instance developing in C# and using MVPL for tests.

CCR - Common Concurrent Runtime

CCR has been developed to handle multi-tasking, which is mandatory for development of robotics. Professional robotic code has to be event-based and asynchronous. Not polling the robots for info with timing loops that consume CPU power unnecessarily.

A robot cannot be developed without such runtime. For example if we consider a robot that depends on N sensors S(1)....S(N) to move forward. If the robot uses a sequential program, it will have to wait until *all* sensors return the information. If a sensor in the chain becomes faulty, the robot may be frozen and unable to move.

With the CCR, the robot will move forward - maybe less accurately - even if a sensor is faulty, and will report the faulty sensor to a diagnostic platform.



The above picture represents a ground 6x6 robot with 3 engines commanding each pair of wheels.

- On the left, the sensors are polled sequentially so in case of a faulty sensor (in the middle) the whole robot is frozen;
- On the left, with the Common Concurrent Runtime, a faulty sensor will not prevent the two other motors from moving the robot forward, while it may run with less efficiency.

DSS - Decentralized Software Services

The DSS component is a service which aims at solving intercommunication and event scheduling between sensors.

DSS allows each service component to update asynchronously the information of the `map` and `motion mode`, two very important information items.

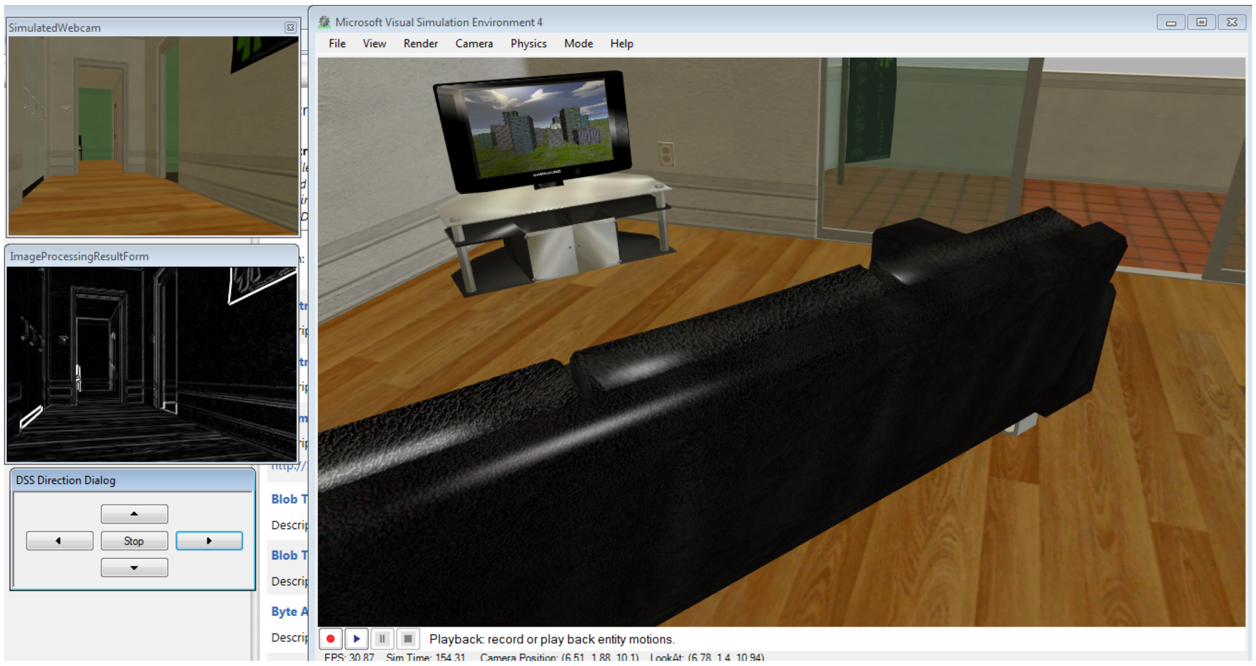
- An imaging service component may recognize objects
- An acoustic service may estimate distance between these objects

These two services will “feed” the information without interfering with each other.

