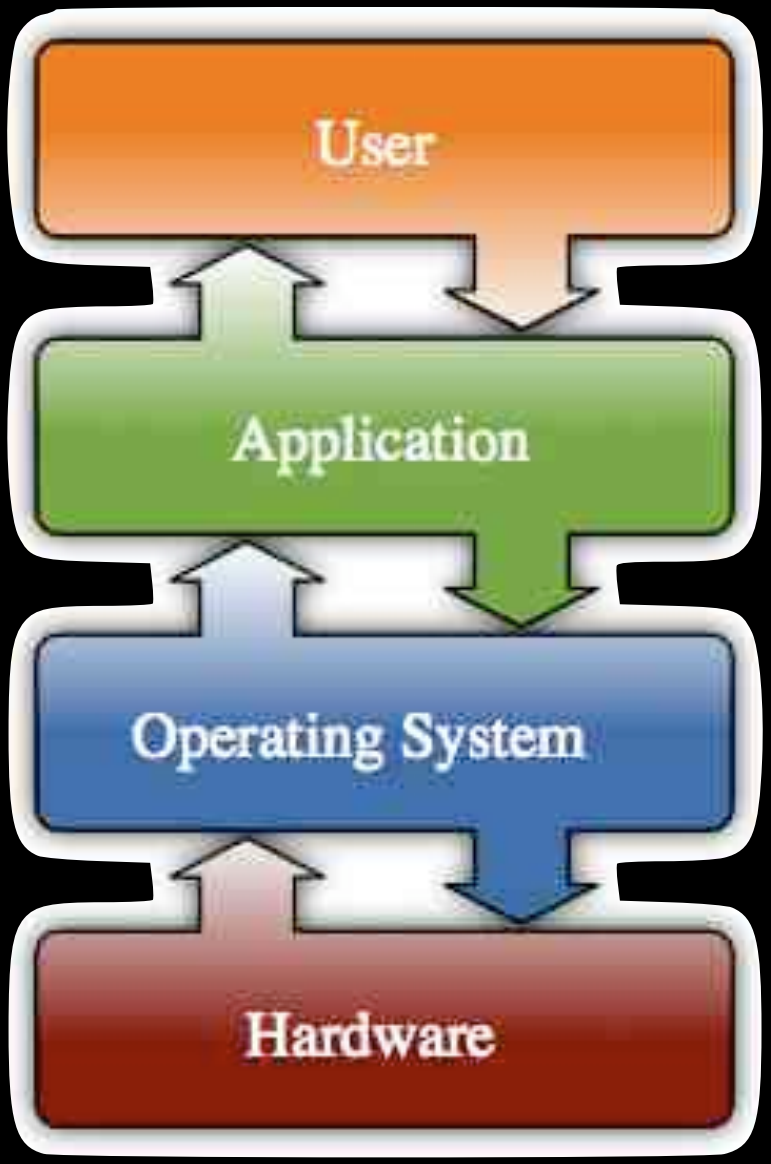
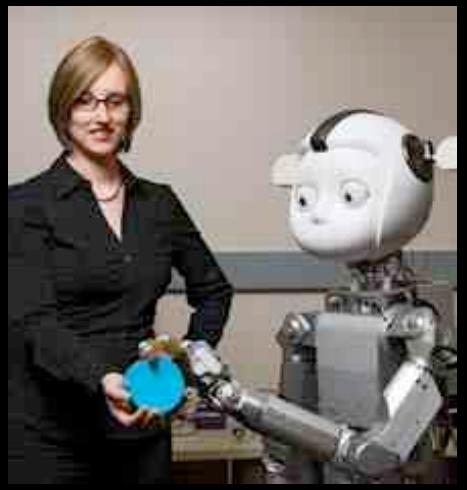
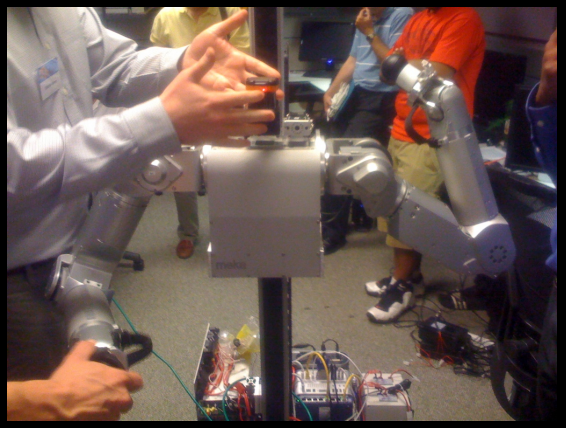



