

SAPIENZA
UNIVERSITÀ DI ROMA

DIPARTIMENTO DI INFORMATICA
E SISTEMISTICA ANTONIO RUBERTI

PRESEBUNG
AIS

Robot Sensors in ROS MARRtino

Giorgio Grisetti, Maurilio Di Cicco

Outline

- Robot Devices
 - Overview of Typical sensors and Actuators
 - Operating Devices in ROS
- Mobile Bases
- MARRtino
 - Hardware
 - Firmware
 - ROS interface
- Simulation: Stage
- Homework

Mobile Base

- A mobile platform is a device capable of moving in the environment and carrying a certain load (sensors and actuators)
- At low level the inputs are the desired velocities of the joints, and the output is the state of the joints
- At high level it can be controlled with linear/ angular velocity, and provides the relative position of the mobile base w.r.t. an initial instant, obtained by integrating the joint's states (odometry).



Proprioceptive Sensors for Ego-Motion

- Wheel encoders mounted on the wheels
- IMU:
 - Accelerometers
 - Gyros
- The estimate of ego-motion is obtained by **integrating** the sensor measurements of these devices. This results in an accumulated drift due to the noise affecting the measurement
- In absence of an external reference there is **no way** to recover from these errors



Exteroceptive Sensors

Perception of the environment

Active:

- Ultrasound
- Laser range finder
- Structured-light cameras
- Infrared

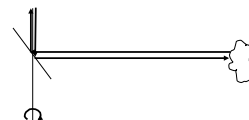
Time of flight

Passive:

- RGB Cameras
- Tactiles

Intensity-based

Laser Range Scanner



Properties

- High precision
- Wide field of view
- Approved security for collision detection

Robots Equipped with Laser Scanners



Zora:

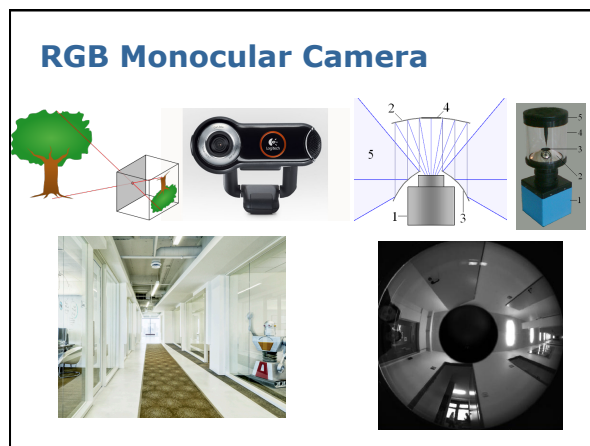
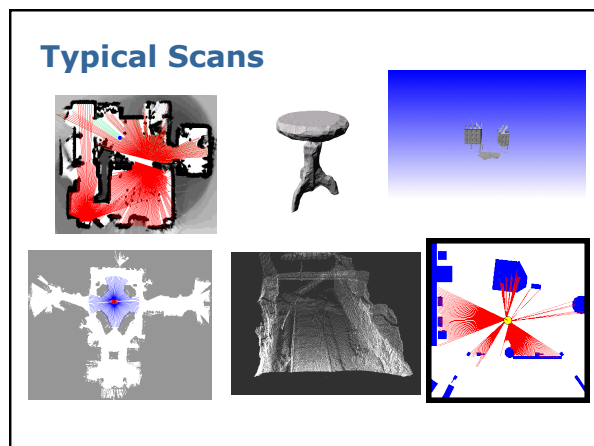


Groundhog:



Herbert:





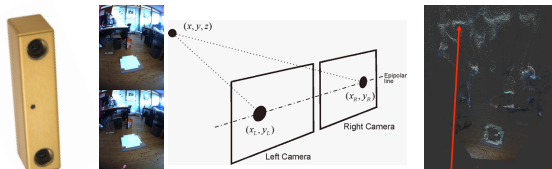
RGB Monocular Camera

- Cameras measure the intensity of the light projected onto a (typically planar) ccd through a system of lenses and/or mirrors
- Provide a lot of information
- Project 3D onto 2D, which results in the unobservability of the depth
- The scene can be reconstructed by multiple images (see SfM)

Stereo Camera

- Stereo cameras are combination of 2 monocular cameras that allow triangulation, given a known geometry.
- If the corresponding points in the images are known, we can reconstruct the 3D scene.
- Error in the depth depends on the distance!
- Sensible to lack of texture

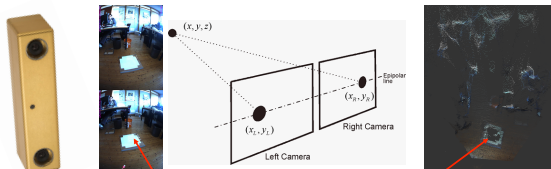
Stereo Camera



reconstruction from top

- Stereo cameras are combination of 2 monocular cameras that allow triangulation, given a known geometry.
- If the corresponding points in the images are known, we can reconstruct the 3D scene.
- Error in the depth depends on the distance!
- Sensible to lack of texture