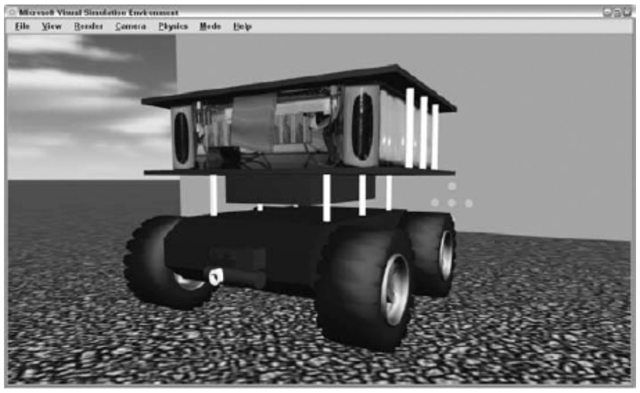
VSE - Visual Simulation Environment

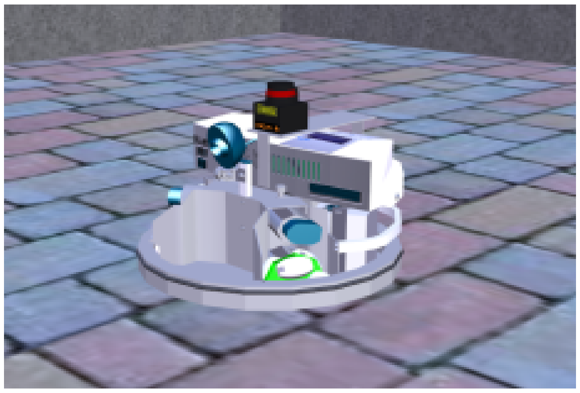
Before building and acquiring hardware for a robot - which hardware can be expensive - a simulation of how the robot will behave is needed, eventually correcting bugs and implementation problems.

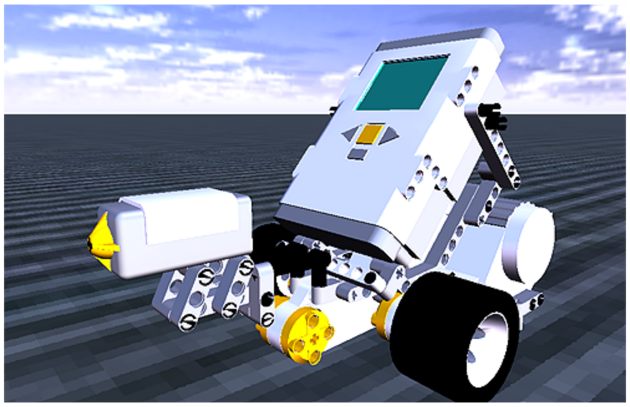
The Visual Simulation Environment is a 3D based virtual reality system, using Microsoft XNA 2.0 platform.

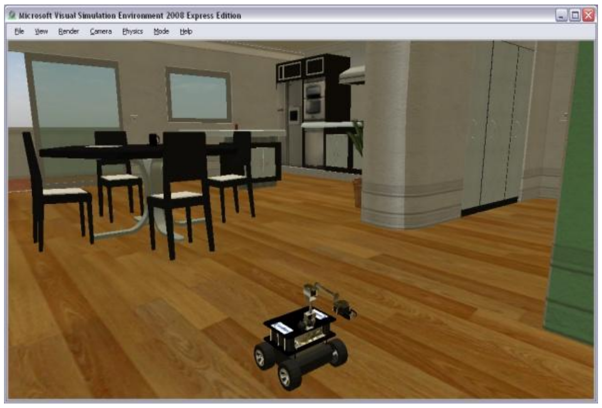

SimulatedWebcam
ImageProcessingResultForm
DSS Direction Dialog
Stop
Microsoft Visual Simulation Environment 4
File View Render Camera Physics Mode Help
Playback: record or play back entity motions.
Blob T
Descrip
Byte A

Its aims are recreating the conditions of use of a robot. For example an apartment with obstacles (furniture, TV…) in the case of a domestic servicing robot.


Microsoft Visual Simulation Environment
File View Render Camera Physics Mode Help





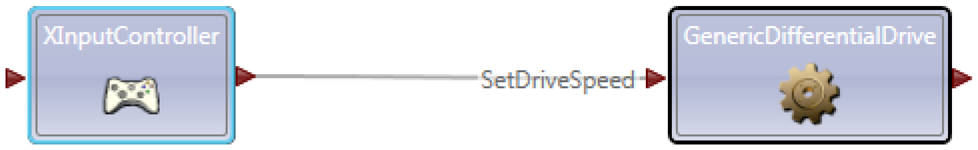

Microsoft Visual Simulation Environment 2008 Express Edition
File View Render Camera Physics Mode Help

All the sensors can be debugged and visualized in their own windows. The tool is very useful and very effective at simulating the behavior of a robot.