# Topic 2:

# Robot Middleware

app-stracting robot hardware

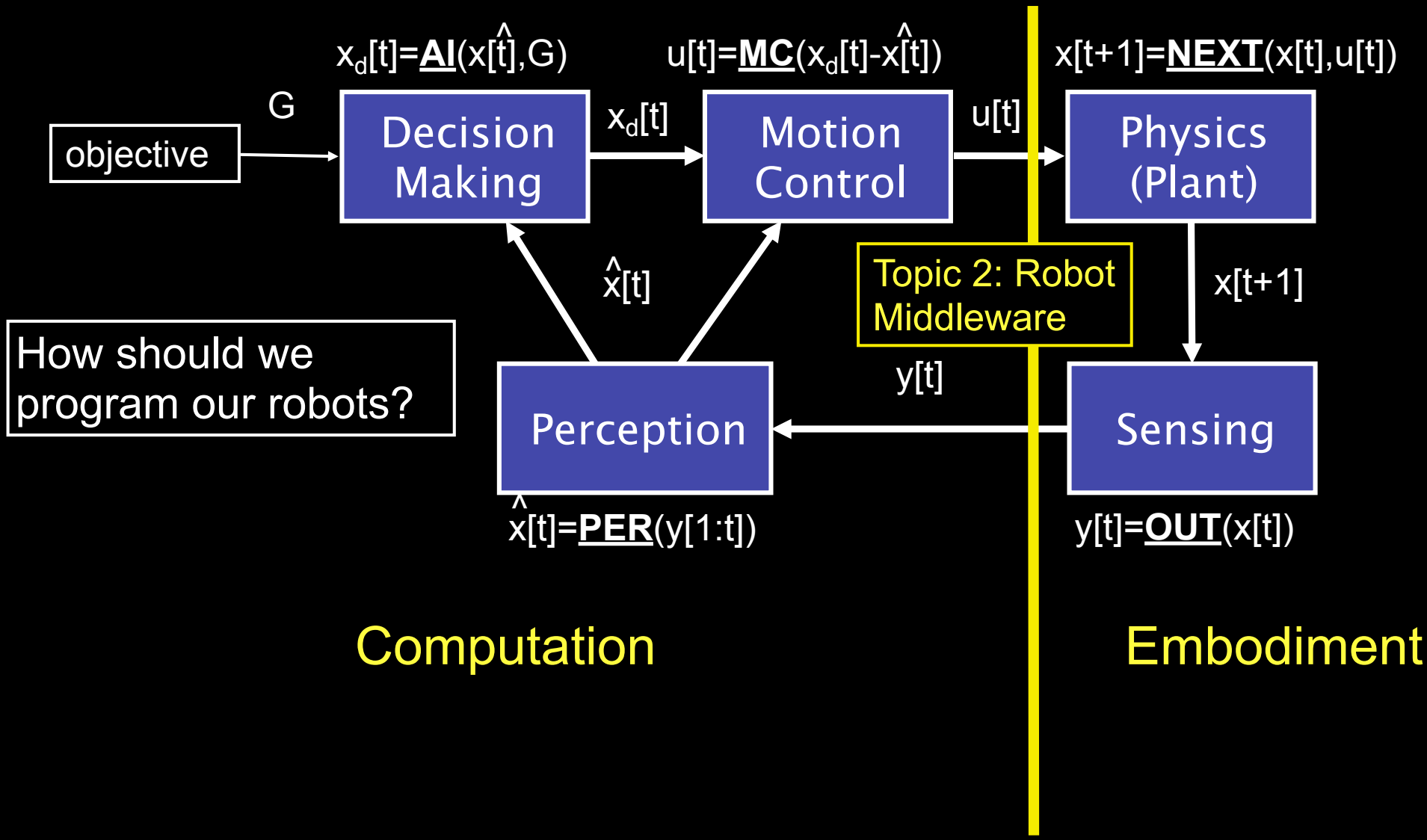



NewScientist
Robots to get their own operating system

# robot control loop

- someone please sketch on the board

# The Robot Control Loop

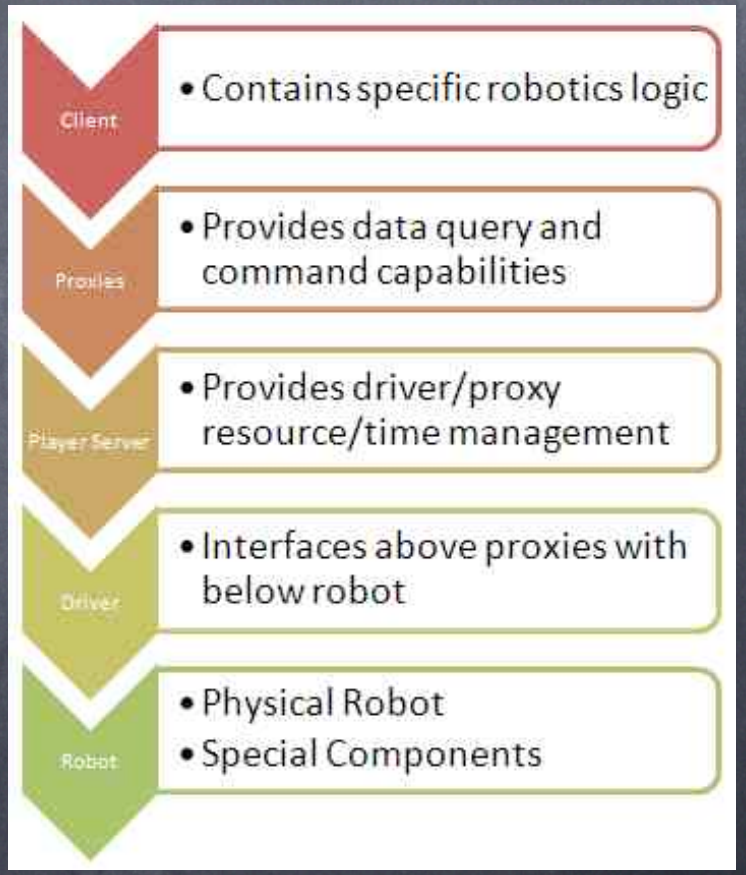


# Straightforward approach

- Just write and compile a program to perform robot's "cognitive" functions
- Scalability to growth in functionality?