Stereo Camera



reconstruction from top

- Stereo cameras are combination of 2 monocular cameras that allow triangulation, given a known geometry.
- If the corresponding points in the images are known, we can reconstruct the 3D scene.
- Error in the depth depends on the distance!
- Sensible to lack of texture

RGBD Cameras




- Cameras that are able to sense the color and the depth even with poor/no texture
- Use an active light source and retrieve the depth either
 - via stereo triangulation (emitter and source are in different positions)
 - Time of flight (emitter and source are in the same position)
- Environment conditions should allow to sense the emitted light.
- Typically OK indoors

How to access a Device in ROS?

- Each device is a node
- The input topics are the commands that the device can output
- The output topics are the feedback given by the device.
- In `sensor_msgs/` many messages for the common sensors are defined.
- Use `rosmmsg show <message_name>` to see the format of a message.
- To start a device it is sufficient to start the corresponding node and to give it the necessary configuration parameters. These include
 - Specific devices parameters (e.g. which serial port/usb device, the resolution of an image, and so on..)
 - The **name** of the **reference frame** in the sensor

Mobile Base in ROS

- Typical mobile bases are mapped as ROS nodes that

- Publish messages of type

- `nav_msgs/Odometry`

These messages specify the odometry

- Subscribes to messages of type

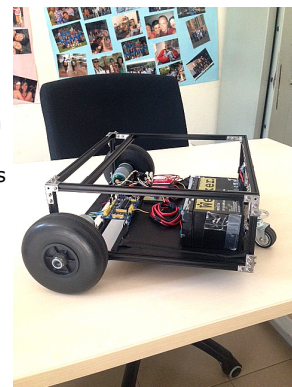
- `geometry_msgs/Twist`

That specify the desired translational and rotational velocities

All this looks very similar to TurtleSim, but the transforms and the velocities are computed in 3D

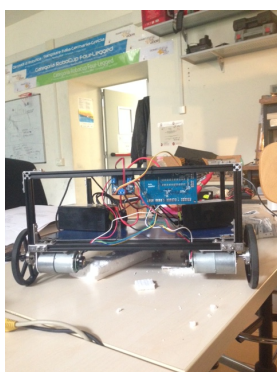
MARRtino

- Is a simple but complete mobile base designed to be used in the MARR course.
- The cost of the parts is around 300 euro
- It is entirely open source
- It is integrated in ros through a simple node that publishes/ subscribes standard topics



MARRtino

- Is a simple but complete mobile base designed to be used in the MARR course.
- The cost of the parts is around 300 euro
- It is entirely open source
- It is integrated in ros through a simple node that publishes/ subscribes standard topics



MARRtino: Frame

- The frame of MARRtino is realized with T-slot aluminium bars that can be assembled together by screws, bolts and junction elements, without the need of cutting or soldering

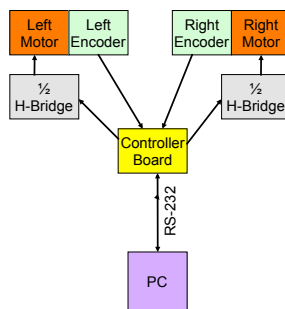


MARRtino: Frame

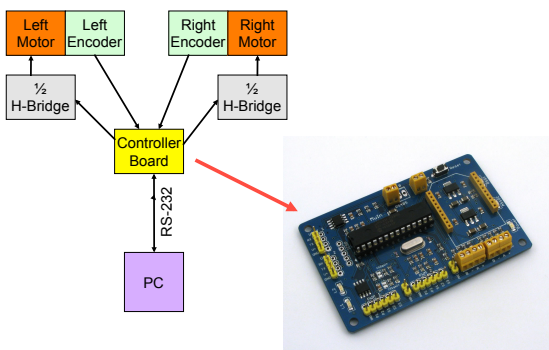
- The frame of MARRtino is realized with T-slot alluminium bars that can be assembled together by screws, bolts and junction elements, without the need of cutting or soldering