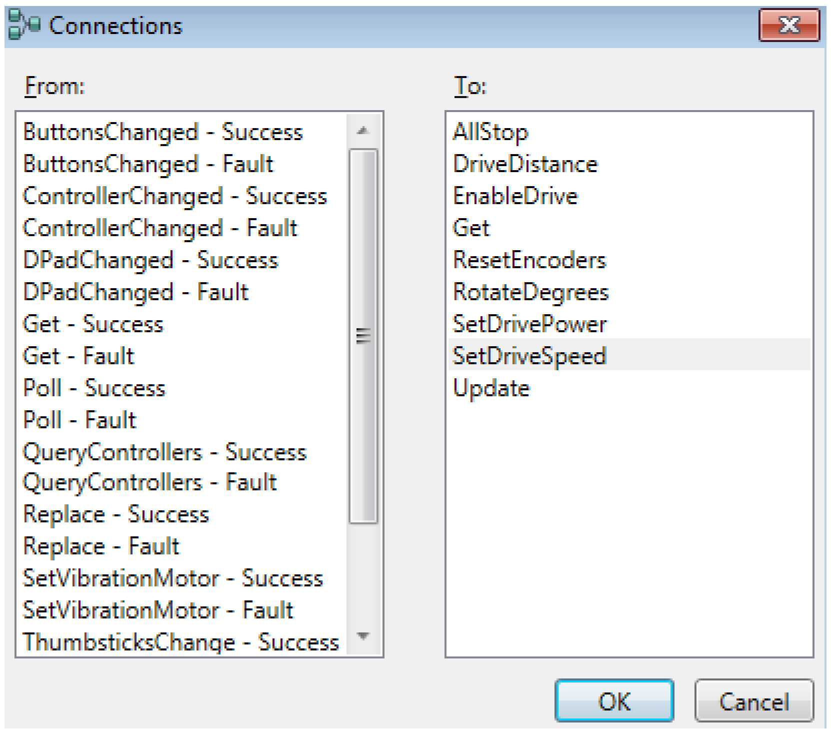
Overview of the MRDS Robotic Visual Language

MRDS allows to manipulate - with the visual language - “Activities” and “Services”. This means that services - for example infrared vision, laser probes, motors hardware components or image recognition services - will be connected all together by logical constraints or “activities”.

Here for instance we connect a XInputController (a “joystick”) to a differential drive (a left and a right motor)



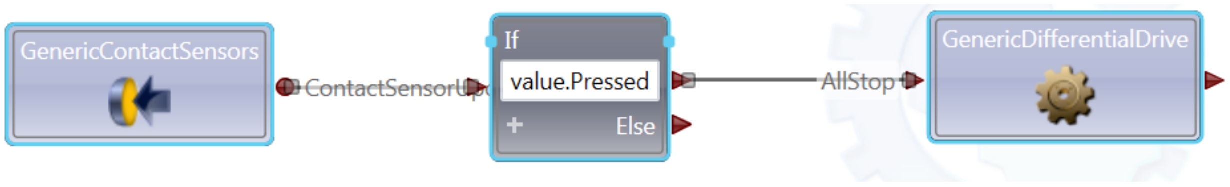
We can program visually how the joystick will interact with the differential drive:



From:	To:
ButtonsChanged - Success	AllStop
ButtonsChanged - Fault	DriveDistance
ControllerChanged - Success	EnableDrive
ControllerChanged - Fault	Get
DPadChanged - Success	ResetEncoders
DPadChanged - Fault	RotateDegrees
Get - Success	SetDrivePower
Get - Fault	SetDriveSpeed
Poll - Success	Update
Poll - Fault	
QueryControllers - Success	
QueryControllers - Fault	
Replace - Success	
Replace - Fault	
SetVibrationMotor - Success	
SetVibrationMotor - Fault	
ThumbsticksChange - Success	

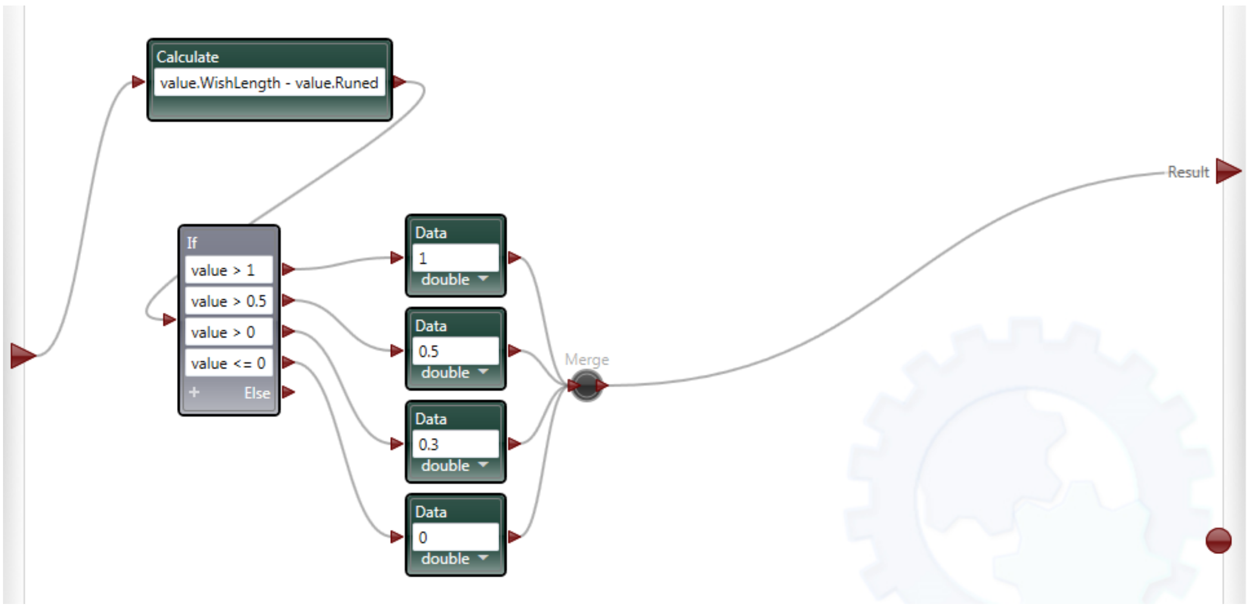
Here we show some more complicated examples, developing code for a LEGO NXT patrol robot.

