Developing on ROS Framework

ROS packages and facilities

part 3

Rodrigo Ventura
João Reis
Institute for Systems and Robotics
Instituto Superior Técnico, Lisboa

Lisbon, 23-26 June 2013

Sponsored by project FP7-ICT-2011-9-601033 (MOnarCH)

Robot drivers

- There are ROS drivers for a large variety of robots

PR2

Husky

TurtleBot

Lego NXT

Pioneer

Nao

Care-O-bot

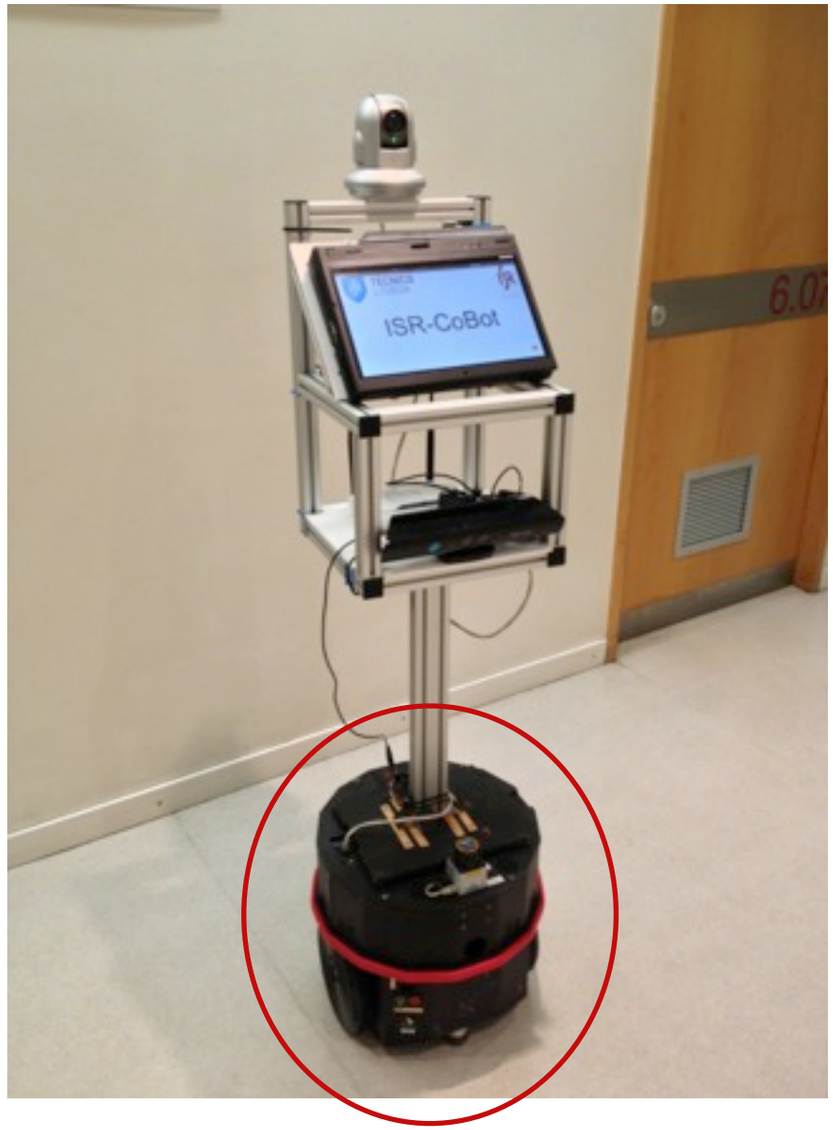
Erratic

Roomba

...

Robot drivers

- Case study: drivers for ISR-CoBot
  (Nomadic Scout platform customized by IdMind)



2x Faulhaber motor controllers

+

IdMind electronics

Robot drivers

- Case study: drivers for ISR-CoBot
  (Nomadic Scout platform customized by IdMind)
  - **packages** scout_drivers **(and** scout_msgs**)**
  - **nodes:**
    - sensors: **publishes battery status and sonar readings**
    - motors: **motor commands and encoder readings**
    - joydrive: **controlls robot motion from joystick**
  - **topics:**
    - /scout/battery **battery status**
    - /scout/sonars **sonar readings**
    - /scout/motion **motor commands**
    - /scout/motors **encoder readings**
  - **services:**
    - /scout/motion **alternative motor commands (unused)**

Robot drivers

- message types:

```
# file: ScoutBatteryMsg.msg
Header header
float32 battery1
float32 battery2
```

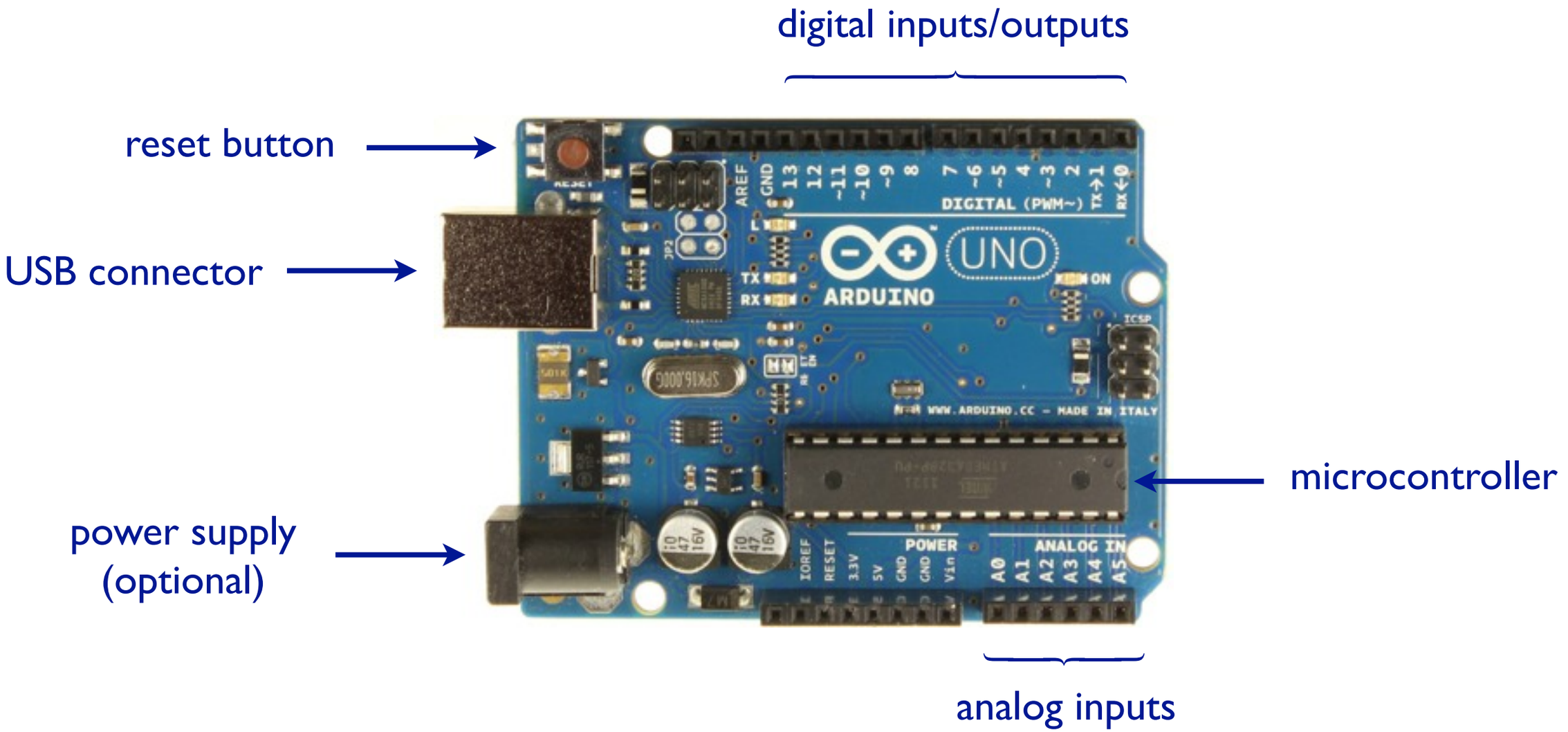
```
# file: ScoutSonarsMsg.msg
Header header
float32[] sonars
```

```
# file: ScoutMotionMsg.msg
bool  enable
int32 velocity_left
int32 velocity_right
```

```
# file: ScoutMotorsMsg.msg
Header header
int32 count_left
int32 count_right
```

