PROJECT ROSCOE SYSTEM DESIGN DOCUMENT

Jonathan Groff

8/11/2004

DESCRIPTION

ROSCOE is an acronym for Robot Operating System Open Connected Environment and is a 'work-rated' prototype robot using ROS (Robot Operating System) as the sensor and control bus. Using ROS, we can provide a platform to develop new and better instrumentation and algorithms with the confidence that new generations of hardware will easily integrate. In contrast to most robotics which represent a monolithic, one-off, compilation of sensors glued together with the hopes and dreams of a better tomorrow, ROS provides a decoupled network architecture with high cohesion and fault tolerance. If we are to rely on machines such as these, an architecture such as this seems advantageous. An easy migration to swarm robotic is also the promise of an evolved ROS Core distribution.

The barrier to entry with ROS has been high. It is not a dilettante system and in its entirety comprises the whole of an operating system. A major objective is to provide a more accessible version written in Java, which moves the platform more toward a cohesive set of modules the casual developer can work with. The target platform is the ARM SBC series and Java was the choice due to the DBX Jazelle Java acceleration in the ARM-J cores.

HARDWARE

Specialized hardware was avoided to keep construction simple and costs low. Electric bike hubs were chosen for their power and relative quiet. A device that would be as home in the home or the field was an objective of the design. Dynamic stability issues were avoided through the use of a rear castor with 360 degree rotation which bears the weight of the battery and motor controller housed in a standard automotive battery box.

PHYSICAL DIMENSIONS AND SPECIFICATIONS

Length Overall: 24”

Width and Wheel Track: 16”

Height: 32” max.

Weight: Est 60-70 lbs.

Wheel diameter: 16”

Number of Wheels: 3, 2 driven, 1 360 degree passive castor

Top Speed: Est 35-40 MPH

Drive Type: Differential, IMU integrated

Power Bus: 48V

Base OS: Linux/ROS

Microcontrollers: 3 Raspberry Pi, 1 Mega2560

Network: Switch managed IP

Sensors: IMU, 2 cameras (front/rear), 2 forward ultrasonic sensors

Est. Cost: ~$1200

Image 1: ROSCOE

The main housing is a 20mm ammunition case chosen for its durability and low cost. Two heavy L brackets connect the battery case and rear castor to the ammo case. A polycarbonate sheet is under the battery box and another shields the battery from impact. The primary wireless access point, IMU and cameras are comprised of a Parrot ARDrone board set housed in a 35mm underwater camera case on top of the unit. A standard camera tripod spade mount affixes it. The ultrasonic sensors and camera function through 2 holes drilled in the front of the case. An additional ultrasonic sensor is mounted at floor level and is interfaced to the Arduino Mega2560 attached via USB to one of the Raspberry Pi's in the local network the robot maintains. Data from the ultrasonic and motor controller is asynchronously streamed over the USB port to the RPi and multiplexed onto the ROS publish/subscribe bus.
Currently 2 Raspberry Pi's are used , with an additional slot for another . The ARDrone wireless access point routes traffic via a wireless USB dongle on one RPi, then through a 48V switch to the other nodes in the network. Affixed to the top of the unit are the ARDrone target recognition stickers to identify the unit to aerial drones.

POWER BUS

The system is 48V via LiPo integrated BMU cell. Two buck converters supply 12 volts for the ARDrone and 5V for the other logic. Raw 48V is supplied to the motor controller and network switch.

MOTOR CONTROLLER

The single biggest cost is the motor controller which was chosen for reliability and ease of use. A Roboteq VBL2360 was chosen for dual channel rating in the range desired. The unit communicates via RS232 to the Mega2560 attached via USB to the RPi. It provides battery voltage status and motor status which is multiplexed on the ROS bus.