If a contact sensor is engaged, the robot shall stop:



The robot stores several variables: `Length, Size, Stage,Wishlength,Runed, CurrentSpeed` etc...

The program makes the robot turn off a given angle at each start of a new stage and then makes it run smoothly, e.g. with a progressive initial acceleration and final deceleration. For this a simple visual procedure named `CalculateSpeed` is created.

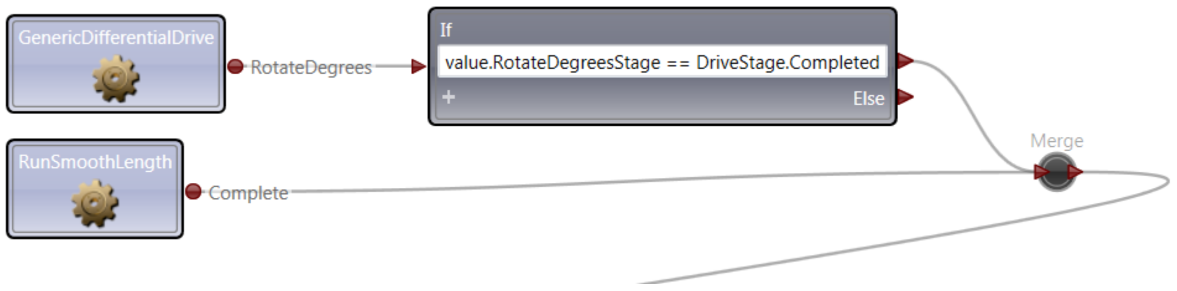


The function simply computes the difference `d=L-x` where `L` is the “wished” length and `x` the actual length and returns the following speeds :

- `1 if d>1`
- `0.5 if 0.5<d<1`
- `0.3 if 0<d<0.5`
- `0 if d<0`

Initially the robot will run with a speed unit of `1`, then decelerate to `0.5` then again to `0.3` and stop when `x=L`.

Another procedure named runL defines how the behave between each “stages” , eg between each turn of his patrol.



At the start, the robot will rotate by an initial value then will increment in such ways that the robots make some sorts of rounds.

```
-  Initial Stage#1: Robot rotates then run smoothly until
   desired length has been covered
-  Stage#2:  Robot rotates by an increment then run smoothly
   until desired length has been covered
-  Etc…
```

The `RunL` and `RunSmoothLength` blocks simply formalize that behavior.

When running by pressing F5, this will launch a simulation of a LEGO NXT patrol robot which will patrol around the territory. The code, basically, programs a robot to use a contact sensor to patrol “smoothly” around a given territory.

A Sample robotic project developed with MRDS

Here we present the main lines of the development of a typical small robotic project using LEGO robots package[1]. The whole development was done with MRDS and we provide this example to demonstrate the capacities of the IDE.

The constraints consist in the organization of a small robotic contest. Robots are placed inside a closed surface. The center of the surface is made of black rug.

Robots must use the LEGO components only and have the same control abilities, namely move forward and backward; turn clockwise, counterclockwise, left and right; and make reverse left turns and reverse right turns.

Sumo contest

Robots shall win against each other inside the closed area. Robots will push other robots to win, meaning when the opponent robot has been entirely thrown away from the center black rug or is pushed down and cannot continue its operations.