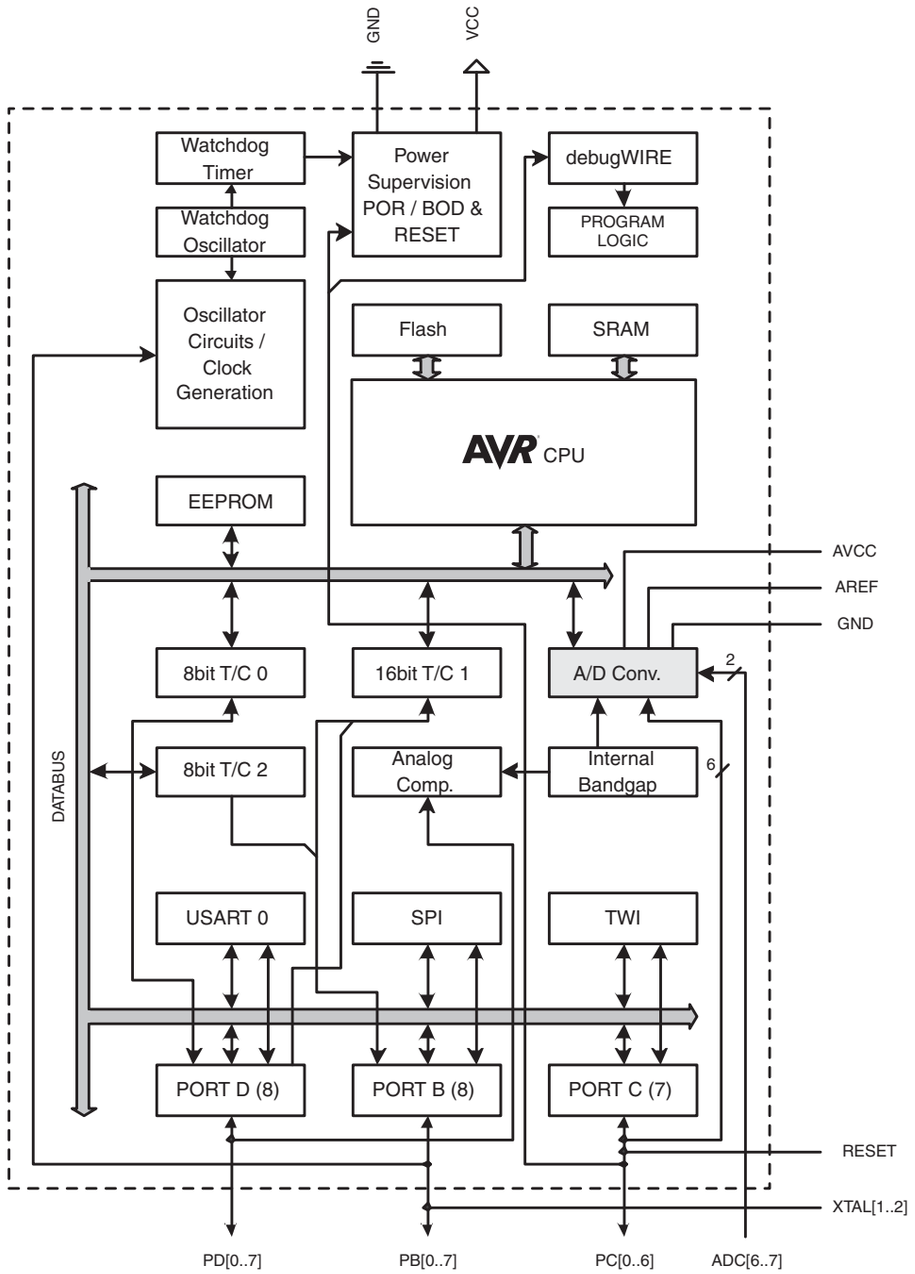
Arduino platform

- Example: Arduino Uno



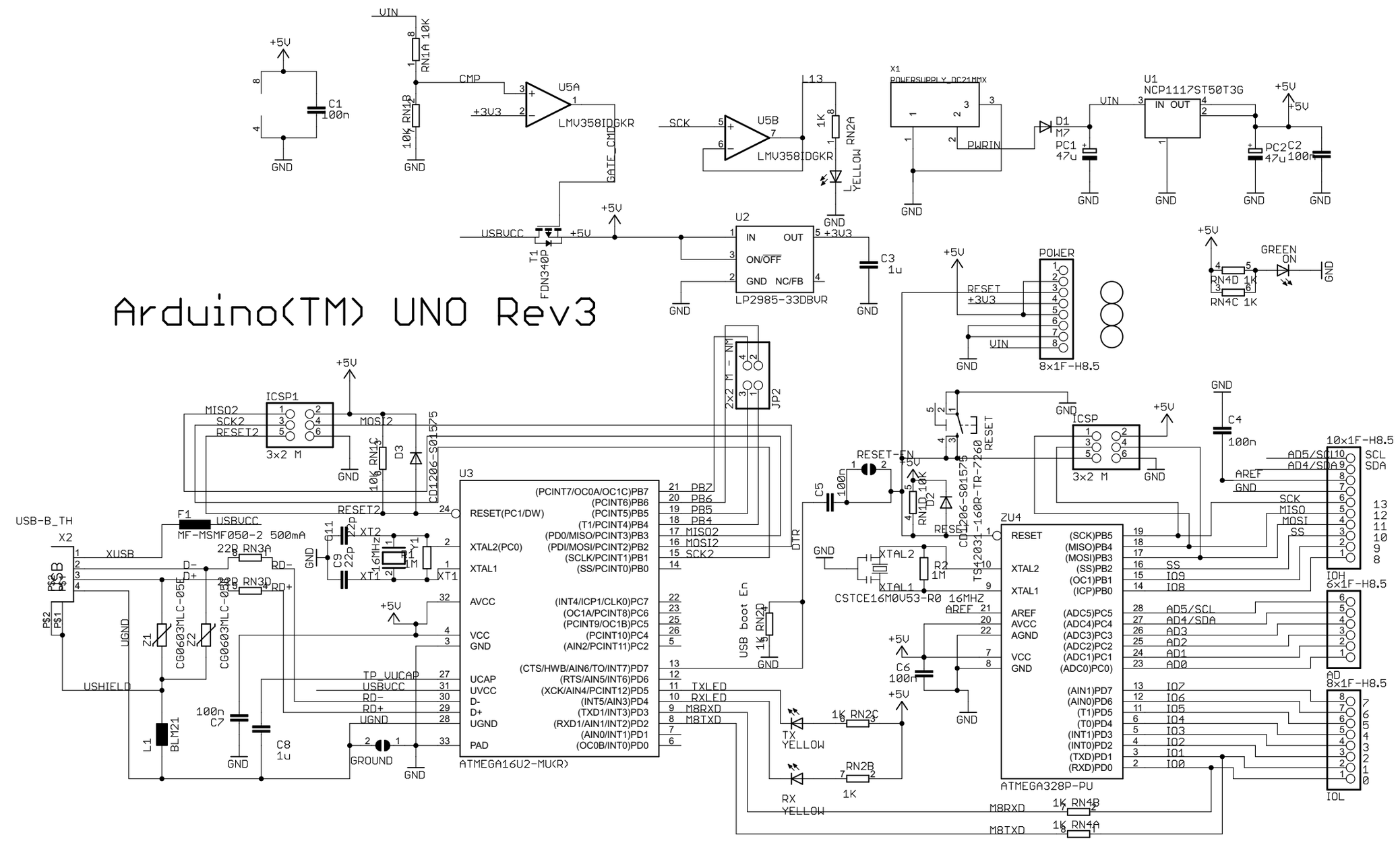
Arduino platform

- Arduino Uno microcontroller
  - ATMEL ATmega328 @ 16MHz
  - 8-bit AVR RISC core
  - 32KB flash
  - 1KB EEPROM
  - 2KB RAM



