Microsoft Robotics Studio (VO)

- Robopedia - Components -

Date de mise en ligne : lundi 9 octobre 2006

Description :
Gives a brief overview of the uses of Microsoft Robotics Studio and explores the details of supporting new hardware devices

---

PoBot
Microsoft has finally entered the Robotics scene. What have they made? How does it work and can it help you?

Almost each month since June 2006, Microsoft has released a new version of their Robotics Studio (MRS). Now in its fourth release (October when writing) it is becoming a more stable code base that looks like it has a strong future.

An MRS application is a group of services running on one or more nodes that exchange messages between each other to create the desired behaviour. The services exchange information through a flavour of webservice called Web Services Application Protocol (WSAP).

A machine can have many nodes, and services in each node can communicate with services within the node, with services in nodes on the same machine or with services in nodes on other machines.

Each service has its own state, and care is taken to separate actions which change that state and those that don't.

Enabling all this message exchange is an MS technology that has not yet made it into the core of any .net language, but is being tested in MRS as a possible candidate for inclusion in the Common Language Runtime. It is the CCR, or Coordination and Concurrency Runtime. It is very powerful, and to those not familiar with service orientated architectures, it can be very confusing. The CCR and related language extensions rely on some .net2 features including generic types and iterators. The CCR extends this to allow you to iterate through code blocks, using the 'yield return' construct.

Fundamental to the CCR is the notion of Ports. Ports are queues which expect a certain type to be sent to them, and hold a list of handlers which will be called when a that type is received.

\[// Create a port : Port _timerPort = new Port() ;\]

\[// Define the handler : void TimerHandler(DateTime signal) // do something\]

\[// Activate the handler : Activate(Arbiter.Receive(true, _timePort, TimerHandler)) ;\]

Now anytime you want that handler to fire, you post a message to the port with a DateTime type.

\[_timerPort.Post(DateTime.Now) ;\]

The example above is not useful on its own, but this same technique can be used with ports that support many types and many handlers. You can create joined receivers which wait for two or more posts, easily implement a
scatter/gather patterns, create choices on the results of many ports etc. Further, you can have many services
subscribed to a port which will all get the message. This can be useful, for instance, for using sonar data to make a
map, but also using it as a bumper. However heavy and unyielding your map making is, you know that your bumper
service will get the message and deal with it.

If you have ever tried making a procedural or threaded application which tries to walk forwards, while checking
encoders, scanning sonar, checking bumpers, making maps etc. you will know that it can be difficult to coordinate all
these activities effectively.

On to some examples...

The SRF235 sonar module from Devantech speaks I2C and can return a Range reading in centimeters. To interface
your PC to the I2C bus, you can use Devantech's very cheap USBI2C module, or the wireless RF03/CM02
combination.

So, a micro-controller based application would have a loop to read the two bytes of range data every few hundred
milliseconds.

Using MRS you would make a service which polls the device to update its own state to store the Range data. The
state can then be read by another service using a GET command, or if the service is configured to use subscriptions,
all subscribed services would be notified upon change of the state.

State can be serialised to a browser as XML:
Raw XML of the SRF235

Using XSLT, the XML can be transformed into an HTML form which allows configuration to be set through a web browser.

**SRF235 Hardware Service**

The SRF235 is a narrow beam ultrasonic ranger, with a beam width of just 15°. An industrial high frequency air coupled transducer operating at 235KHz is used for both Transmit and Receive.

You can then make instances of generic contracts such as Sonar and SonarAsBumper that subscribe to the specific sonar and can be used by other hardware agnostic services.

To any microcontroller head all the above should seem very wasteful. Why XML ? Why an operating system ? No real time ? WTF ?

But !, PCs are becoming fast and cheap and the CCR can handle around 90000 soap message per second (on a multi-processor machine).

Using generic contracts allows for logic to be re-used from one robot or application to the next, no matter if is is Pioneer with a lazer range finder, or a LEGO NXT.

Inter-node communication can let you use your PC as the brain for webcam framegrabbing and object recognition while letting a dumber onboard device do PID control. You could even download a weather forecast and tell your bot to head indoors if rain is coming.
MS has added support for generic two wheel drives, webcams, text-to-speech, sonar, contact sensors etc. These generic contracts allow one of MRS's most powerful features, which is its integration with the AGEA physics engine, so you can send your virtual robot off into a maze with real physics and test the position of your bumpers and the integrity of your code while watching its simulated webcam.

For a quick overview of a service based program, see the Say Fire example [http://www.pobot.org/MRS-Example-Show-me-a-picture-when.html]

That is all for now. There is much more documentation on Microsoft's site, including some good videos on the use of the CCR beyond robotics on the channel 9 wiki.


MRS OCTOBER : http://msdn.microsoft.com/robotics/...

Devantech Drivers : http://www.robot-electronics.co.uk/...
[http://www.robot-electronics.co.uk/shop/Examples.htm]

Enjoy!

Chris Kilner