- Portability to other robots?
- Reusability for other projects?
- Interoperability btwn functions/languages?
- How useful is a "one-of" solution?

# Robot Middleware

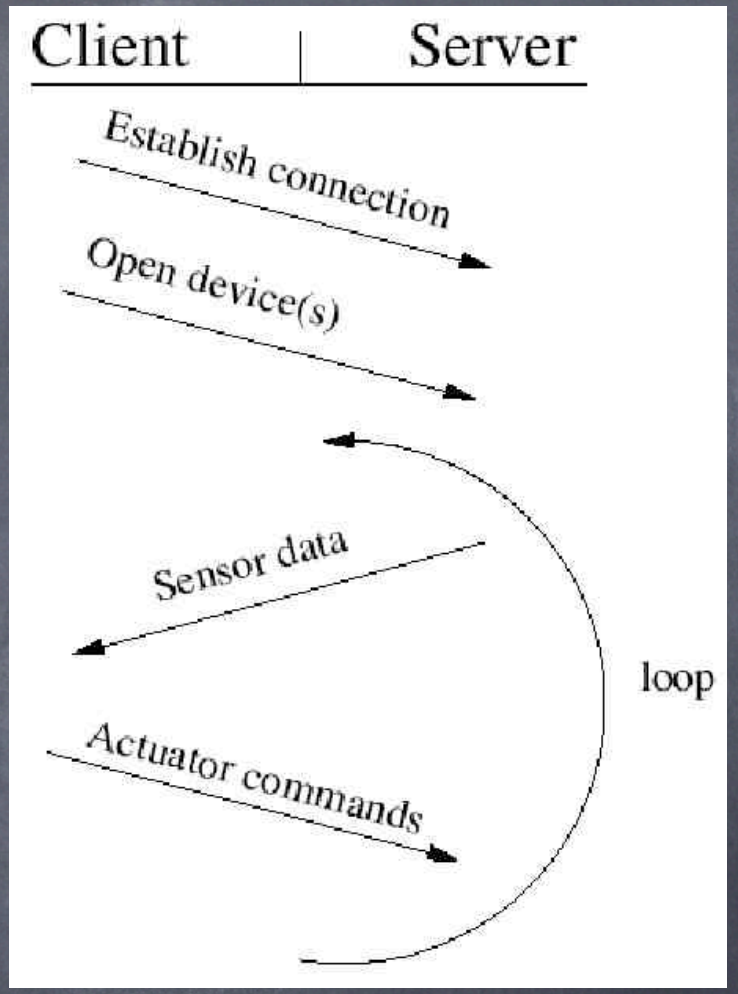
- Provide an abstraction layer and drivers between computation and embodiment
  - Analogy: hardware abstraction layer in an operating system
- Several existing packages
  - Player: client/server model
  - ROS: peer-to-peer model