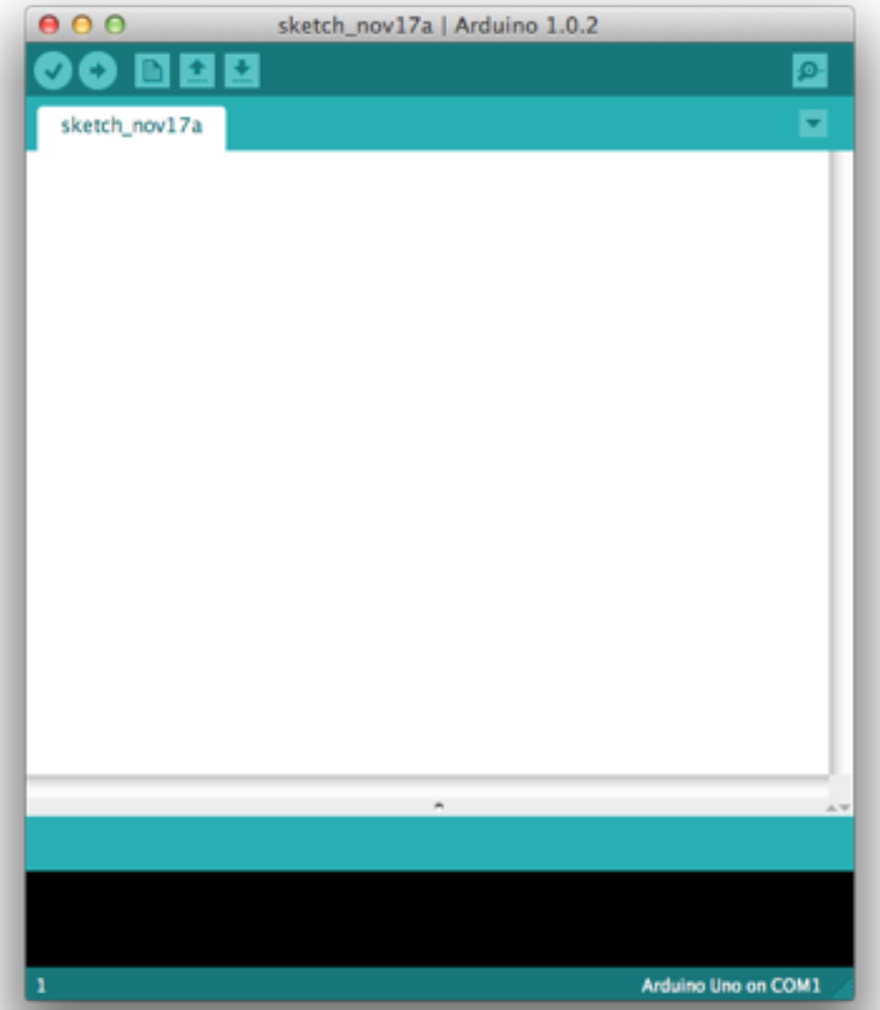
Arduino platform

- Schematic:



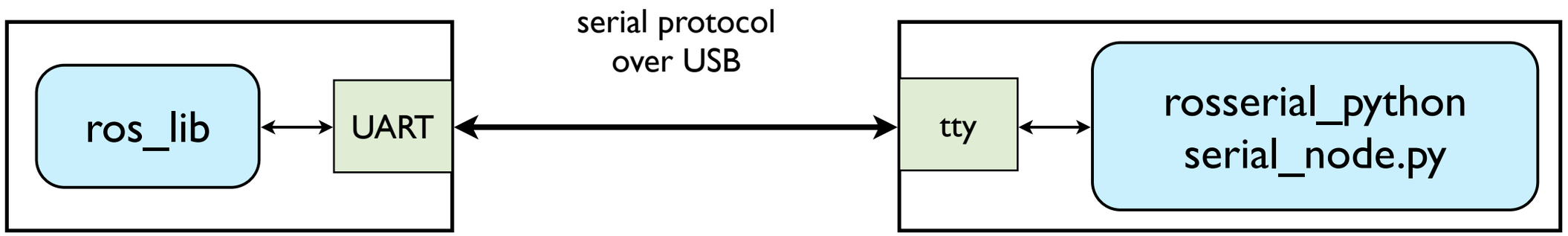
Arduino platform

- Integrated Development Environment (IDE)
  - Sketch = source file (*.ino)
  - Sketchbook = set of Sketches
  - Wiring: C/C++ based language



Using ROS with Arduino

- ROS stack rosserial **provides serialization mechanisms to stream ROS messages over the serial link (over USB) to a PC**
  - **Arduino side:** ros_lib **from** rosserial_arduino **package has to be installed into Arduino IDE libraries**
  - **PC side:** serial_node.py **node from** rosserial_python **package has to be running**



Using ROS with Arduino

- Installation and patching (for Linux Ubuntu):

1. **install** rosserial **stack**

```
sudo apt-get install ros-fuerte-rosserial
```

install ros_lib into Arduino IDE

```
roscd rosserial_arduino
cp -r libraries/ros_lib <sketckbook>/libraries
```

2. **patch** <sketchbook>/libraries/ros_lib/ArduinoHardware.h:

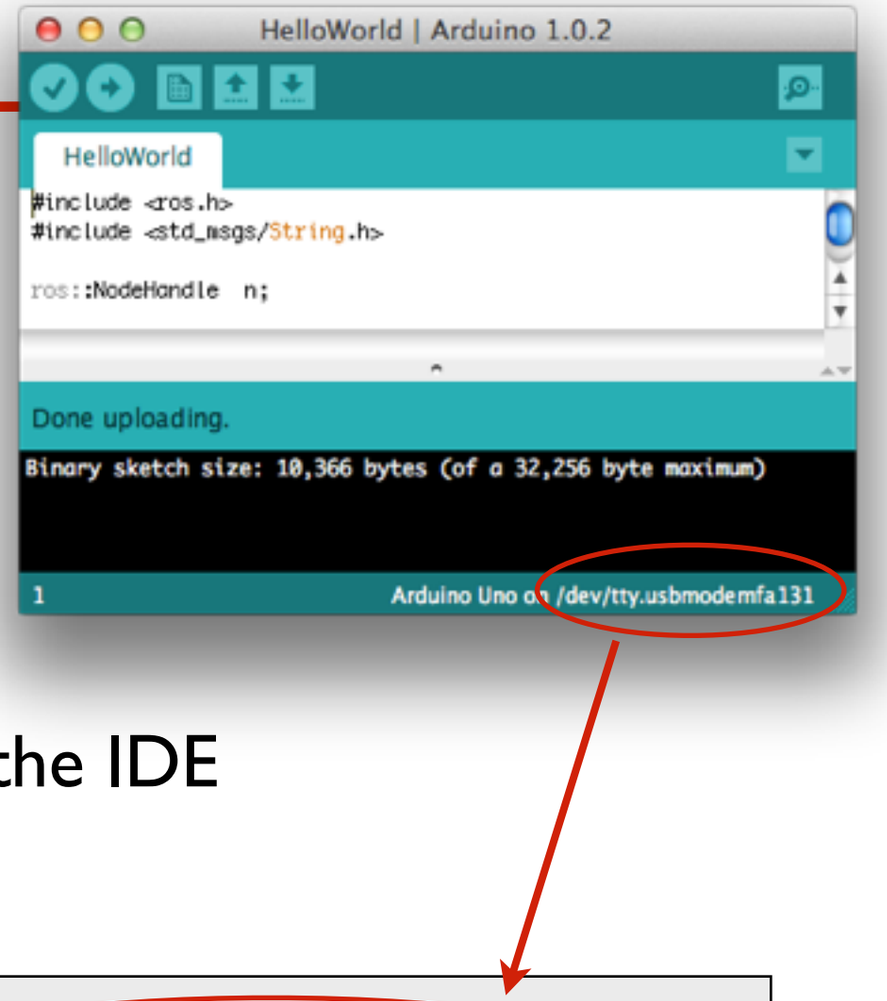
   replace *#include "WProgram.h"* ***with*** *#include "Arduino.h"*

