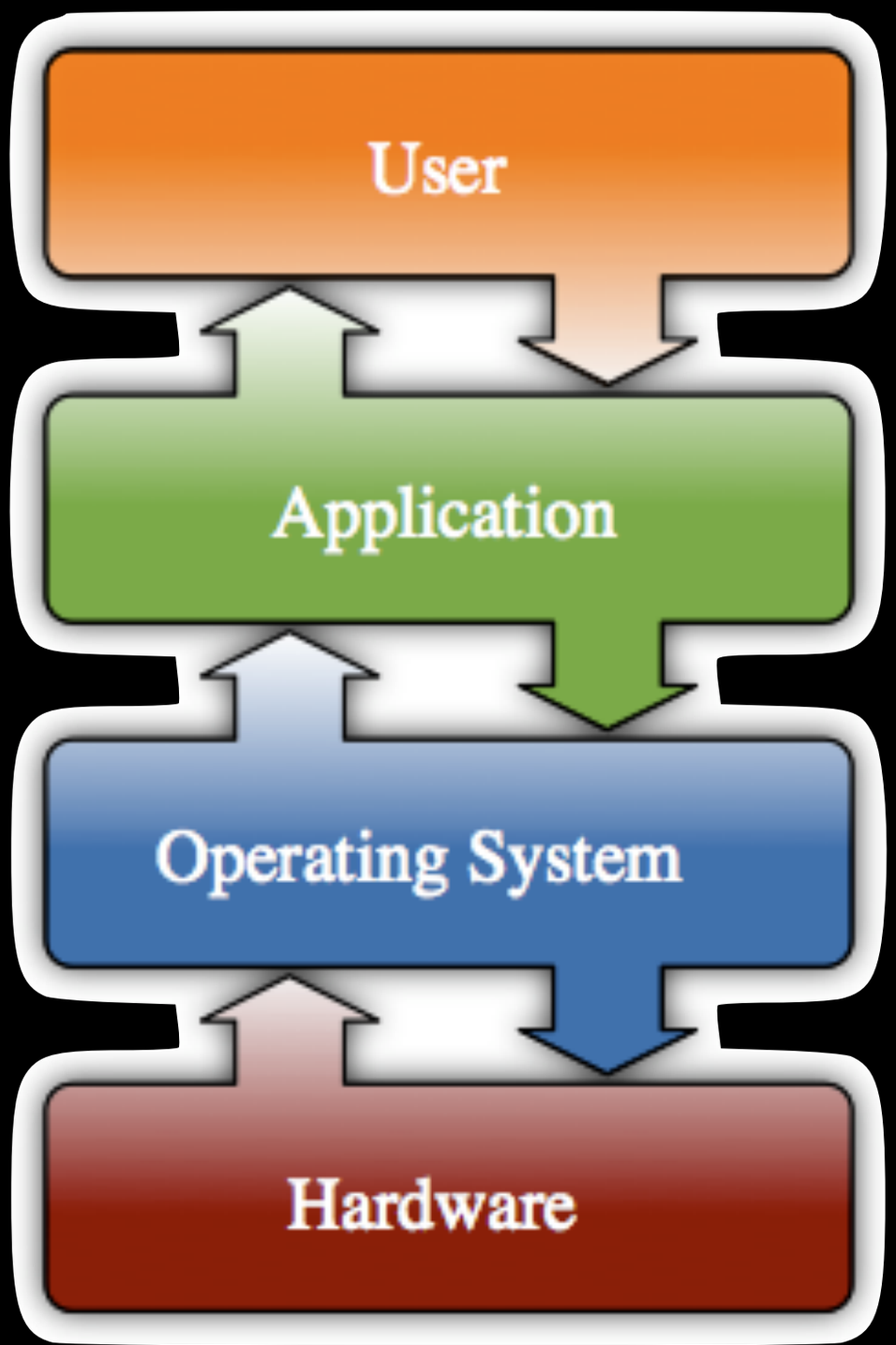
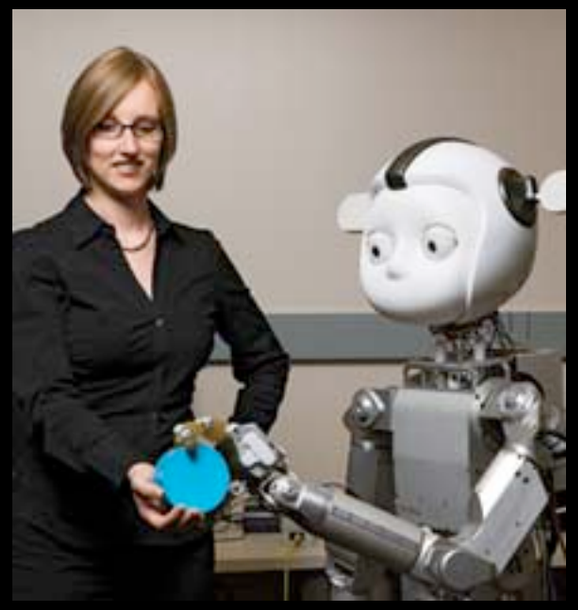
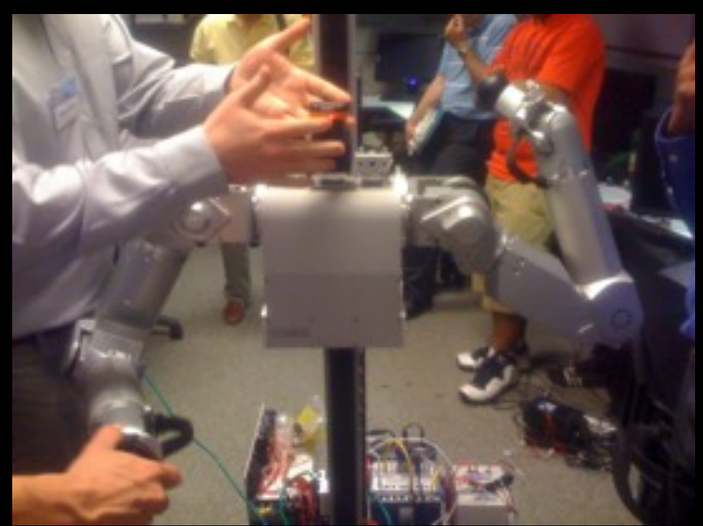




Topic 2:

Robot Middleware

app-stracting robot hardware



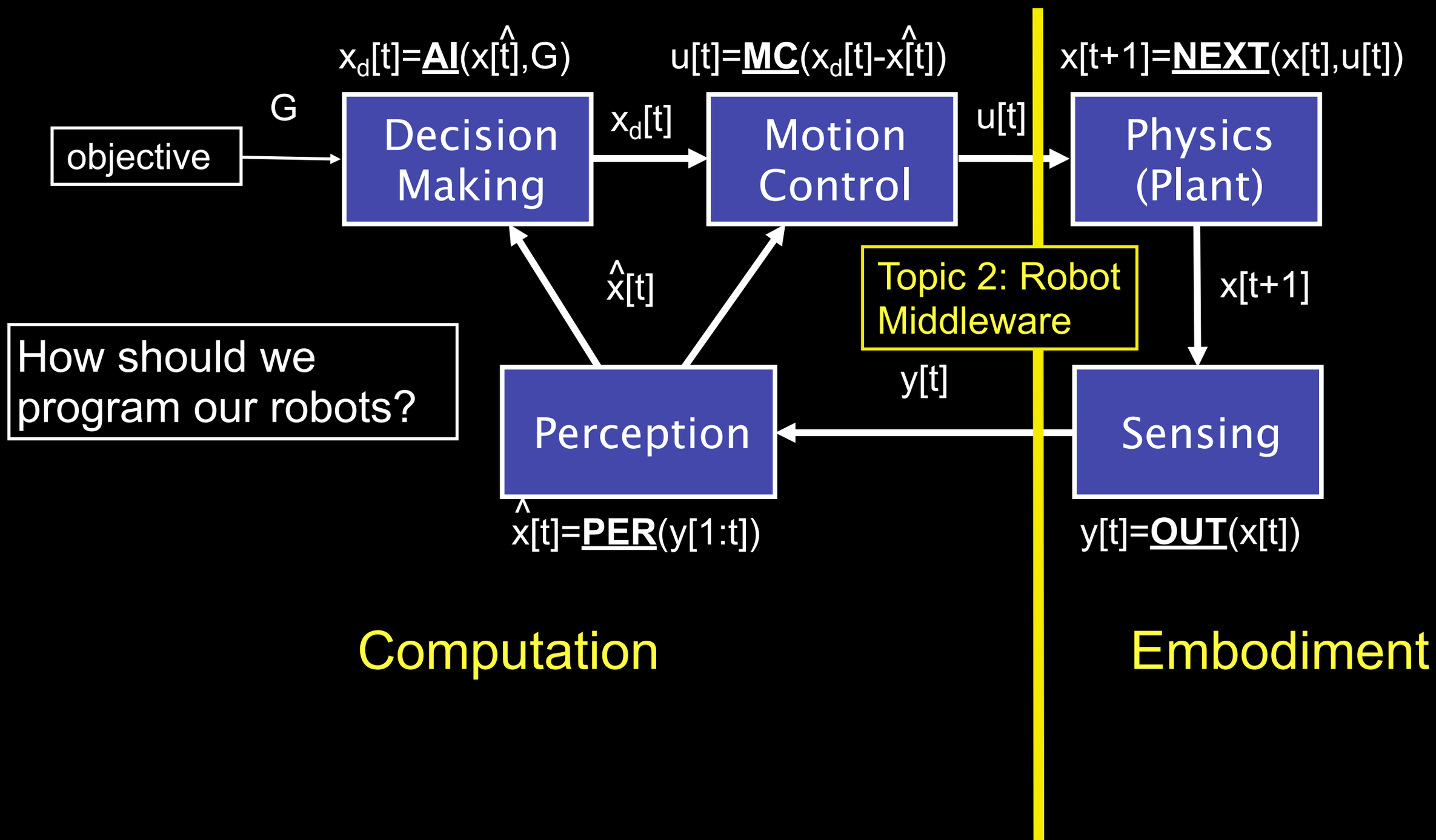
NewScientist

Robots to get their own operating system

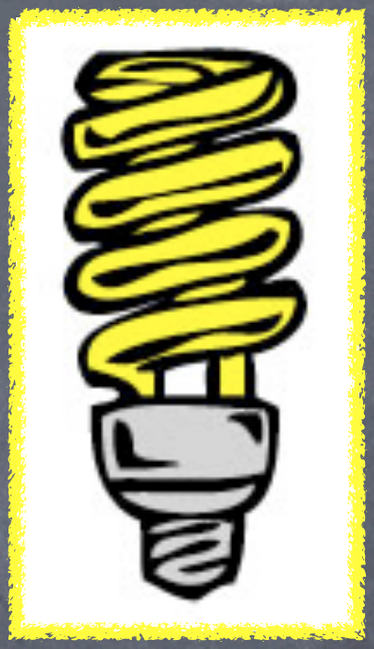
robot control loop

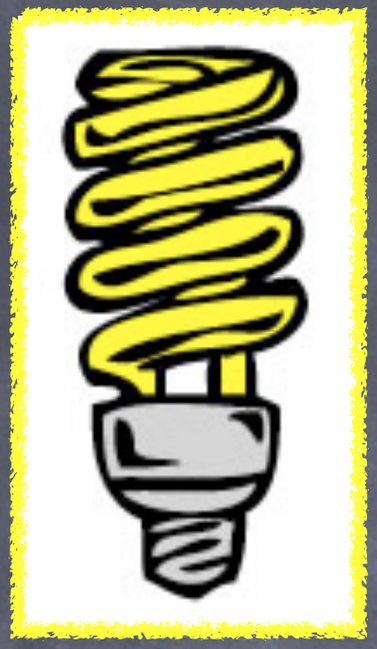
- someone please sketch on the board

The Robot Control Loop



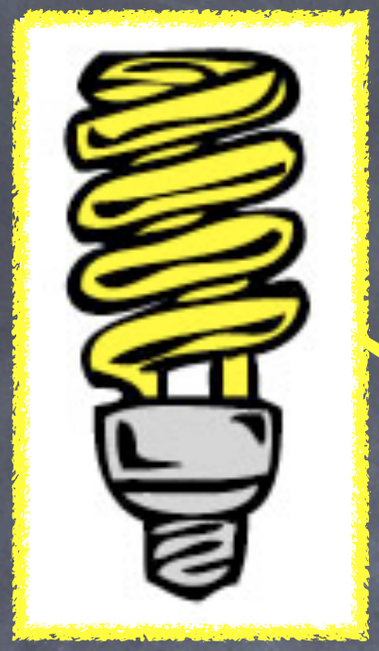
How to get your code on this robot?





USB

USB-serial



Building and running control code to interface with these devices?

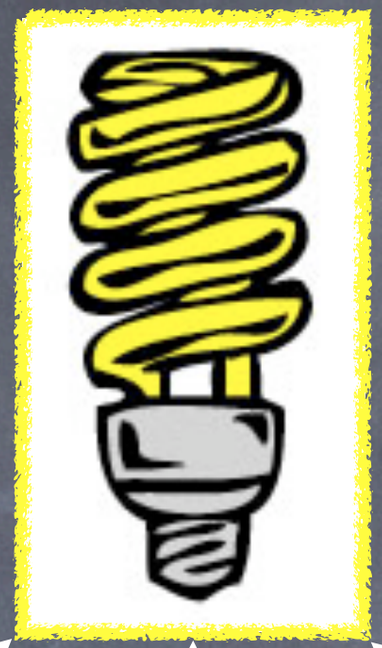
USB

USB-serial

Straightforward approach

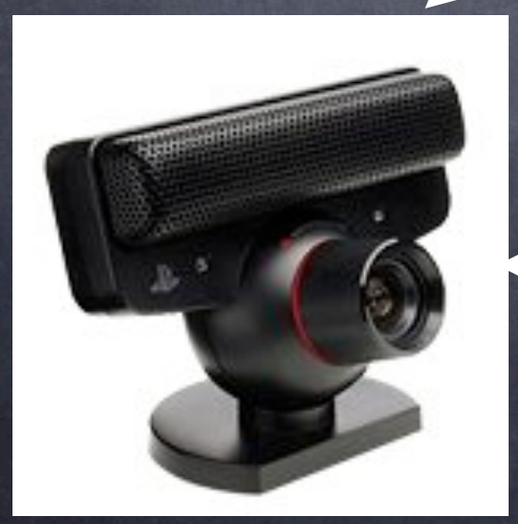
- Just write and compile a program to perform robot's "cognitive" functions
- Hardcode to specific devices?
- Scalability to growth in functionality?
- Portability to other robots?
- Reusability for other projects?
- Interoperability btwn functions/languages?
- How useful is a "one-off" solution?