# Player

(playerstage.sourceforge.net)

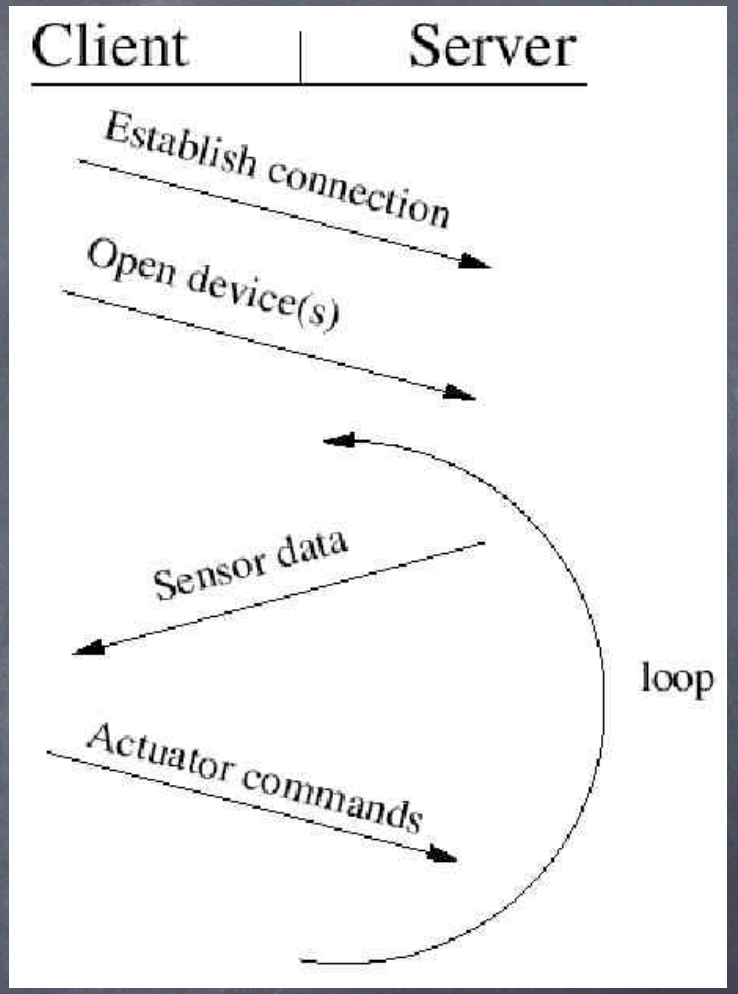
- Client-server architecture
- Publish-subscribe messaging
- Server runs on robot
  - exposes devices as proxies
  - publishes data continuously
- Clients (applications) access server
  - subscribe to robot proxies
  - read(), compute, set vars, read()



# Player

(playerstage.sourceforge.net)

- Messaging via network (TCP/IP)
  - no dependence on prog. lang
  - client libraries (eg, libplayerc)
- Proxies expose interfaces to both
  - hardware devices
  - existing implemented algorithms
- Player is the default pkg for cs148



# Player proxies

- what proxies would we have for this mobile robot?
  - sketch on board
- how could a client make it move and react to obstacles?

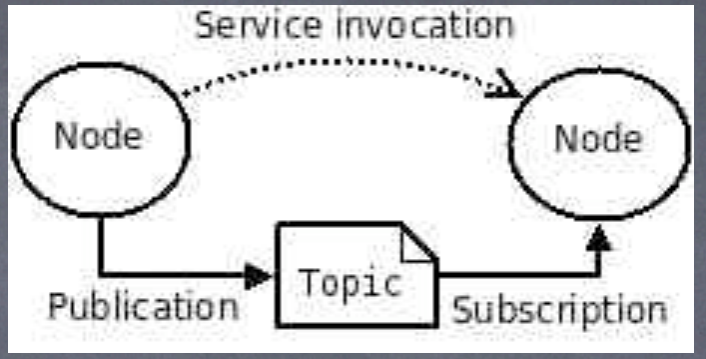
# Player proxies

- Proxies of interest
  - position2d: planar movement and odometry
  - bumper
  - ir: virtual walls, floor detect
  - camerauvc: usb camera
  - camera1394: firewire camera
  - blobfinder: color recognition