COGNITIVE FUNCTIONS

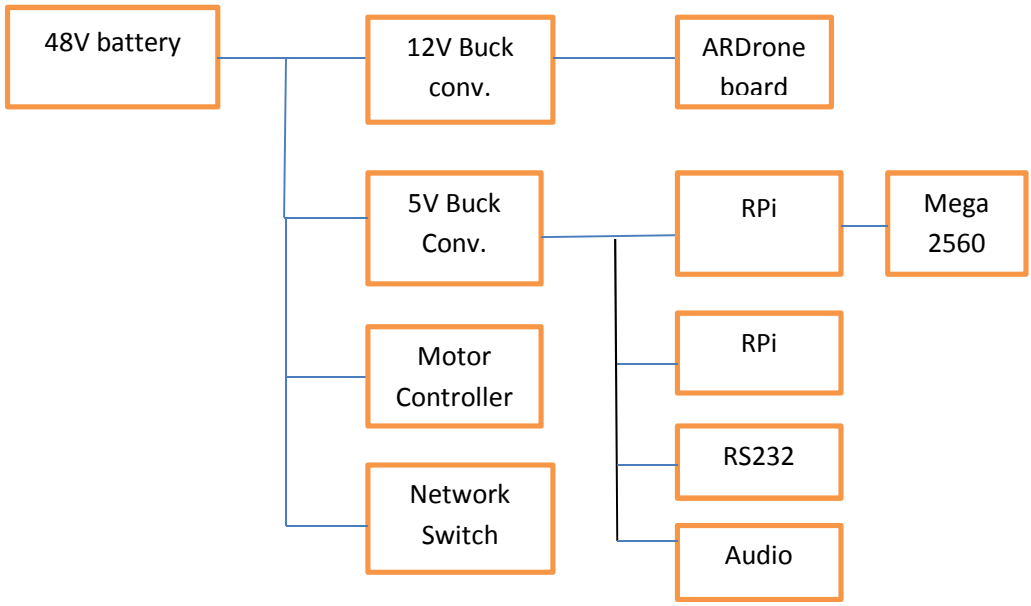
Upon completion of the basic hardware platform and implementation of the ROS bus, work will commence on utilization of the new science of Category Theory and its application to sensor fusion in the ROSCOE environment. Using the open source BigSack and Relatrix, a functor-based object deep store will be used to form commutative joins on stored sensor data and make decisions based on these commutative structures. The concept at this point is to have data from camera frames, ultrasonic, and other available relevant sensor data stored at those times when decision-making occurs and performing the natural transformation of functors to relate the new concepts to the previous ones. The work described is illustrated by the following publications:

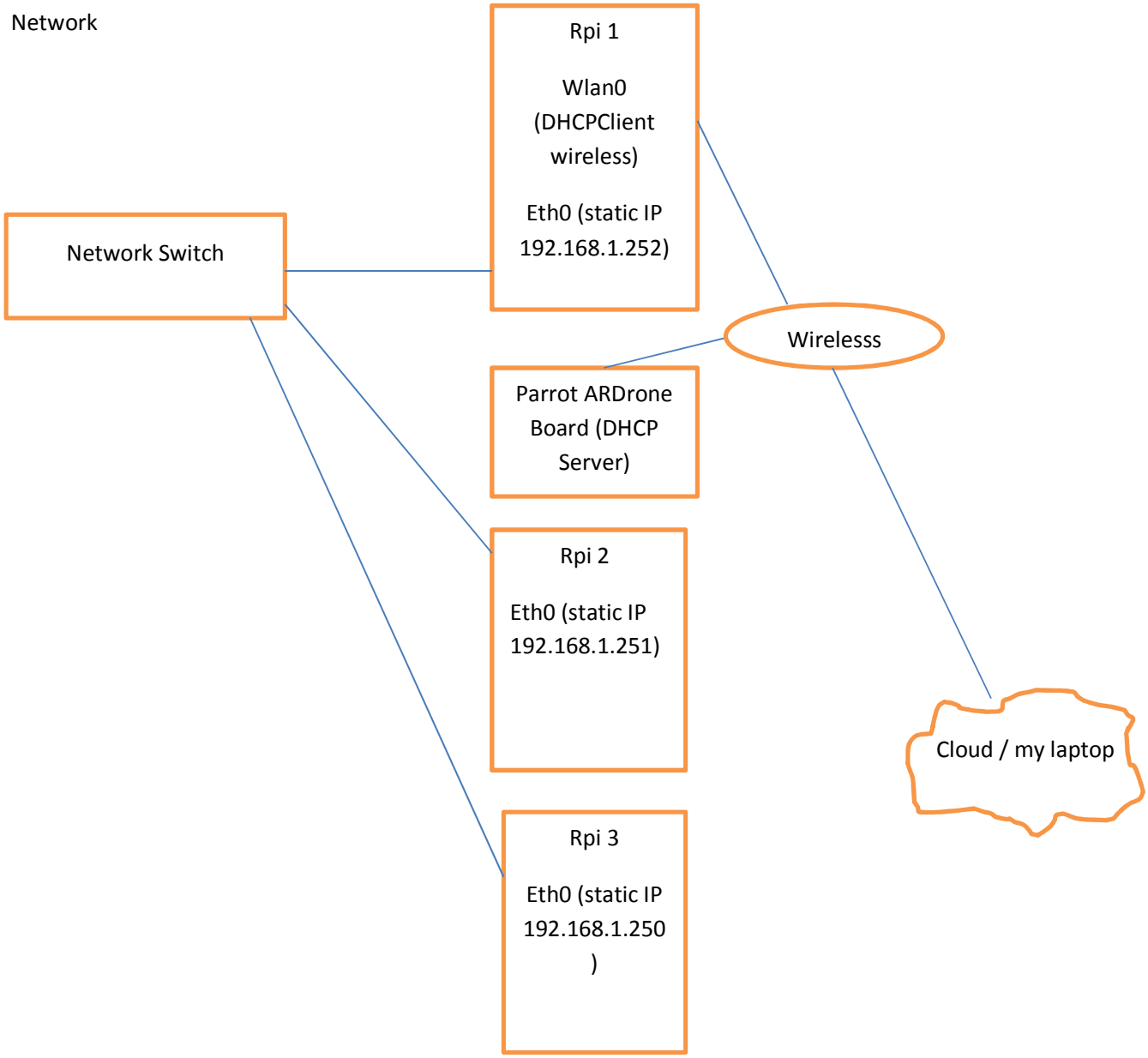
Applying category theory to improve the performance of a neural architecture
http://www.sciencedirect.com/science/article/pii/S0925231209001052
A Model of Human Categorization and Similarity Based Upon Category Theory
http://repository.unm.edu/handle/1928/6724?show=full
Category Theory Applied to Neural Modeling and Graphical Representations (2000)
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.32.2635