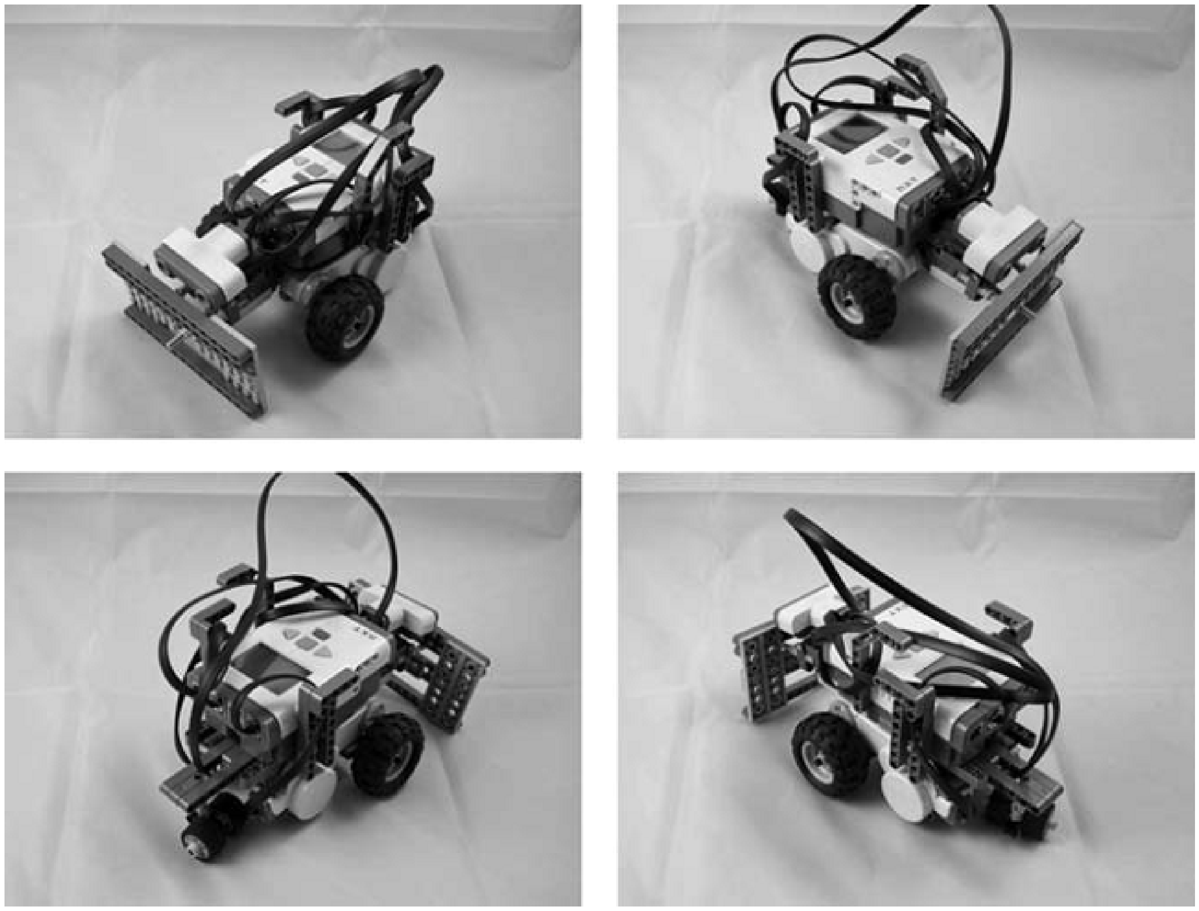
Design

There are 4 groups of competitors.

Group#1

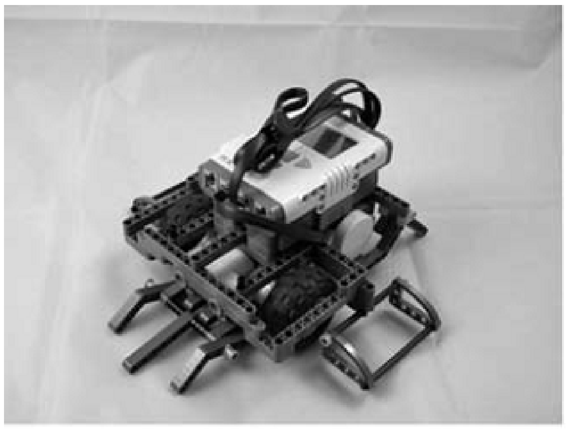
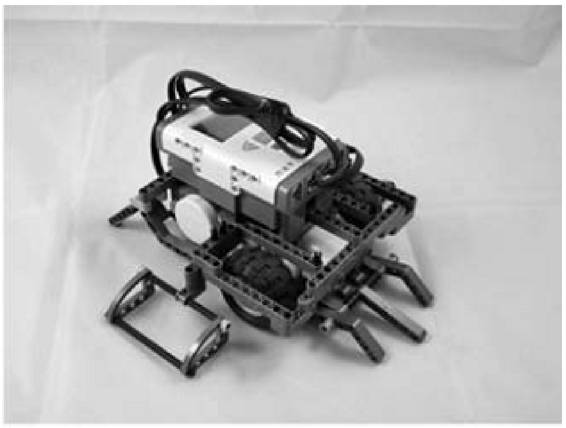
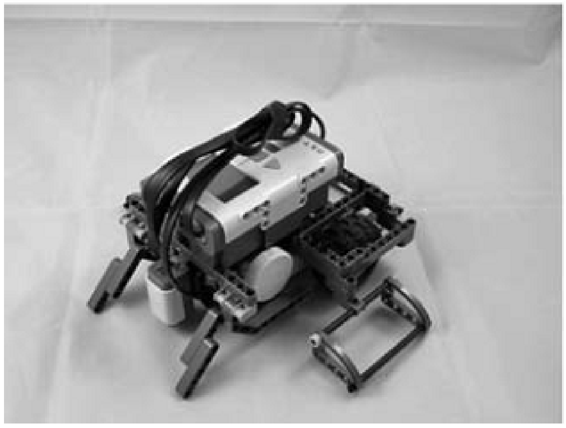
The first group uses an attack-oriented hardware design. Thrust and torque can be increased from a large gear installed on the wheel. A square repelling board is installed on the front.