Software



specific PS3
camera driver

direct serial
communication



USB

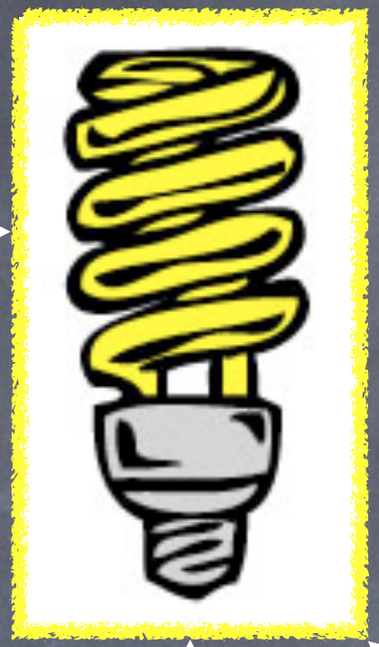
USB-
serial

Hardware

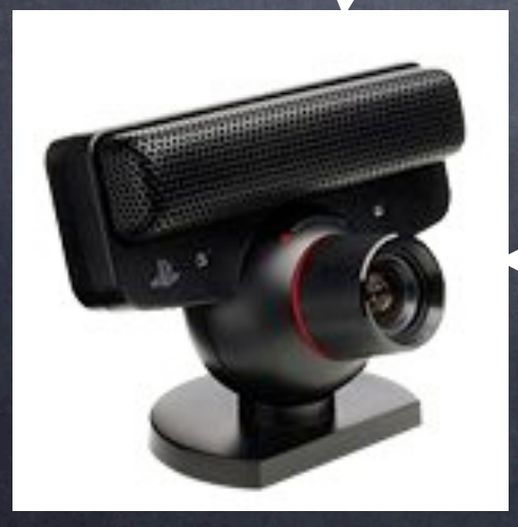
Direct serial programming

- create open programming interface
- player_comms.cc (link)
- open serial connection
- main loop over polling create devices
- sources of latency?

video4linux,
Gstreamer



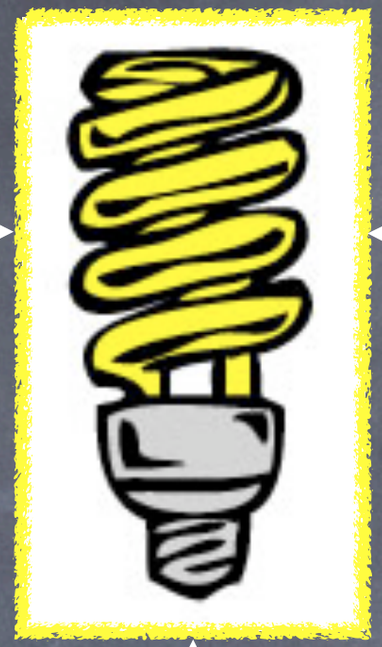
direct serial
communication



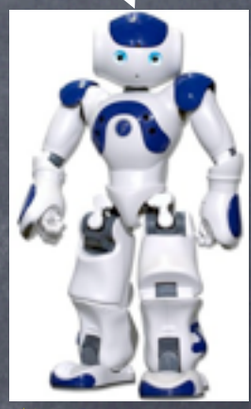
USB

USB-
serial

video4linux,
Gstreamer



robot
middleware

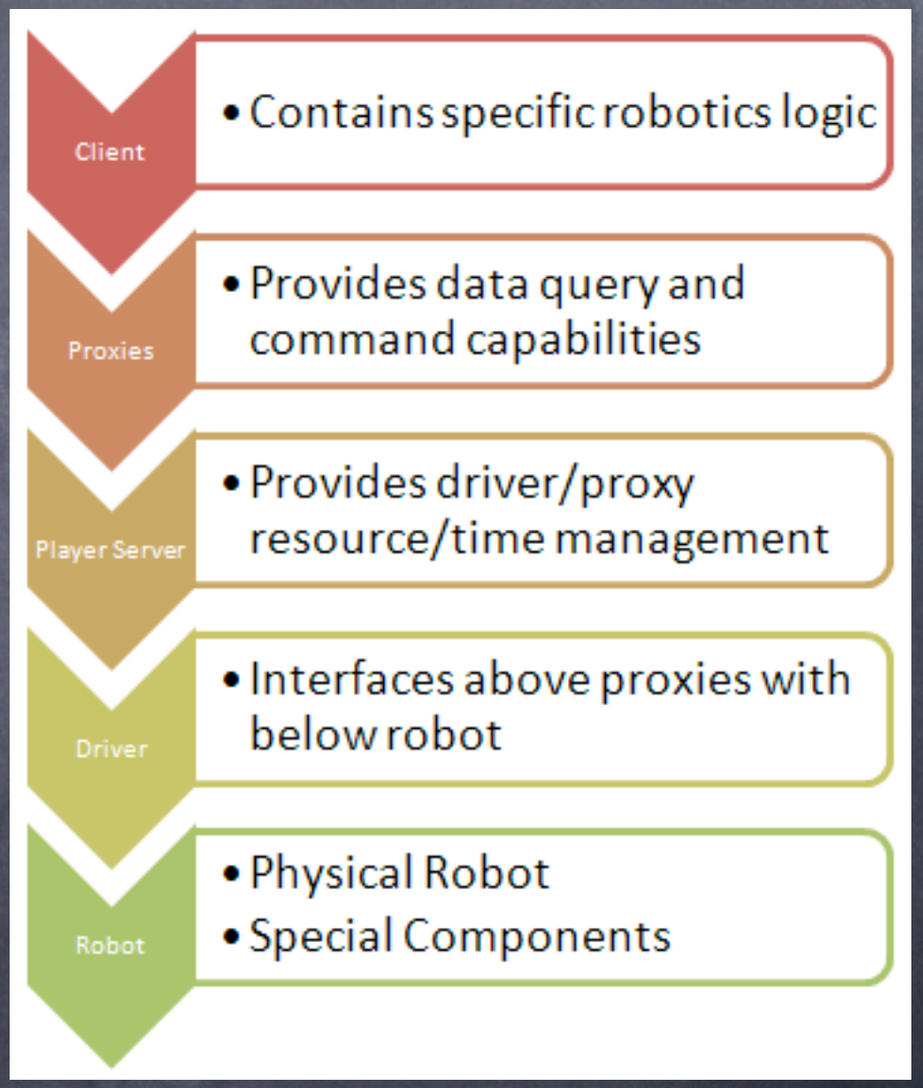


USB

USB-
serial

Robot Middleware

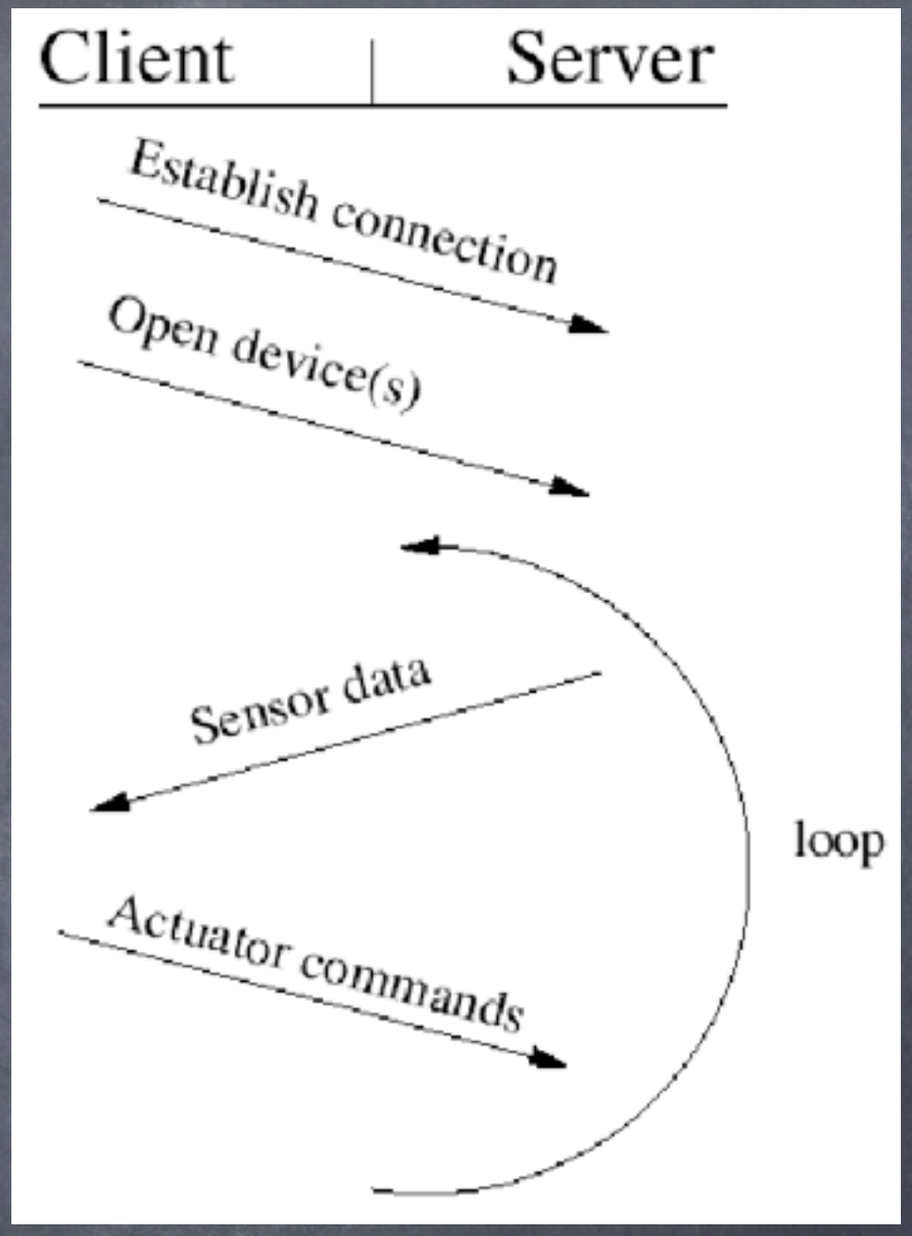
- Provide an abstraction layer and drivers between computation and embodiment
  - Analogy: hardware abstraction layer in an operating system
- Several existing packages
  - Player: client/server model
  - ROS: peer-to-peer model