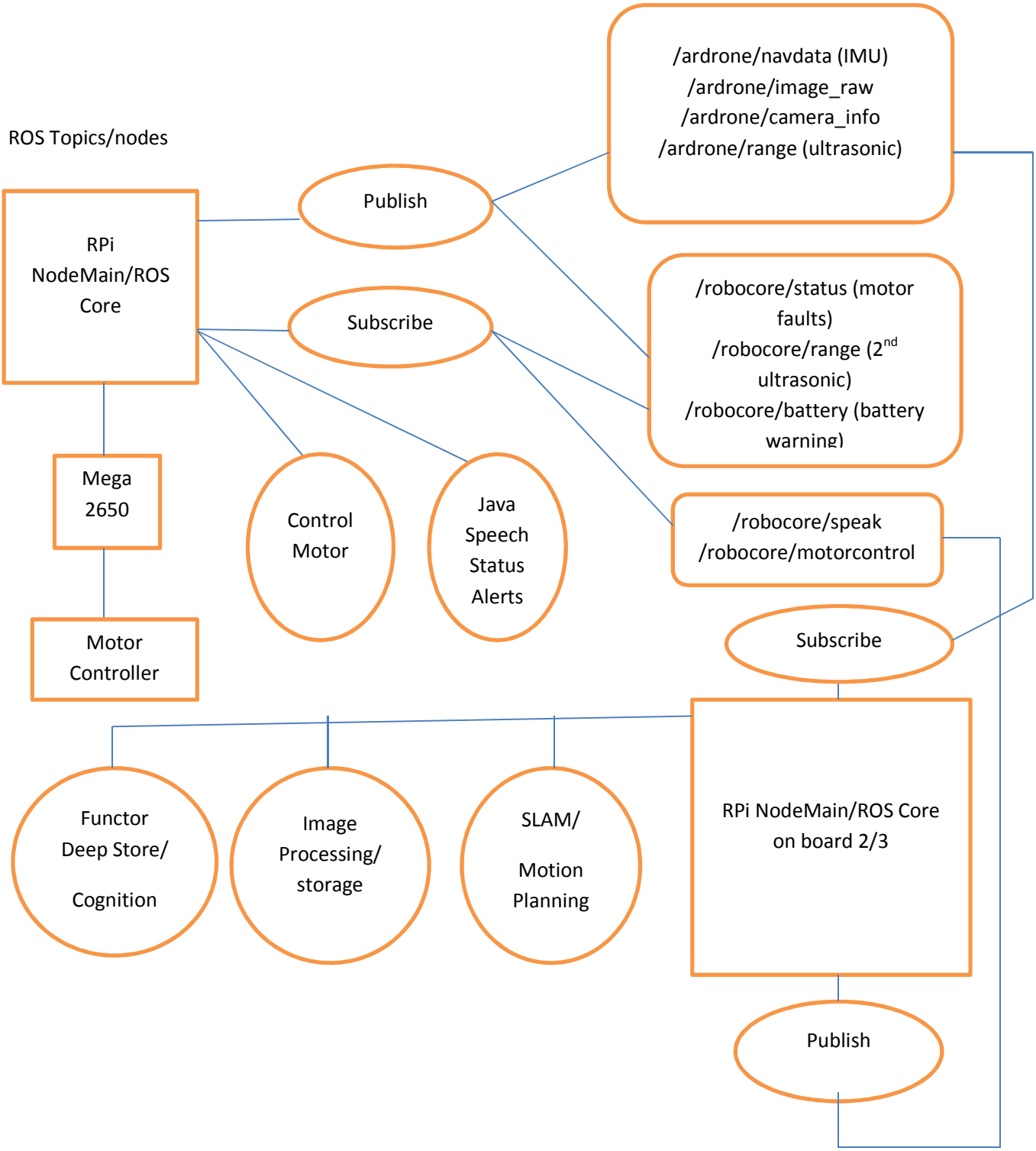
TASKING

An overall method of tasking the robot was based on 3D printing applications and solid modeling. By drawing the area to be tasked and generating the proper G codes referenced to the relative or absolute real-world coordinate system, the machine could be set to perform any given task in the area defined by the generated solid model. Interspersed with the overall tasking, the cognitive functions and SLAM processes of the robot would intervene in those situations where the task is interrupted by the introduction of obstacles or events. More than anything else, the above functionality as a whole is the primary final objective of this robot. It has yet to be seen whether this new science of Category theory can bring us closer to usable machine intelligence.

Power Bus


48V battery
12V Buck conv.
ARDrone board
5V Buck Conv.
RPi
Mega 2560
Motor Controller
RPi
RS232
Network Switch
Audio


Network
Rpi 1
Wlan0 (DHCPClient wireless)
Eth0 (static IP 192.168.1.252)
Network Switch
Wirelesss
Parrot ARDrone Board (DHCP Server)
Rpi 2
Eth0 (static IP 192.168.1.251)
Cloud / my laptop
Rpi 3
Eth0 (static IP 192.168.1.250)


ROS Topics/nodes
/ardrone/navdata (IMU)
/ardrone/image_raw
/ardrone/camera_info
/ardrone/range (ultrasonic)
Publish
RPi NodeMain/ROS Core
Subscribe
/robocore/status (motor faults)
/robocore/range (2nd ultrasonic)
/robocore/battery (battery warning)
Mega 2650
Control Motor
Java Speech Status Alerts
/robocore/speak
/robocore/motorcontrol
Motor Controller
Subscribe
Functor Deep Store/ Cognition
Image Processing/ storage
SLAM/ Motion Planning
RPi NodeMain/ROS Core on board 2/3
Publish