- "player" server with cfg file
- "make" to build client; link to libplayerc
- run client anywhere on network

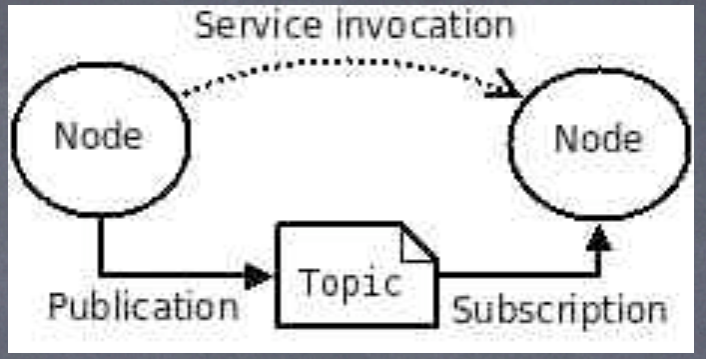
```c
#include <stdio.h>
#include "playerc.h"

int main(int argc, const char **argv)
{
  int i;
  playerc_client_t *client;
  playerc_position_t *position;

  // Create a client object and connect to the server; the server must
  // be running on "localhost" at port 6665
  client = playerc_client_create(NULL, "localhost", 6665);
  if (playerc_client_connect(client) != 0)
  {
    fprintf(stderr, "error: %s\n", playerc_error_str());
    return -1;
  }

  // Create a position proxy (device id "position:0") and susbscribe
  // in read/write mode
  position = playerc_position_create(client, 0);
  if (playerc_position_subscribe(position, PLAYER_ALL_MODE) != 0)
  {
    fprintf(stderr, "error: %s\n", playerc_error_str());
    return -1;
  }

  // Enable the robots motors
  playerc_position_enable(position, 1);

  // Start the robot turning slowing
  playerc_position_set_cmd_vel(position, 0, 0, 0.1, 1);

  for (i = 0; i < 200; i++)
  {
    // Read data from the server and display current robot position
    playerc_client_read(client);
    printf("position : %f %f %f\n",
           position->px, position->py, position->pa);
  }

  // Shutdown and tidy up
  playerc_position_unsubscribe(position);
  playerc_position_destroy(position);
  playerc_client_disconnect(client);
  playerc_client_destroy(client);

  return 0;
}
```

# ROS (ros.org)



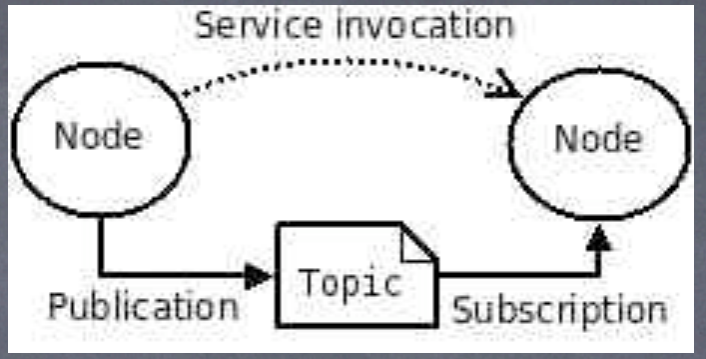

- "Robot Operating System"
- Peer-to-peer architecture over network
- Every robot function (processing) is a node
  - Node essentially its own executing process
- Nodes subscribe to and publish on topics
  - Master node runs topic name service
- Topics: basic message structure for broadcasting data
- Services for node-to-node communication

# ROS (ros.org)




- Client libraries (eg, roscpp, rospy, rosjava)
- Package management: integrated build system
  - rosmake to build, rosrun to execute nodes
- Integration of external packages and repositories
  - OpenCV, OpenRAVE, Player, etc.
- brown-ros-pkg contains drivers for Create
  - WARNING! path not yet traveled

# ROS (ros.org)




- 200-level credit for using ROS for your projects
- WARNING! Path not yet traveled
  - ROS stability issues, adaptation will be necessary
  - we will be learning with you during projects
  - extensive use of ROS mailing list and online documentation