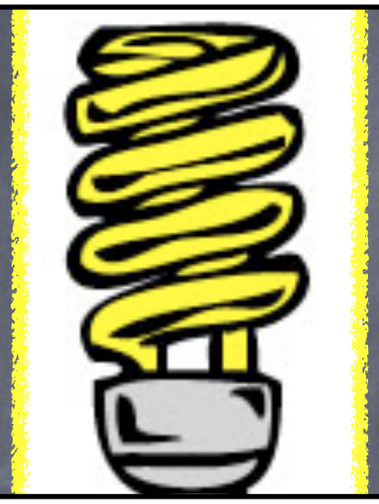
Player

(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()



custom controller

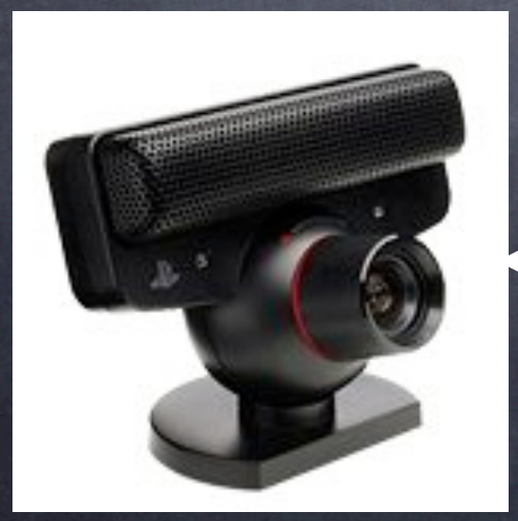


Player robot client

subscribe

publish

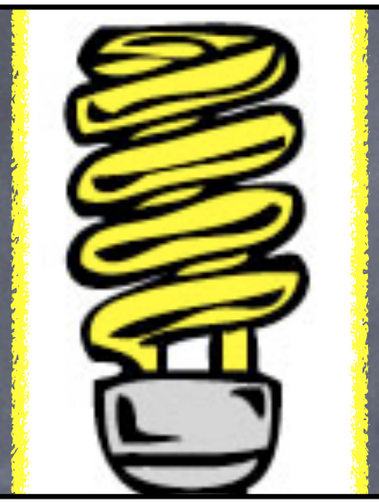
Player robot server



USB

USB-serial

custom controller



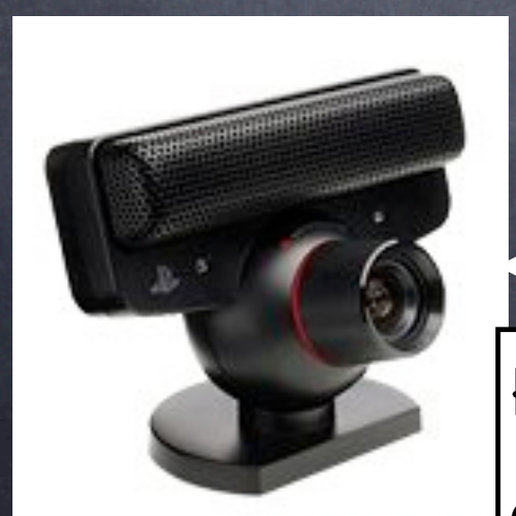
Player robot client

subscribe

network

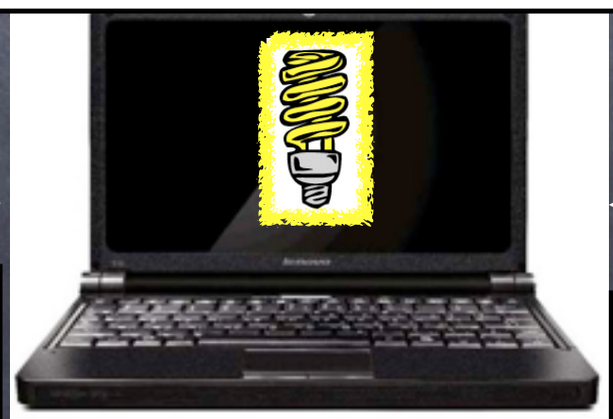
publish

Player robot server



USB

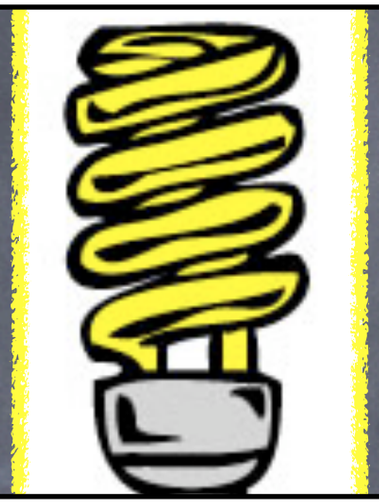
PS3 cam driver



USB-serial

Create driver

custom controller



camera proxy

Player robot client

position2d proxy

subscribe

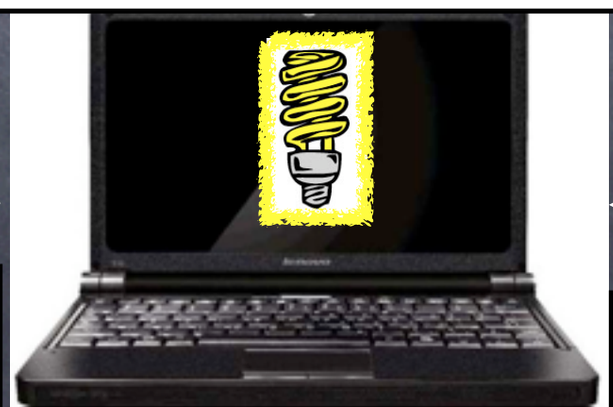
network

publish

USB

PS3 cam driver

Player robot server