Using ROS with Arduino

- Publishing from Arduino

```
#include <ros.h>
#include <std_msgs/String.h>

ros::NodeHandle n;

std_msgs::String msg;
ros::Publisher pub("/my_topic", &msg);

int count = 0;
char data[100];

void setup()
{
  n.initNode();
  n.advertise(pub);
}

void loop()
{
  sprintf(data, "Hello world %d", ++count);
  msg.data = data;
  pub.publish(&msg);
  n.spinOnce();
  delay(1000);
}
```

Using ROS with Arduino



- How to run this program:
  1. upload program to Arduino using the IDE
  2. run interface node in PC

```
rosrun rosserial_python serial_node.py /dev/tty.usbmodemfa131
```

6. use rostopic echo to display messages

```
rostopic echo /my_topic
```

Using ROS with Arduino

- Subsribing from Arduino

```
#include <Servo.h>
#include <ros.h>
#include <std_msgs/UInt16.h>

ros::NodeHandle nh;

Servo servo;
int led = 13;

void callback(const std_msgs::UInt16 &cmd_msg) {
  servo.write(cmd_msg.data);  // set servo angle, should be from 0-180
  digitalWrite(led, HIGH-digitalRead(led));  // toggle led
}

ros::Subscriber<std_msgs::UInt16> sub("servo", callback);

void setup() {
  nh.initNode();
  nh.subscribe(sub);

  pinMode(led, OUTPUT);
  servo.attach(9);  // attach it to pin 9
}

void loop() {
  nh.spinOnce();
  delay(1);
}
```

Using ROS with Arduino

- How to run this program:
  1. upload program to Arduino using IDE
  2. run interface node in PC

```
rosrun rosserial_python serial_node.py /dev/tty.usbmodemfa131
```

1. use rostopic pub to publish messages

```
rostopic pub /servo std_msgs/UInt16 30
```

servo angle in degrees

Using ROS with Arduino

- How to use custom message types from Arduino IDE
  1. create message type files as usual
  2. example: hello_world/msg/Adc.msg

```
uint16 adc0
uint16 adc1
uint16 adc2
uint16 adc3
uint16 adc4
uint16 adc5
```

  1. generate and install message type includes into Arduino ros_lib

```
rosrun rosserial_client make_library.py <sketchbook>/libraries hello_world
```

package name

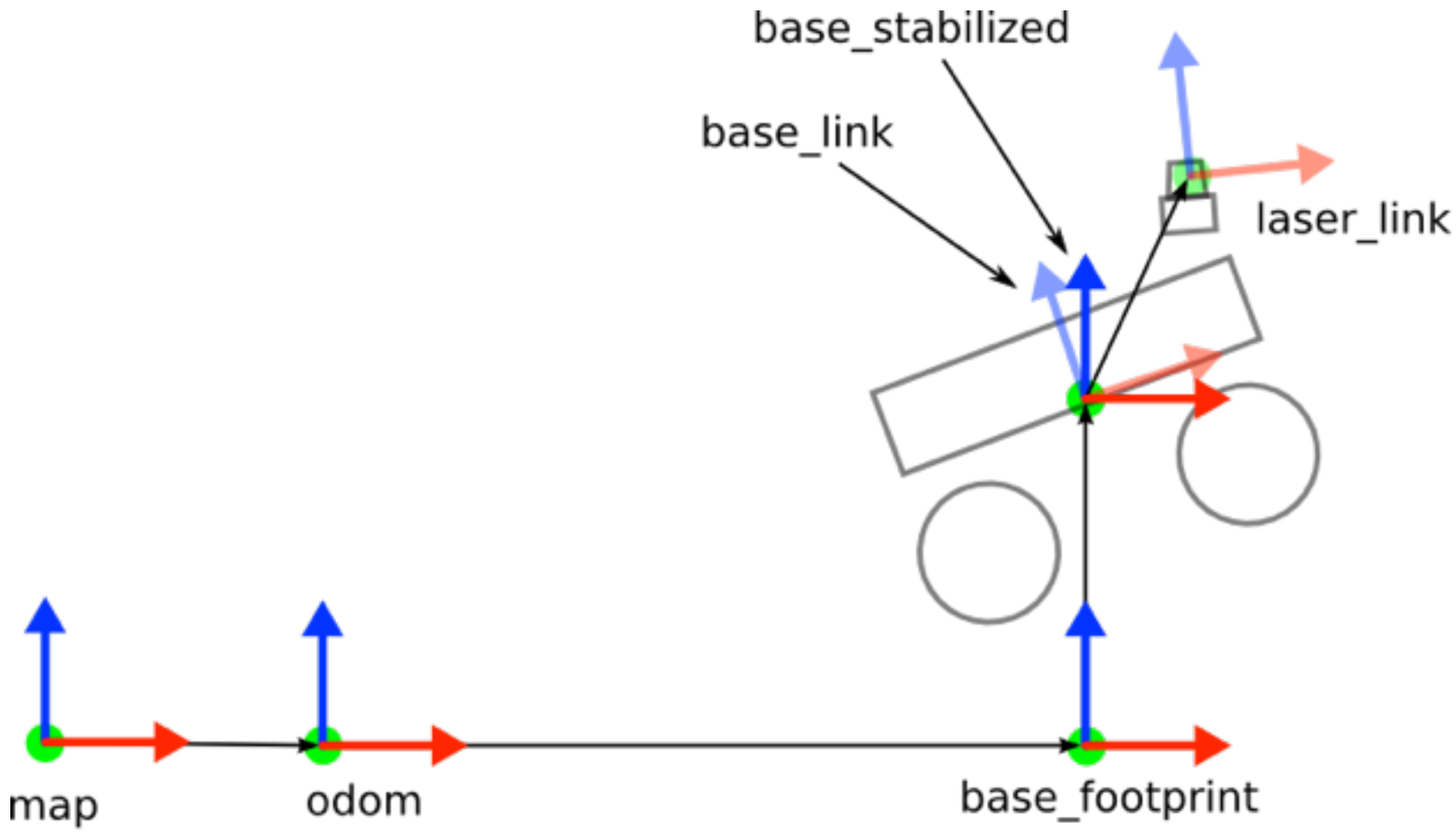
Using ROS with Arduino

- How to use custom message types from Arduino IDE

```
#include <ros.h>
#include <hello_world/Adc.h>

ros::NodeHandle nh;

hello_world::Adc msg;
ros::Publisher pub("adc", &msg);

void setup() {
  nh.initNode();
  nh.advertise(pub);
}

[...]

void loop() {
  msg.adc0 = averageAnalog(0);
  msg.adc1 = averageAnalog(1);
  msg.adc2 = averageAnalog(2);
  msg.adc3 = averageAnalog(3);
  msg.adc4 = averageAnalog(4);
  msg.adc5 = averageAnalog(5);

  pub.publish(&msg);

  nh.spinOnce();
}
```

TF package

- The package tf tracks multiple coordinate frame transformations



- broadcasters publish transforms between pairs of frames
- listeners query transforms between pairs of frames