[1] This example comes from the book “Robot Development Using Microsoft® Robotics Developer Studio” by Shih-Chung Kang , Wei-Tze Chang, Kai-Yuan Gu , Hung-Lin Chi



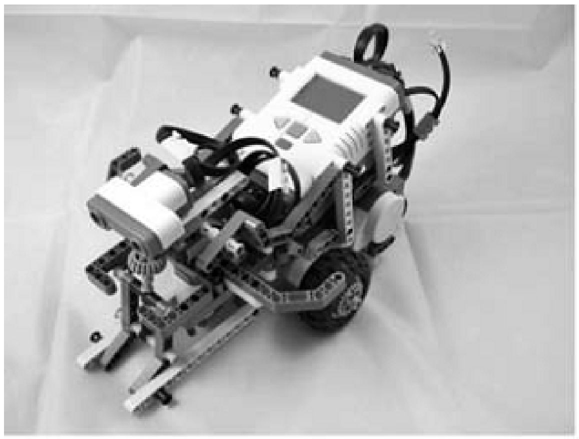
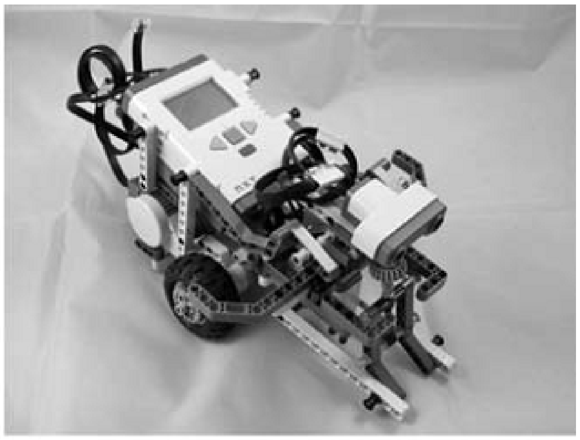
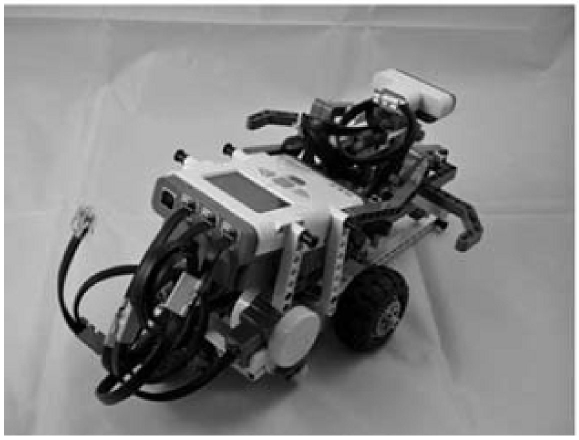
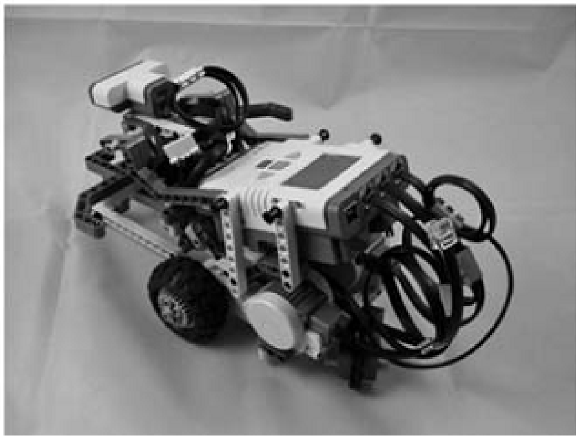
Group#2

The design of the group#2 is defense-inspired. The robots have a low center of gravity, they have anti-collision designed devices. Ultrasonic sensors to detect opponents.