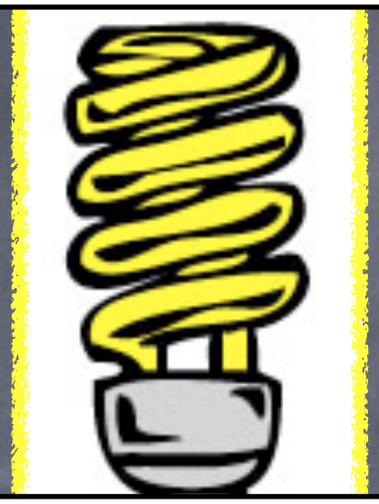
USB-serial

Create driver

cmvision library

custom controller

blobfinder proxy



camera proxy

Player robot client

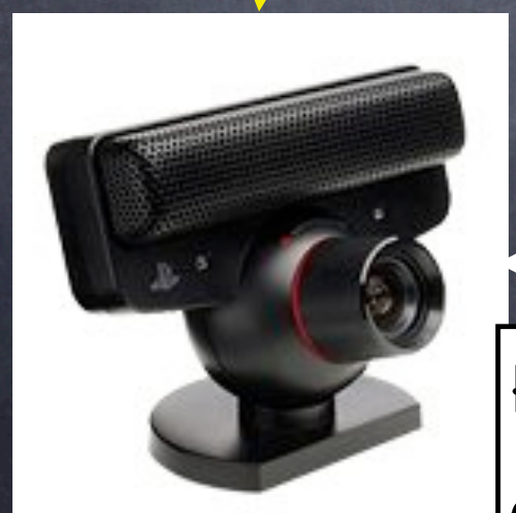
position2d proxy

subscribe

network

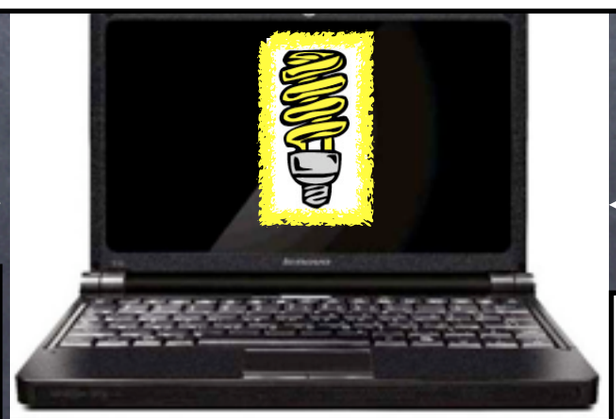
publish

Player robot server



USB

PS3 cam driver



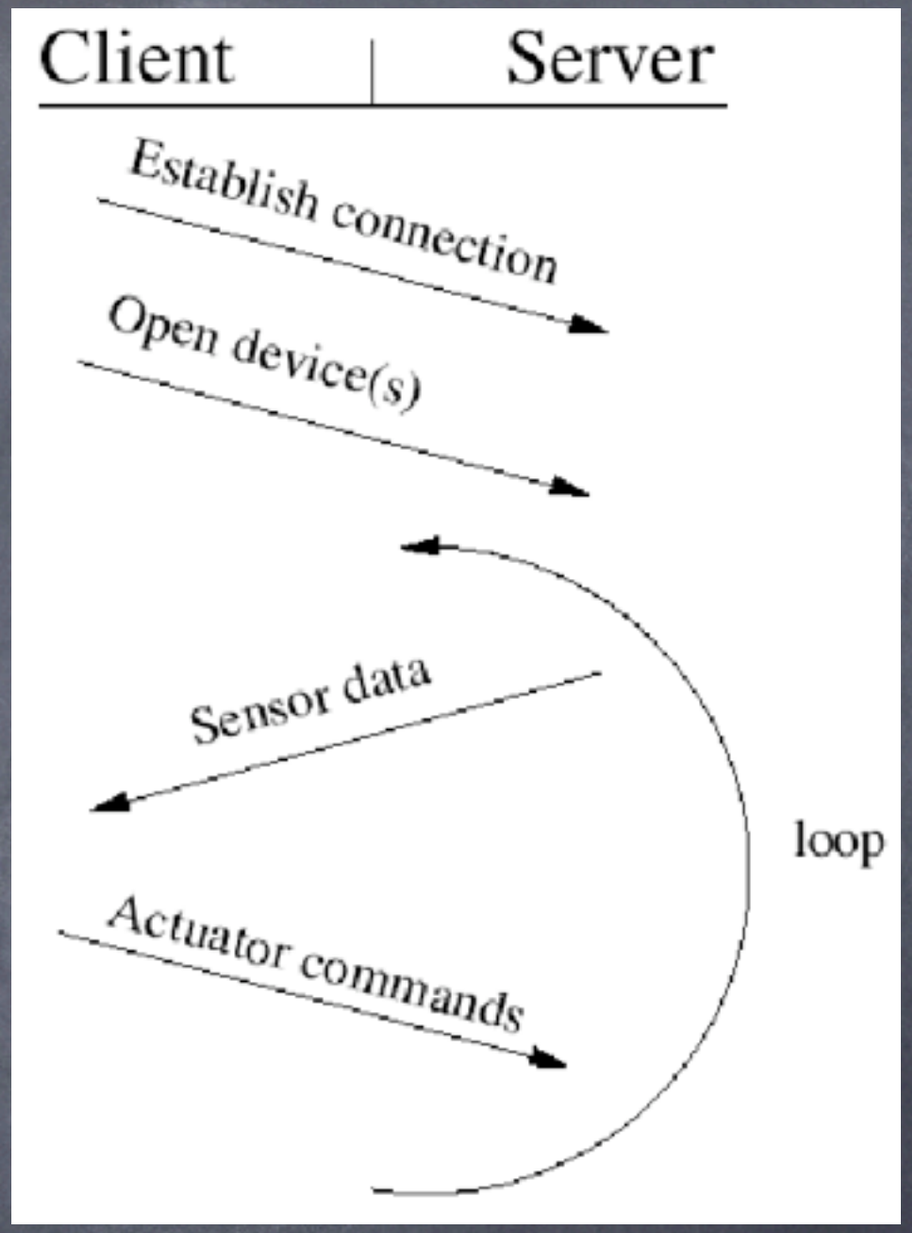
USB-serial

Create driver

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 was 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

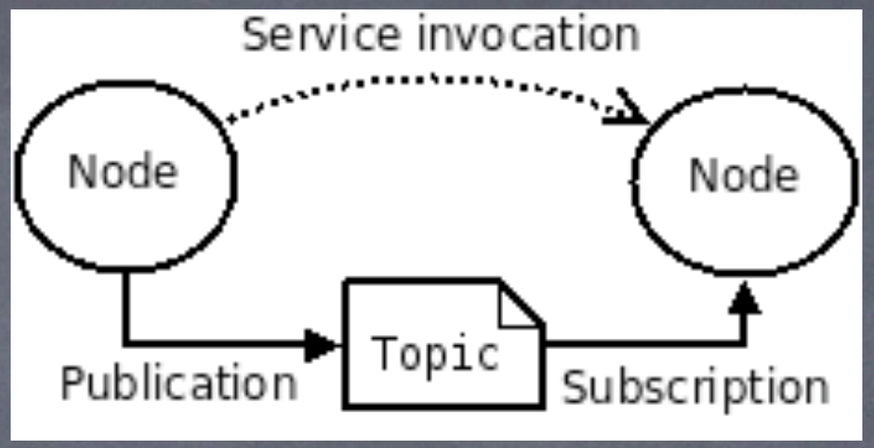
- Proxies of interest
  - position2d: planar movement and odometry
  - bumper
  - ir: virtual walls, floor detect
  - camerauvc: usb camera
  - camera1394: firewire camera
  - blobfinder: color recognition