TF package

- **Package tools:**
  - static_transform_publisher **broadcasts a fixed frame**

    arguments: x y z yaw pitch roll frame child period_in_ms
  - **Demo:**

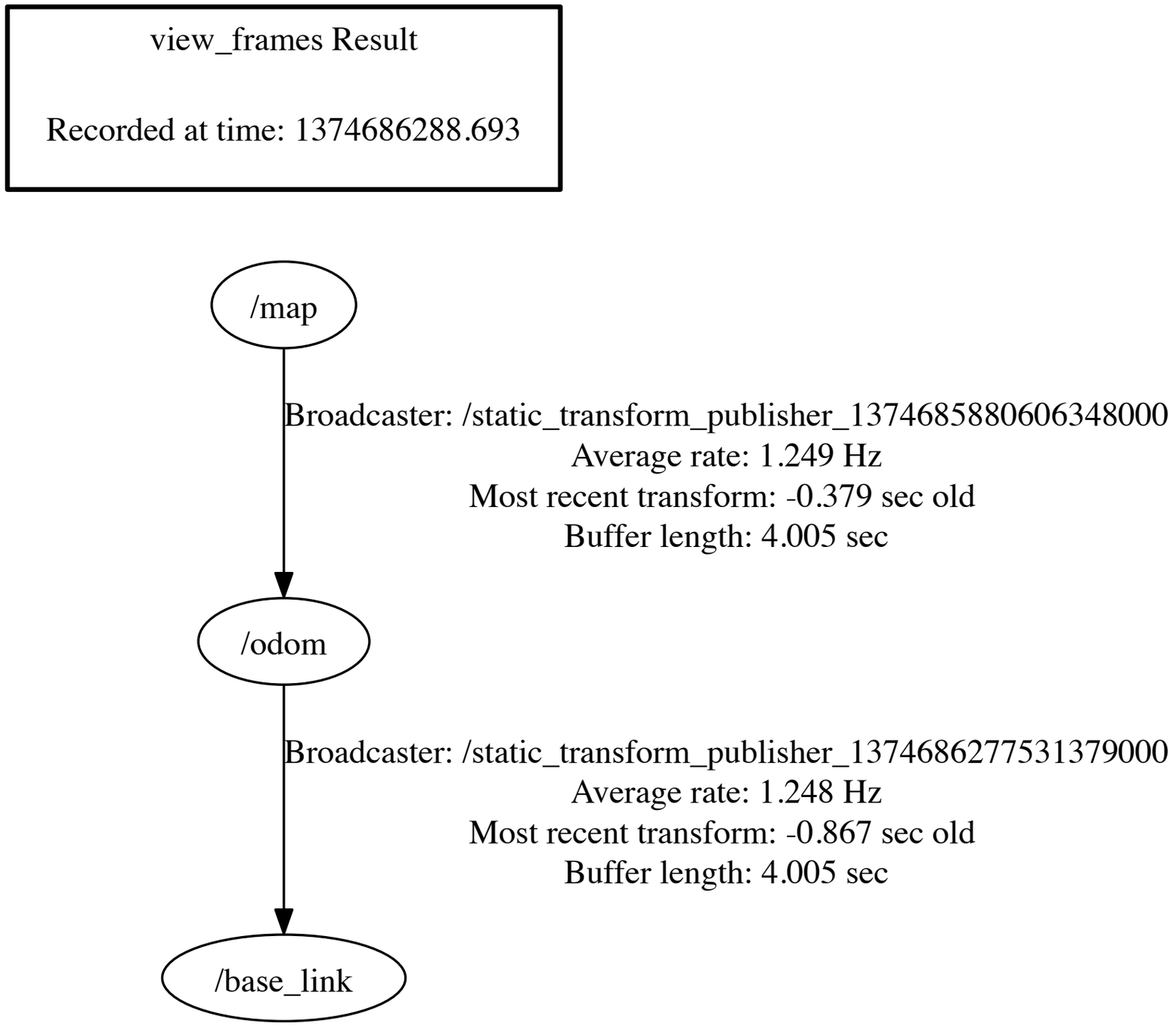
    /map → /odom → /base_link

```
$ rosrun tf static_transform_publisher 10 0 0 0 0 0 /map /odom 1000
```

```
$ rosrun tf static_transform_publisher 0 10 0 0 0 0 /odom /base_link 1000
```

TF package

- view_frames **generates a PDF with a graph of frames, e.g.:**



TF package

- tf_echo **shows transformations between a pair of frames, e.g.:**

```
$ rosrun tf tf_echo /map /base_link
At time 1374686571.905
- Translation: [10.000, 10.000, 0.000]
- Rotation: in Quaternion [0.000, 0.000, 0.000, 1.000]
            in RPY [0.000, -0.000, 0.000]
At time 1374686572.815
- Translation: [10.000, 10.000, 0.000]
- Rotation: in Quaternion [0.000, 0.000, 0.000, 1.000]
            in RPY [0.000, -0.000, 0.000]
At time 1374686573.827
- Translation: [10.000, 10.000, 0.000]
- Rotation: in Quaternion [0.000, 0.000, 0.000, 1.000]
            in RPY [0.000, -0.000, 0.000]
[...]
```

TF package

- **How to broadcast (in Python):**

  tb = tf.TransformBroadcaster()

  tb.sendTransform((x, y, z), quaternion, time, child, frame)

  usually time = rospy.Time.now()

- **How to listen (in Python):**

  tl = tf.TransformListener()

  (trans,rot) = tl.lookupTransform(child, frame, time)

  usually time = rospy.Time(0)

  which is the same as

  time = tl.getLatestCommonTime(child, frame)