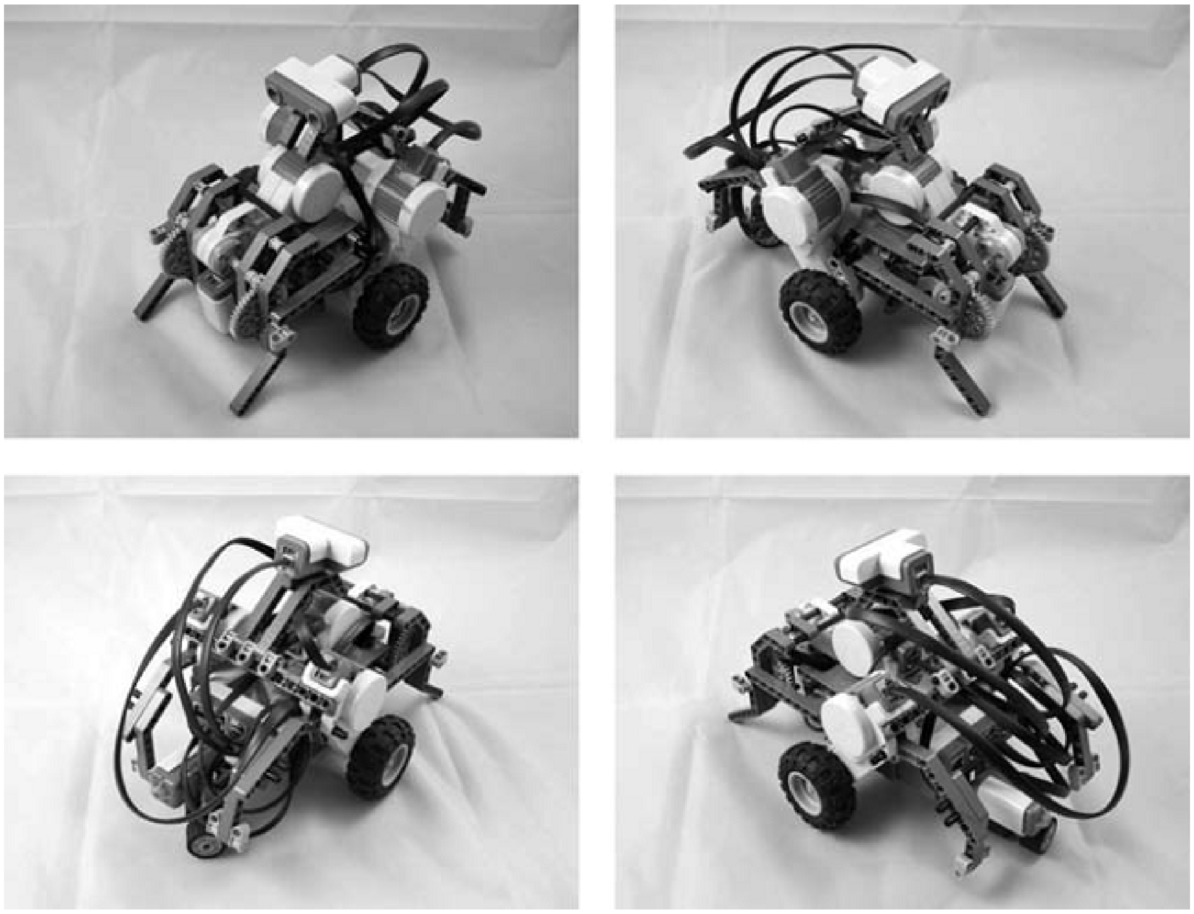
Group #3

These are attack robots. They use three motors and hence are heavier. Robots can "swing" forward and backward. Ultrasonic sensor.

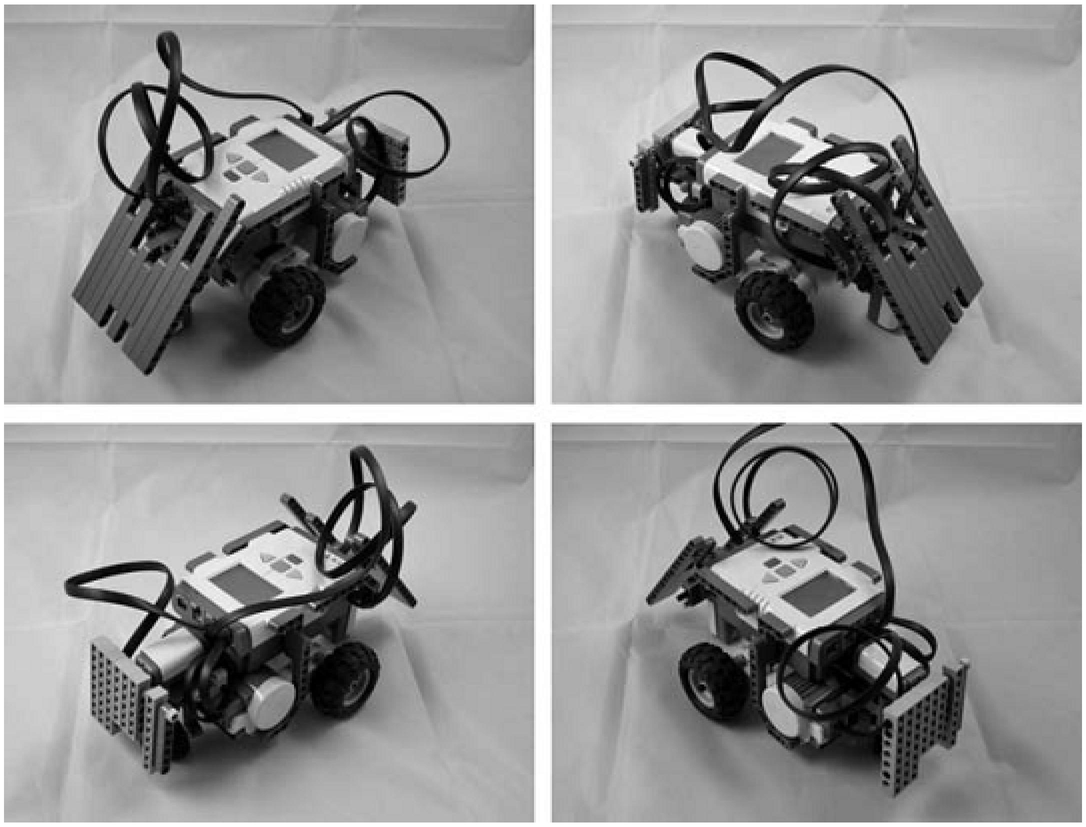
Group #4

Attack-designed robots. Also use three motors. Ultrasonic sensors are used and the robot always rotates on itself.