- "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;
}
```

- add compile line

ROS (ros.org)

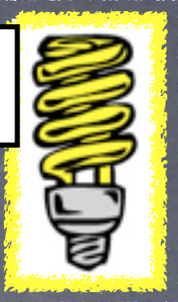


- "Robot Operating System"
- Peer-to-peer architecture over network
- Every robot function (processing) is a node
  - Node essentially its own executing process
- Nodes subscribe to and publish on topics
  - Master node runs topic name service
- Topics: basic message structure for published data
- Services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

cmvision library

blobfinder node

custom controller



controller node

ROS Master

Each node is a separate process.

Nodes talk directly to each other over network.

camera node

PS3 cam driver

USB

USB-serial

Create node

Create driver

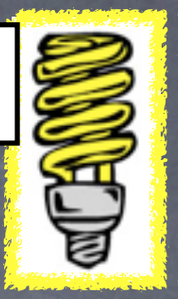
cmvision library

blobfinder node

publish
"blobs"

subscribe
"blobs"

custom controller



controller node

subscribe
"image"

publish
"twist"

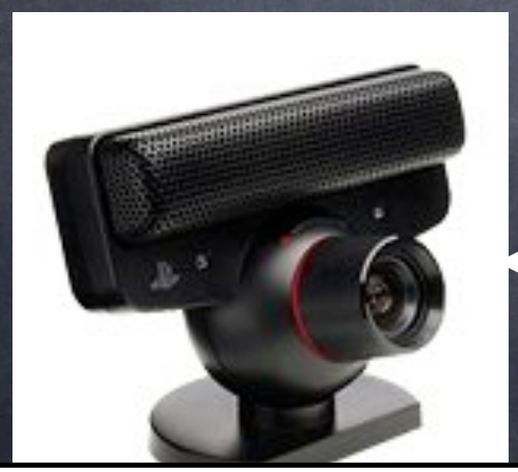
ROS Master

publish
"image"

Nodes publish and subscribe to topics.

subscribe
"twist"

camera node



PS3 cam driver

USB

USB-
serial

Create node

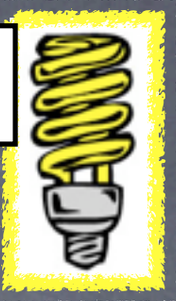
Create driver

cmvision library

publish
"blobs"

subscribe
"blobs"

custom controller



blobfinder node

controller node

subscribe
"image"

publish
"twist"

ROS Master

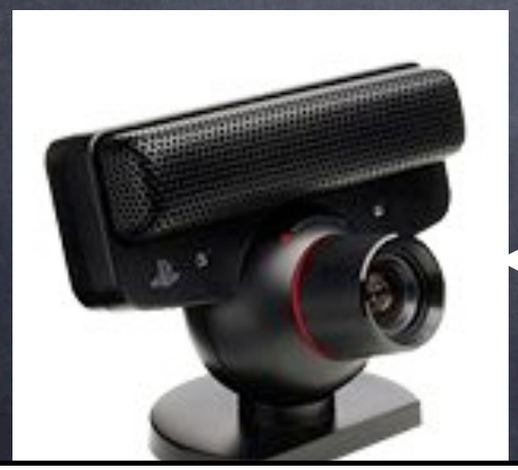
publish
"image"

Master node performs
matchmaking for topics

subscribe
"twist"

camera node

Create node



USB

USB-
serial

PS3 cam driver

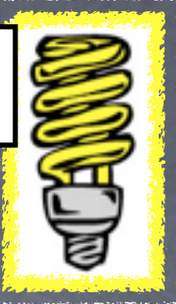
Create driver

cmvision library

publish "blobs"

subscribe "blobs"

custom controller



blobfinder node

controller node

subscribe "image"

publish "twist"

ROS Master

Nodes talk directly once matched.

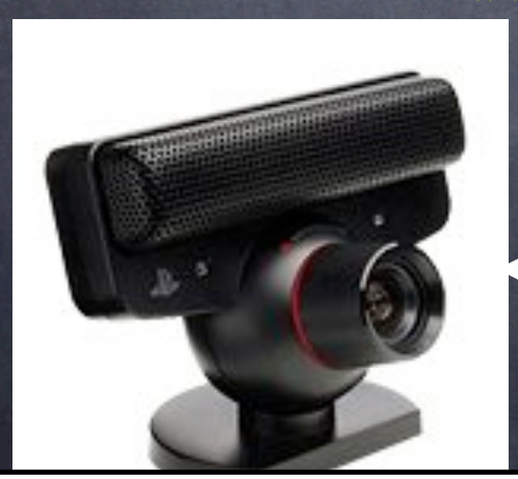
publish "image"

Master listens for new topic requests

subscribe "twist"

camera node

Create node



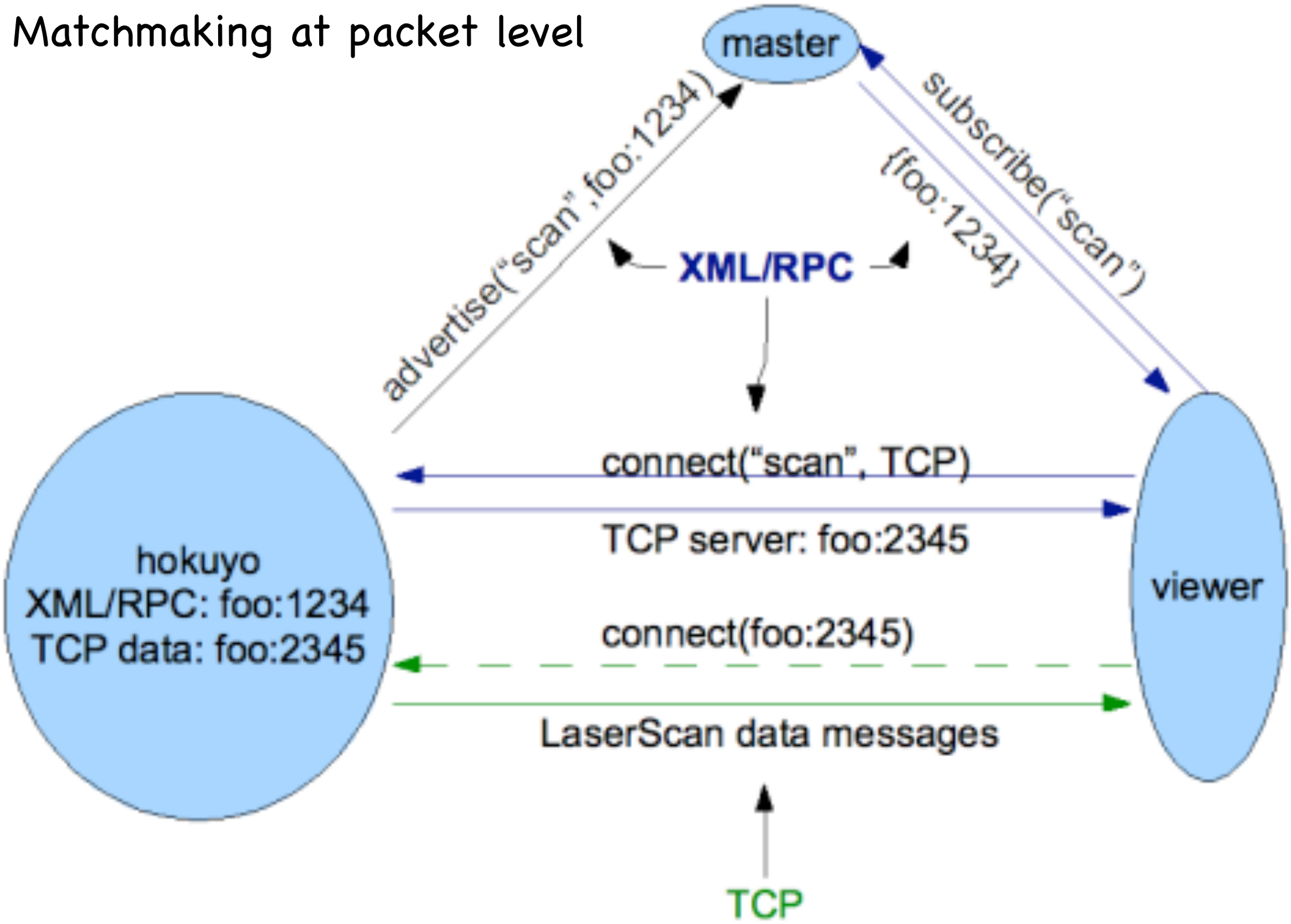
PS3 cam driver

USB

USB-serial

Create driver

Matchmaking at packet level



ROS: as a development environment

- Each project will be a "package", ROS's basic dev unit
  - Packages are built essentially by CMake
- Client libraries (eg, roscpp, rospy, rosjs, rosjava)
- Package management: integrated build system
  - roscreate-pkg to create a package
  - rosmake to build, rosrun to execute nodes
- Integration of external packages and repositories
  - OpenCV, OpenRAVE, Player, etc.
- brown-ros-pkg contains drivers for Create, PS3 cam...