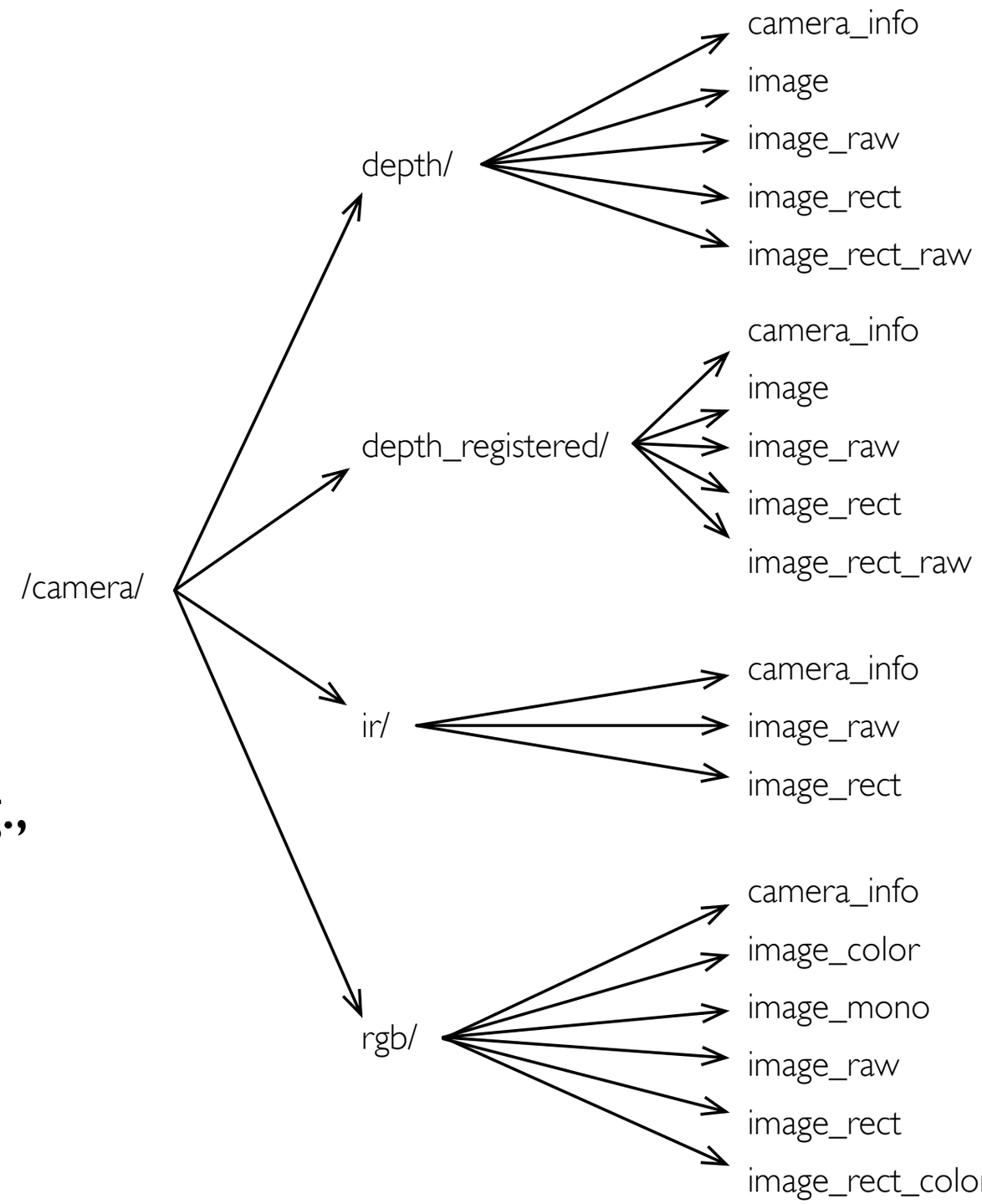
RGB-D camera drivers

- openni_camera: **drivers for various RGB-D cameras, based on OpenNI framework**
  - Microsoft Kinect
  - PrimeSense PSDK
  - ASUS Xtion Pro

- openni_launch: **launch files, namely** launch/openni.launch
  - manager
  - device drivers
  - processing nodelets

RGB-D camera drivers

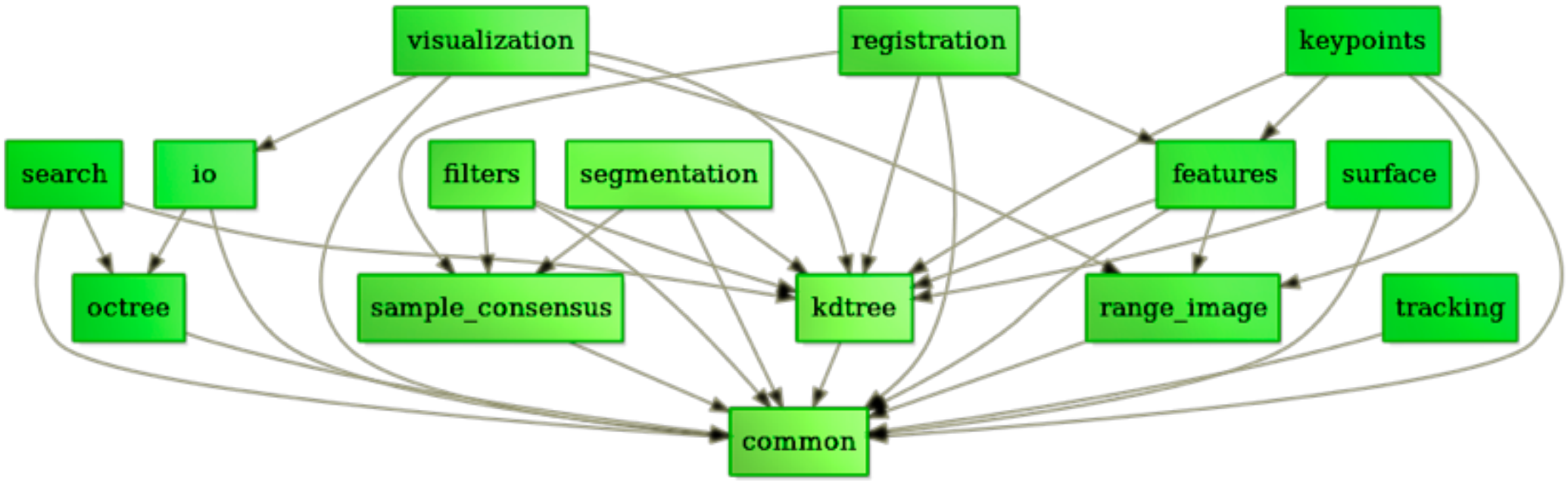
- Published topics (a selection):
  - **all images use** image_transport
  - camera_info **contains camera parameters (intrinsic, etc.)**
  - registered **means extrinsic parameter compensation**
  - rect **means rectification, e.g., distortion**
  - raw/**non-**raw **means unit conversion, e.g., integer vs millimeter depth**



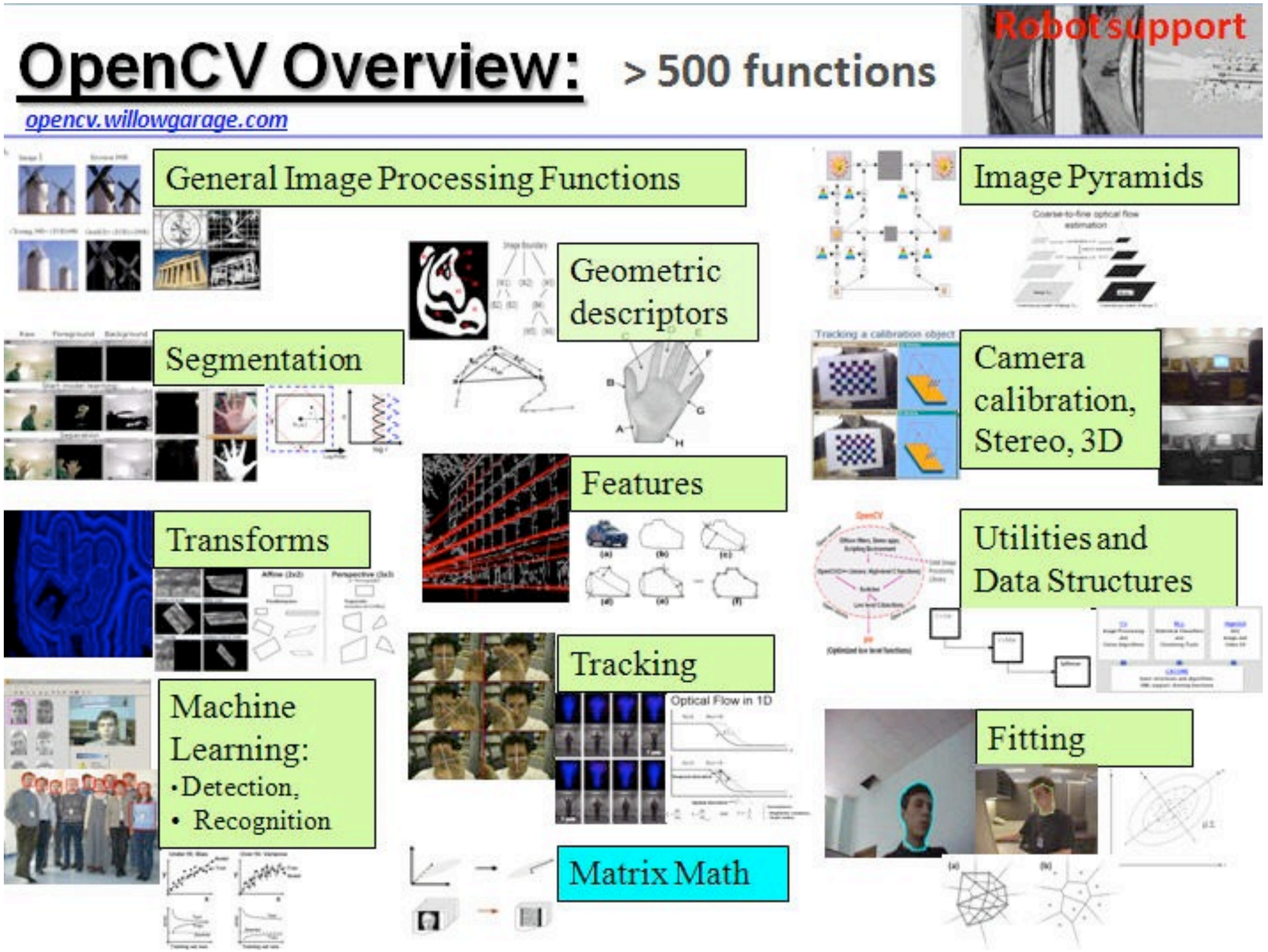
PCL

- Point Cloud Library (PCL)

  powerful library for pointcloud processing
- ROS package: pcl
- **website:** www.pointclouds.org
- PCL code libraries:



OpenCV


OpenCV Overview:
> 500 functions
opencv.willowgarage.com
Robot support
General Image Processing Functions
Image Pyramids
Geometric descriptors
Segmentation
Camera calibration, Stereo, 3D
Features
Transforms
Utilities and Data Structures
Tracking
Machine Learning:
•Detection,
• Recognition
Fitting
Matrix Math

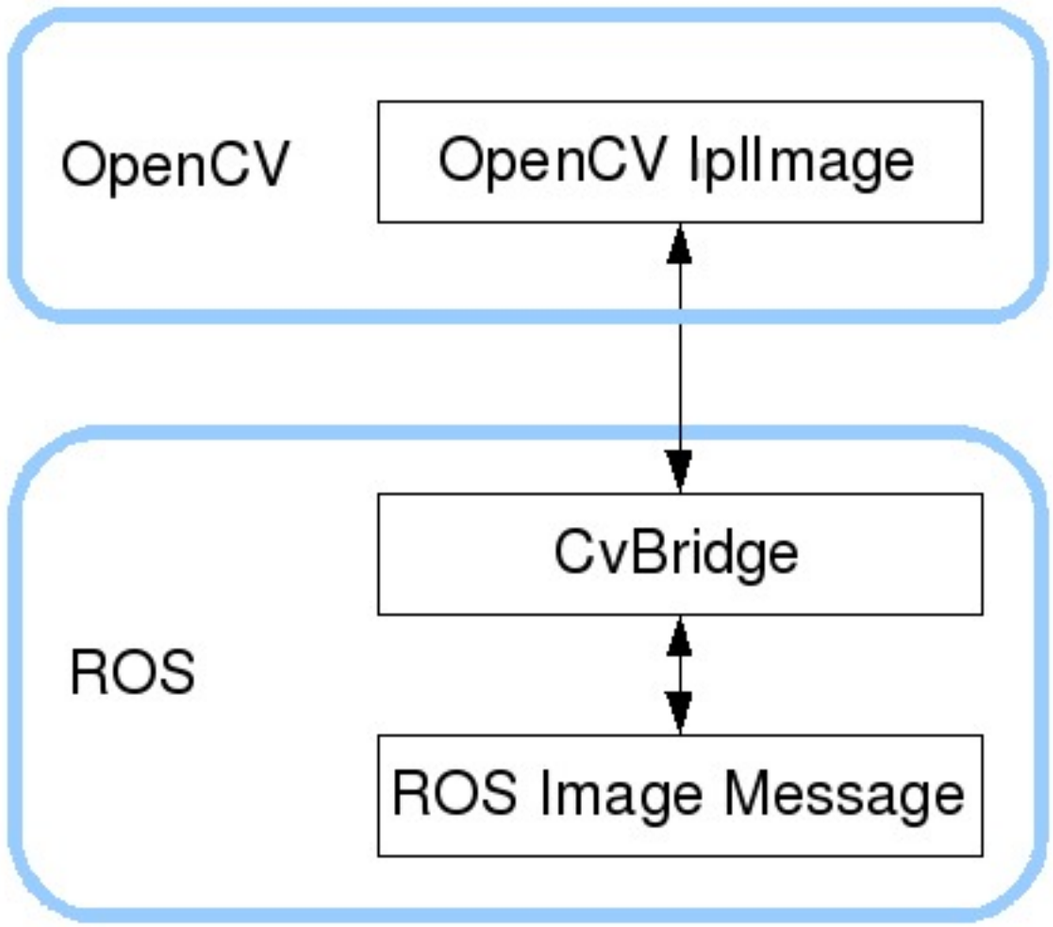
OpenCV

- Python package cv2 provides interface with OpenCV
- Example:


```
import cv2

feature_detector = cv2.SIFT()
image = cv2.imread("photo.jpg")

keypoints, descriptors = feature_detector.detectAndCompute(image, None)
print len(keypoints), "features detected"

print "First keypoint at %s:\n%s"%(keypoints[0].pt, descriptors[0])

cv2.namedWindow("image")
output = cv2.drawKeypoints(image, keypoints)
cv2.imshow("image", output)
cv2.waitKey()
```

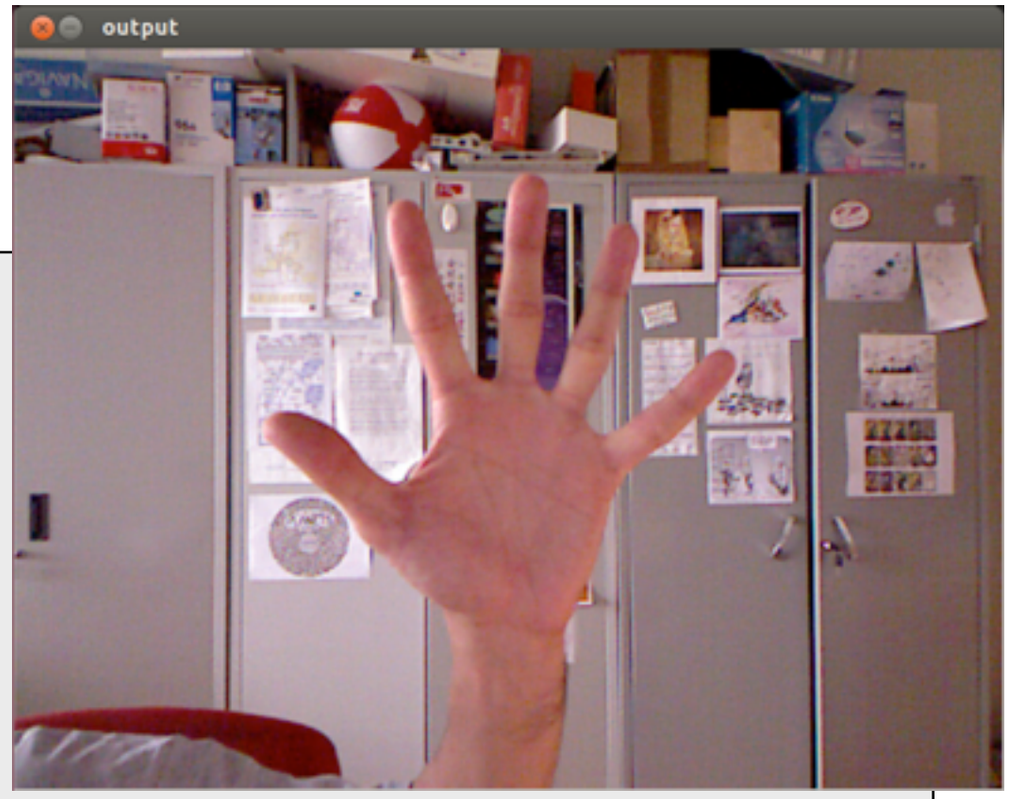
OpenCV

- **ROS package** cv_bridge **converts between OpenCV image representation and ROS message type** Image