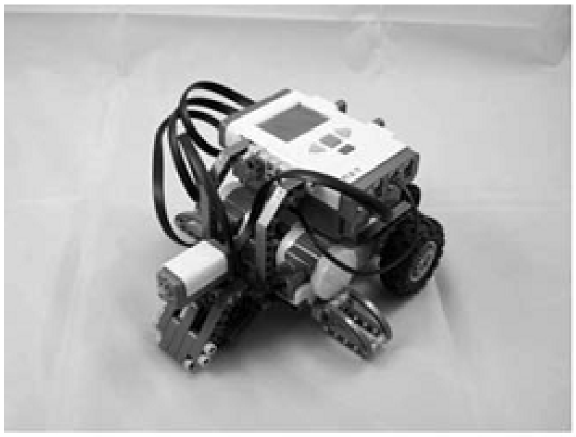
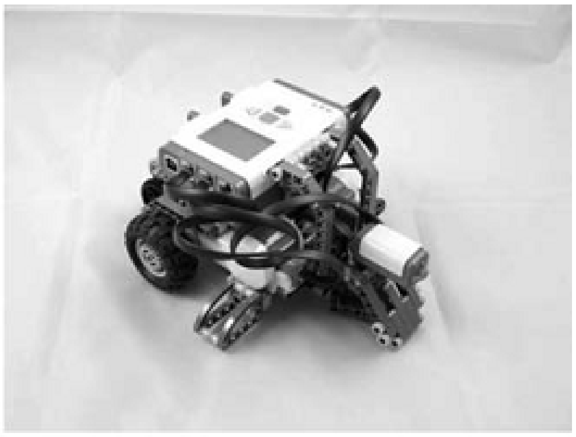
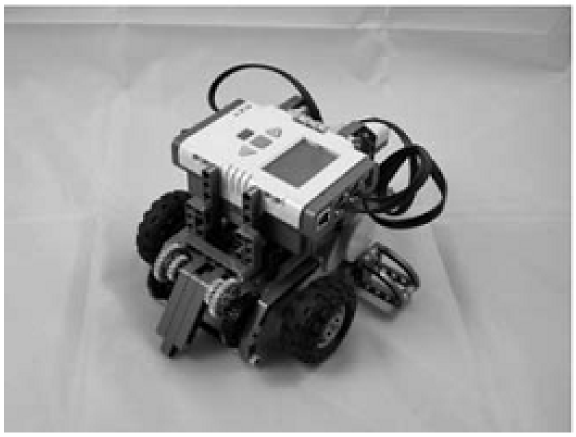
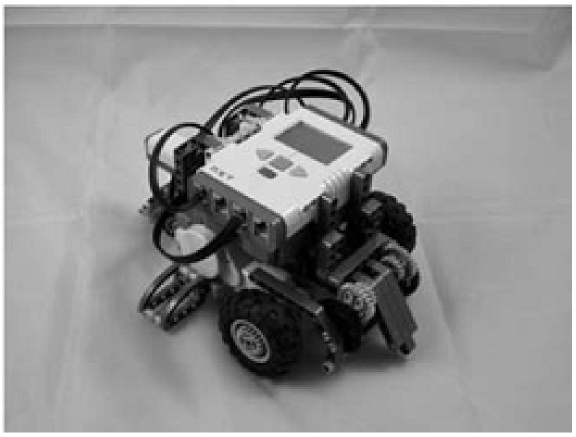
Group #5

The front of the robot has a blocking board, it can thrust to push the enemy when a fight is engaged and it moves in smart ways.



Group #6

These are reference robots for the contest. They can guard themselves from collision with the opponent.They have light sensors that prevent them from going out of bounds. They are doing random movements.

All these robots were developed with the Microsoft Robotics Developer Studio.

For further References

Lego Sumo Contest

Video available at :https://www.youtube.com/watch?v=0Km3fTx4AO4

Additional documentation and source codes

- MRDS Samples: https://archive.codeplex.com/?p=mrdssamples
- MRDS tutorials and lessons: https://sites.google.com/a/caece.net/robot-programming-2010/

Sample for the article MRS robot patrol:

Other Robotics Development Platforms

- Robot Task Commander
- Webots
- Robotc
- Simulink Robotics

ROS: Robot Operating System

Robot Operating System is a set of tools and libraries and was made recently available for windows 10.

The future of Microsoft Robotics

While MRDS has been officially stopped, the robot operating system has been ported to Windows and Microsoft is preparing a new platform for robotics using existing A.I platforms and simulators. MRDS should therefore restart its activities through other compatible software to be developed in the near future by Microsoft.