```
#!/usr/bin/env python
import roslib; roslib.load_manifest('hello_create')
import rospy
#from std_msgs.msg import String
from geometry_msgs.msg import Twist
# listen
from irobot_create_2_1.msg import SensorPacket
# global variables
bump = False

# listen (adapted from line_follower
def processSensing(sensorPacket):
    global bump
    bump = sensorPacket.bumpLeft or sensorPacket.bumpRight
    #newInfo = True


def create_spin_and_bump():
    pub = rospy.Publisher('cmd_vel', Twist)
    rospy.Subscriber('sensorPacket', SensorPacket, processSensing)
    rospy.init_node('create_spin_and_bump')
    #listen
    global bump
    twist = Twist()
    while not rospy.is_shutdown():
        if bump:
            str = "hello create, you have bumped into something  %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0.0; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = -0.5
            bump = False
        else:
            str = "hello create, you can spin now %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0.1; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = 0.5
        pub.publish(twist)
        rospy.sleep(1.0)
if __name__ == '__main__':
    try:
        create_spin_and_bump()
    except rospy.ROSInterruptException: pass
```

invoke python

My first python program "hello_create"

```
#!/usr/bin/env python
import roslib; roslib.load_manifest('hello_create')
import rospy
#from std_msgs.msg import String
from geometry_msgs.msg import Twist
# listen
from irobot_create_2_1.msg import SensorPacket
# global variables
bump = False

# listen (adapted from line_follower
def processSensing(sensorPacket):
    global bump
    bump = sensorPacket.bumpLeft or sensorPacket.bumpRight
    #newInfo = True


def create_spin_and_bump():
    pub = rospy.Publisher('cmd_vel', Twist)
    rospy.Subscriber('sensorPacket', SensorPacket, processSensing)
    rospy.init_node('create_spin_and_bump')
    #listen
    global bump
    twist = Twist()
    while not rospy.is_shutdown():
        if bump:
            str = "hello create, you have bumped into something  %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0.0; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = -0.5
            bump = False
        else:
            str = "hello create, you can spin now %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0.1; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = 0.5
        pub.publish(twist)
        rospy.sleep(1.0)
if __name__ == '__main__':
    try:
        create_spin_and_bump()
    except rospy.ROSInterruptException: pass
```

python includes

My first python program "hello_create"

```
#!/usr/bin/env python
import roslib; roslib.load_manifest('hello_create')
import rospy
#from std_msgs.msg import String
from geometry_msgs.msg import Twist
# listen
from irobot_create_2_1.msg import SensorPacket
# global variables
bump = False

# listen (adapted from line_follower
def processSensing(sensorPacket):
    global bump
    bump = sensorPacket.bumpLeft or sensorPacket.bumpRight
    #newInfo = True


def create_spin_and_bump():
    pub = rospy.Publisher('cmd_vel', Twist)
    rospy.Subscriber('sensorPacket', SensorPacket, processSensing)
    rospy.init_node('create_spin_and_bump')
    #listen
    global bump
    twist = Twist()
    while not rospy.is_shutdown():
        if bump:
            str = "hello create, you have bumped into something  %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0.0; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = -0.5
            bump = False
        else:
            str = "hello create, you can spin now %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0.1; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = 0.5
        pub.publish(twist)
        rospy.sleep(1.0)
if __name__ == '__main__':
    try:
        create_spin_and_bump()
    except rospy.ROSInterruptException: pass
```

import topic definitions

My first python program "hello_create"

```
#!/usr/bin/env python
import roslib; roslib.load_manifest('hello_create')
import rospy
#from std_msgs.msg import String
from geometry_msgs.msg import Twist
# listen
from irobot_create_2_1.msg import SensorPacket
# global variables
bump = False

# listen (adapted from line_follower
def processSensing(sensorPacket):
    global bump
    bump = sensorPacket.bumpLeft or sensorPacket.bumpRight
    #newInfo = True


def create_spin_and_bump():
    pub = rospy.Publisher('cmd_vel', Twist)
    rospy.Subscriber('sensorPacket', SensorPacket, processSensing)
    rospy.init_node('create_spin_and_bump')
    #listen
    global bump
    twist = Twist()
    while not rospy.is_shutdown():
        if bump:
            str = "hello create, you have bumped into something  %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0.0; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = -0.5
            bump = False
        else:
            str = "hello create, you can spin now %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0.1; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = 0.5
        pub.publish(twist)
        rospy.sleep(1.0)
if __name__ == '__main__':
    try:
        create_spin_and_bump()
    except rospy.ROSInterruptException: pass
```

global variable

My first python program "hello_create"

My first python program "hello_create"

```
#!/usr/bin/env python
import roslib; roslib.load_manifest('hello_create')
import rospy
#from std_msgs.msg import String
from geometry_msgs.msg import Twist
# listen
from irobot_create_2_1.msg import SensorPacket
# global variables
bump = False

# listen (adapted from line_follower
def processSensing(sensorPacket):
    global bump
    bump = sensorPacket.bumpLeft or sensorPacket.bumpRight
    #newInfo = True


def create_spin_and_bump():
    pub = rospy.Publisher('cmd_vel', Twist)
    rospy.Subscriber('sensorPacket', SensorPacket, processSensing)
    rospy.init_node('create_spin_and_bump')
    #listen
    global bump
    twist = Twist()
    while not rospy.is_shutdown():
        if bump:
            str = "hello create, you have bumped into something  %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0.0; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = -0.5
            bump = False
        else:
            str = "hello create, you can spin now %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0.1; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = 0.5
        pub.publish(twist)
        rospy.sleep(1.0)
if __name__ == '__main__':
    try:
        create_spin_and_bump()
    except rospy.ROSInterruptException: pass
```

start event loop for create_spin_and bump()

My first python program "hello_create"

```
#!/usr/bin/env python
import roslib; roslib.load_manifest('hello_create')
import rospy
#from std_msgs.msg import String
from geometry_msgs.msg import Twist
# listen
from irobot_create_2_1.msg import SensorPacket
# global variables
bump = False

# listen (adapted from line_follower
def processSensing(sensorPacket):
    global bump
    bump = sensorPacket.bumpLeft or sensorPacket.bumpRight
    #newInfo = True


def create_spin_and_bump():
    pub = rospy.Publisher('cmd_vel', Twist)
    rospy.Subscriber('sensorPacket', SensorPacket, processSensing)
    rospy.init_node('create_spin_and_bump')
    #listen
    global bump
    twist = Twist()
    while not rospy.is_shutdown():
        if bump:
            str = "hello create, you have bumped into something  %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0.0; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = -0.5
            bump = False
        else:
            str = "hello create, you can spin now %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0.1; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = 0.5
        pub.publish(twist)
        rospy.sleep(1.0)
if __name__ == '__main__':
    try:
        create_spin_and_bump()
    except rospy.ROSInterruptException: pass
```

initialize node and topics

My first python program "hello_create"

```
#!/usr/bin/env python
import roslib; roslib.load_manifest('hello_create')
import rospy
#from std_msgs.msg import String
from geometry_msgs.msg import Twist
# listen
from irobot_create_2_1.msg import SensorPacket
# global variables
bump = False

# listen (adapted from line_follower
def processSensing(sensorPacket):
    global bump
    bump = sensorPacket.bumpLeft or sensorPacket.bumpRight
    #newInfo = True


def create_spin_and_bump():
    pub = rospy.Publisher('cmd_vel', Twist)
    rospy.Subscriber('sensorPacket', SensorPacket, processSensing)
    rospy.init_node('create_spin_and_bump')
    #listen
    global bump
    twist = Twist()
    while not rospy.is_shutdown():
        if bump:
            str = "hello create, you have bumped into something  %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0.0; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = -0.5
            bump = False
        else:
            str = "hello create, you can spin now %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0.1; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = 0.5
        pub.publish(twist)
        rospy.sleep(1.0)
if __name__ == '__main__':
    try:
        create_spin_and_bump()
    except rospy.ROSInterruptException: pass
```

initialize node and topics

publish twist messages for Create's "cmd_vel" topic

My first python program "hello_create"