OpenCV

- Example:




```
import rospy
import cv
import cv_bridge
from sensor_msgs.msg import *

bridge = cv_bridge.CvBridge()

def process(msg):
    image = bridge.imgmsg_to_cv(msg)
    cv.ShowImage("output", image)
    cv.WaitKey(10)

def main():
    cv.NamedWindow("output")
    sub = rospy.Subscriber("/camera/rgb/image_color", Image, process)
    rospy.init_node("testing")
    rospy.spin()

if __name__=="__main__":
    main()
```

GMapping

- **Package** gmapping **provides laser-based SLAM**
- **TF graph:**

  /map → /odom → /base_link → /laser_frame

- **How does it work:**
  - **robot odometry publishes** /odom → /base_link
  - gmapping **receives laser scans w.r.t. frame** /laser_frame
  - gmapping **then updates** /map → /odom

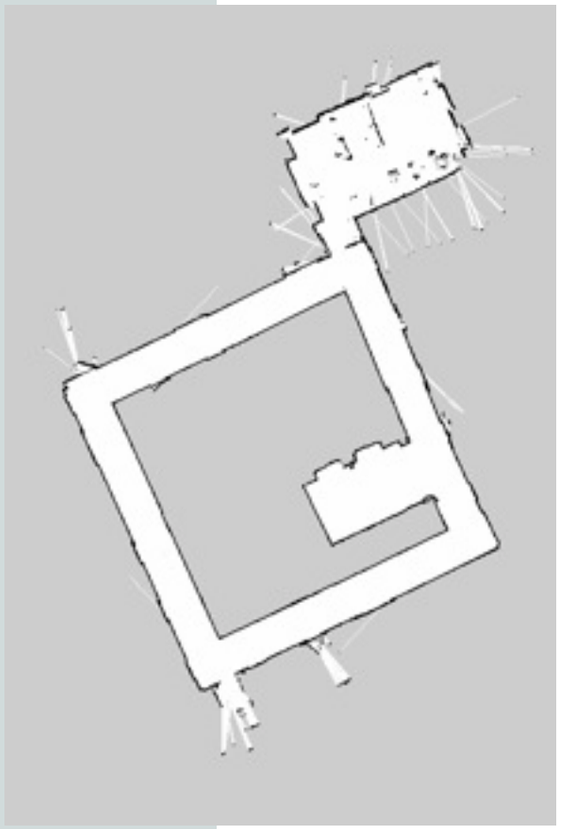
    such that /map → /base_link **is the updated localization**
- **The map is stored in a map server, provided by the package** map_server

GMapping

- Saving the map makes use of the map_server **package:**
- Example:

```
$ rosrun map_server map_saver -f mymap
```

  - two files are generated:

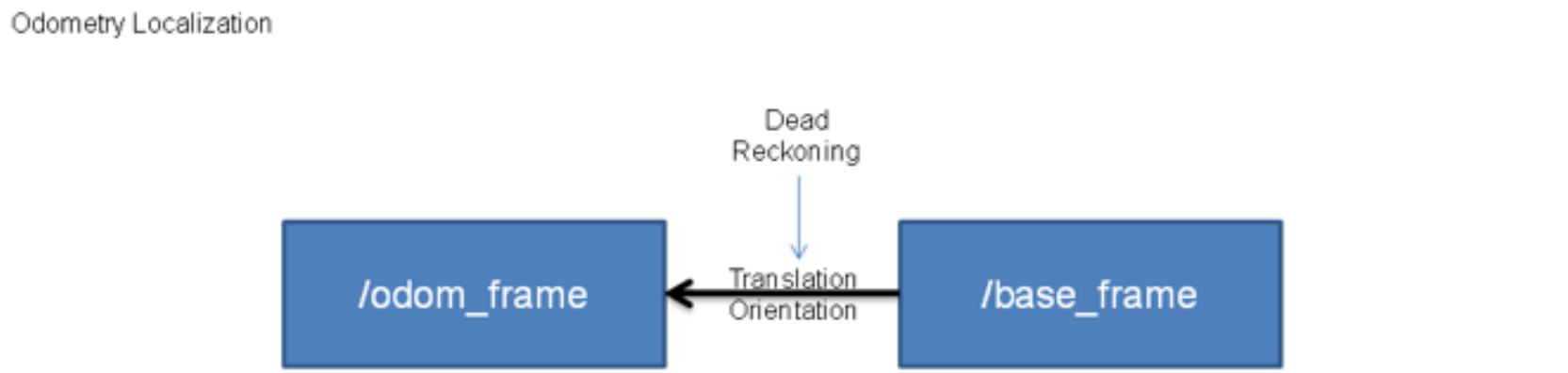


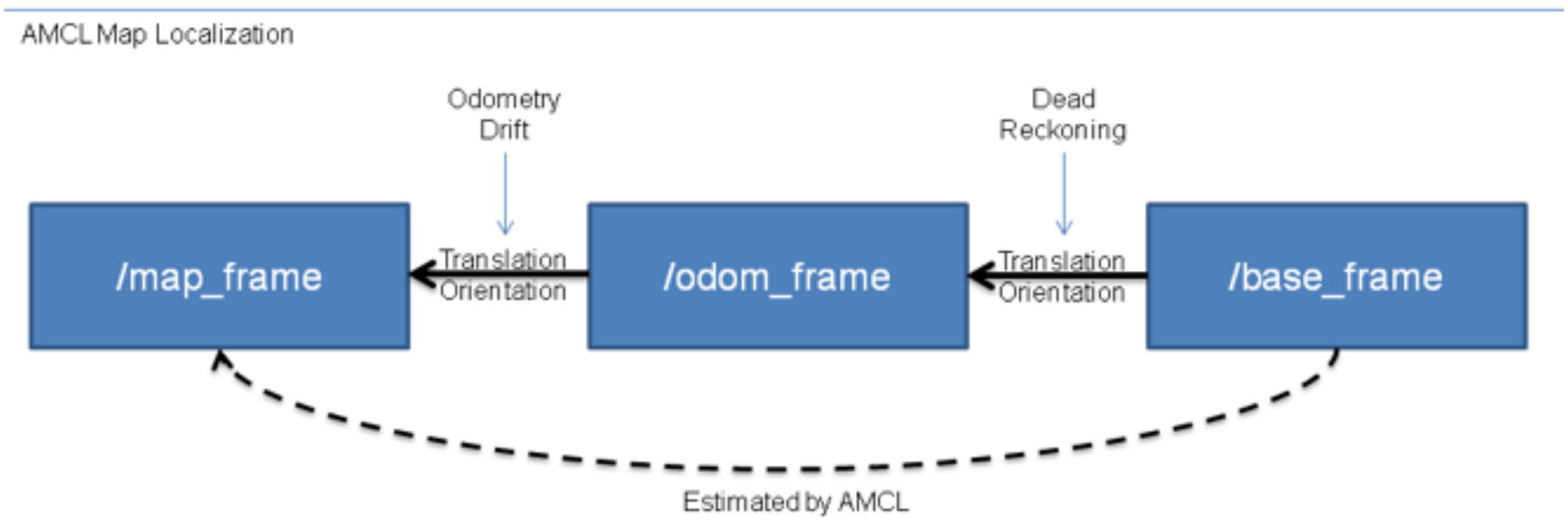
- mymap.pgm: occupancy grip map as a greyscale image (0-255)
- mymap.yaml: information about the map, e.g.:

```
image: mymap.pgm
resolution: 0.050000
origin: [-16.200000, -30.600000, 0.000000]
negate: 0
occupied_thresh: 0.65
free_thresh: 0.196
```

AMCL

- Package amcl provides Monte Carlo localization
- Identical TF graph than GMapping





Other packages

- hector_slam: **another state-of-the-art SLAM**
- octomap: **3D octree-based occupancy grid mapping**
- move_base: **navigation/guidance library**
- trajopt: **trajectory optimization for robot arms**
- openni_tracker: **skeleton estimation for RGB-D cams**
- maplab_rosbag: **package to open rosbags in MATLAB**
- vslam: **visual SLAM**
- **...**