```
#!/usr/bin/env python
import roslib; roslib.load_manifest('hello_create')
import rospy
#from std_msgs.msg import String
from geometry_msgs.msg import Twist
# listen
from irobot_create_2_1.msg import SensorPacket
# global variables
bump = False

# listen (adapted from line_follower
def processSensing(sensorPacket):
    global bump
    bump = sensorPacket.bumpLeft or sensorPacket.bumpRight
    #newInfo = True


def create_spin_and_bump():
    pub = rospy.Publisher('cmd_vel', Twist)
    rospy.Subscriber('sensorPacket', SensorPacket, processSensing)
    rospy.init_node('create_spin_and_bump')
    #listen
    global bump
    twist = Twist()
    while not rospy.is_shutdown():
        if bump:
            str = "hello create, you have bumped into something  %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0.0; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = -0.5
            bump = False
        else:
            str = "hello create, you can spin now %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0.1; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = 0.5
        pub.publish(twist)
        rospy.sleep(1.0)
if __name__ == '__main__':
    try:
        create_spin_and_bump()
    except rospy.ROSInterruptException: pass
```

initialize node and topics

subscription to Create's sensorPacket topic, spawns message handler thread

My first python program "hello_create"

```
#!/usr/bin/env python
import roslib; roslib.load_manifest('hello_create')
import rospy
#from std_msgs.msg import String
from geometry_msgs.msg import Twist
# listen
from irobot_create_2_1.msg import SensorPacket
# global variables
bump = False

# listen (adapted from line_follower
def processSensing(sensorPacket):
    global bump
    bump = sensorPacket.bumpLeft or sensorPacket.bumpRight
    #newInfo = True


def create_spin_and_bump():
    pub = rospy.Publisher('cmd_vel', Twist)
    rospy.Subscriber('sensorPacket', SensorPacket, processSensing)
    rospy.init_node('create_spin_and_bump')
    #listen
    global bump
    twist = Twist()
    while not rospy.is_shutdown():
        if bump:
            str = "hello create, you have bumped into something  %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0.0; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = -0.5
            bump = False
        else:
            str = "hello create, you can spin now %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0.1; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = 0.5
        pub.publish(twist)
        rospy.sleep(1.0)
if __name__ == '__main__':
    try:
        create_spin_and_bump()
    except rospy.ROSInterruptException: pass
```

initialize node and topics

register node with master

My first python program "hello_create"

```
#!/usr/bin/env python
import roslib; roslib.load_manifest('hello_create')
import rospy
#from std_msgs.msg import String
from geometry_msgs.msg import Twist
# listen
from irobot_create_2_1.msg import SensorPacket
# global variables
bump = False

# listen (adapted from line_follower
def processSensing(sensorPacket):
    global bump
    bump = sensorPacket.bumpLeft or sensorPacket.bumpRight
    #newInfo = True


def create_spin_and_bump():
    pub = rospy.Publisher('cmd_vel', Twist)
    rospy.Subscriber('sensorPacket', SensorPacket, processSensing)
    rospy.init_node('create_spin_and_bump')
    #listen
    global bump
    twist = Twist()
    while not rospy.is_shutdown():
        if bump:
            str = "hello create, you have bumped into something  %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0.0; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = -0.5
            bump = False
        else:
            str = "hello create, you can spin now %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0.1; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = 0.5
        pub.publish(twist)
        rospy.sleep(1.0)
if __name__ == '__main__':
    try:
        create_spin_and_bump()
    except rospy.ROSInterruptException: pass
```

allocate and initialize variables

My first python program "hello_create"

```
#!/usr/bin/env python
import roslib; roslib.load_manifest('hello_create')
import rospy
#from std_msgs.msg import String
from geometry_msgs.msg import Twist
# listen
from irobot_create_2_1.msg import SensorPacket
# global variables
bump = False

# listen (adapted from line_follower
def processSensing(sensorPacket):
    global bump
    bump = sensorPacket.bumpLeft or sensorPacket.bumpRight
    #newInfo = True


def create_spin_and_bump():
    pub = rospy.Publisher('cmd_vel', Twist)
    rospy.Subscriber('sensorPacket', SensorPacket, processSensing)
    rospy.init_node('create_spin_and_bump')
    #listen
    global bump
    twist = Twist()
    while not rospy.is_shutdown():
        if bump:
            str = "hello create, you have bumped into something  %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0.0; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = -0.5
            bump = False
        else:
            str = "hello create, you can spin now %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0.1; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = 0.5
        pub.publish(twist)
        rospy.sleep(1.0)
if __name__ == '__main__':
    try:
        create_spin_and_bump()
    except rospy.ROSInterruptException: pass
```

handle event for subscribed topic

event loop

publish control

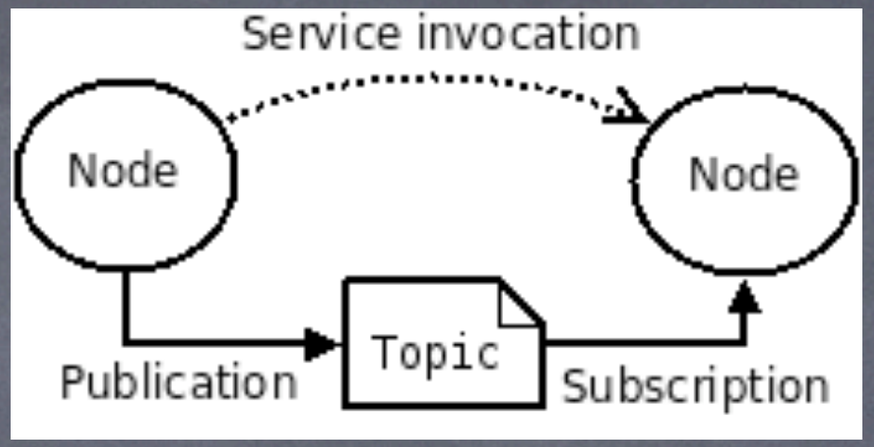
DO NOT USE sleep

```python
#!/usr/bin/env python
import roslib; roslib.load_manifest('hello_create')
import rospy
#from std_msgs.msg import String
from geometry_msgs.msg import Twist
# listen
from irobot_create_2_1.msg import SensorPacket
# global variables
bump = False

# listen (adapted from line_follower
def processSensing(sensorPacket):
    global bump
    bump = sensorPacket.bumpLeft or sensorPacket.bumpRight
    #newInfo = True


def create_spin_and_bump():
    pub = rospy.Publisher('cmd_vel', Twist)
    rospy.Subscriber('sensorPacket', SensorPacket, processSensing)
    rospy.init_node('create_spin_and_bump')
    #listen
    global bump
    twist = Twist()
    while not rospy.is_shutdown():
        if bump:
            str = "hello create, you have bumped into something  %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0.0; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = -0.5
            bump = False
        else:
            str = "hello create, you can spin now %s"%rospy.get_time()
            rospy.loginfo(str)
            twist.linear.x = 0.1; twist.linear.y = 0; twist.linear.z = 0
            twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = 0.5
        pub.publish(twist)
        rospy.sleep(1.0)
if __name__ == '__main__':
    try:
        create_spin_and_bump()
    except rospy.ROSInterruptException: pass
```

My first python program "hello_create"

What behavior will this node produce?

ROS (ros.org)



- WARNING! ROS can be a moving target
  - ROS stability issues, adaptation may be necessary
    - C-Turtle is considered stable
- we will be learning with you during projects
- extensive use of ROS mailing list (ros-users) and online documentation (ros.org)