MARRtino: Electronics



MARRtino: Electronics



MARRtino: Electronics

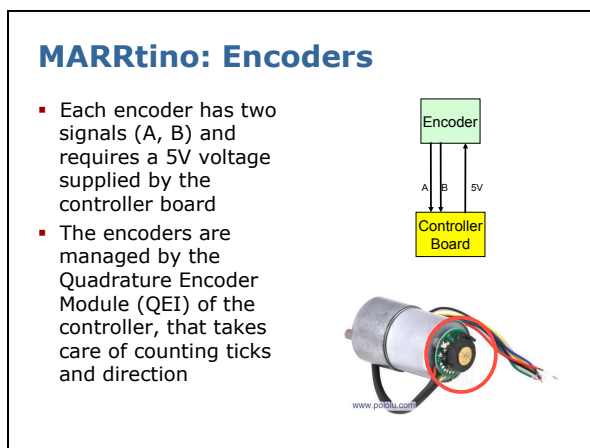
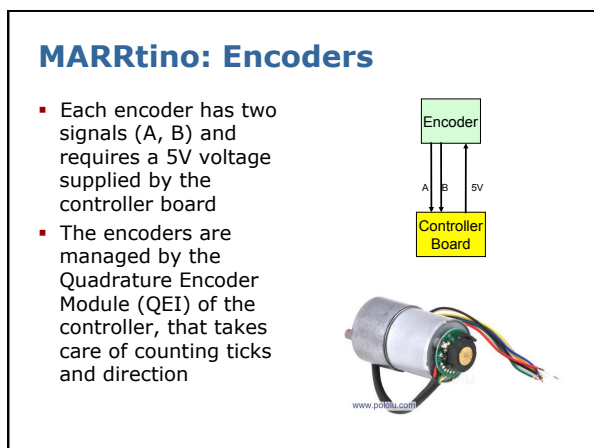
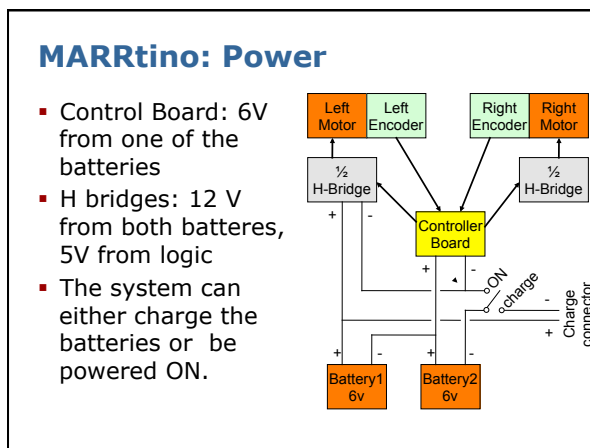
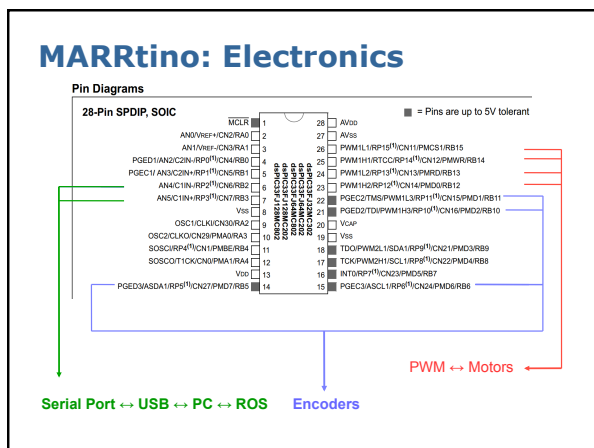
Pin Diagrams

28-Pin SPDIP, SOIC

| | | | |
|----|---|----|---|
| 1 | MCLR | 28 | VDD |
| 2 | AN0REF+CN2/R4 | 27 | AVSS |
| 3 | AN1REF+CN4/R4 | 26 | PWM1L1/RP19 ⁽¹⁾ CN11/PMCS1/RB15 |
| 4 | PGE1/AN2/CN2N1/RP9 ⁽¹⁾ CN4/RB0 | 25 | PWM1H1/RTCC/RP14 ⁽¹⁾ CN12/PMWR/RB14 |
| 5 | PGE1/AN3/CN2N1/RP9 ⁽¹⁾ CN5/RB1 | 24 | PWM1L2/RP19 ⁽¹⁾ CN13/PMRD/RB13 |
| 6 | AN4/C1N1/RP2 ⁽¹⁾ CN6/RB2 | 23 | PWM1H2/RP12 ⁽¹⁾ CN14/PMDO/RB12 |
| 7 | AN5/C1N1/RP2 ⁽¹⁾ CN7/RB3 | 22 | PGE2/TD3/PM1L3/RP11 ⁽¹⁾ CN15/PMO1/RB11 |
| 8 | VSS | 21 | PGE2/TD3/PM1H3/RP10 ⁽¹⁾ CN16/PMO2/RB10 |
| 9 | OSC1/CLKI/CN3/R4A2 | 20 | VDDP |
| 10 | OSC2/CLKO/CN28/PMAD/R3A | 19 | VSS |
| 11 | SSOSC/RP4 ⁽¹⁾ CN11/PMSE/RB4 | 18 | TCK/PMML1/SDA1/RP9 ⁽¹⁾ CN21/PMO3/RB9 |
| 12 | SSOSCD/TCK/CN0/PM1A1/R4A | 17 | TCK/PMMDH1/SCL1/RP9 ⁽¹⁾ CN22/PMD4/RB8 |
| 13 | VDD | 16 | NT0/RP7 ⁽¹⁾ CN23/PMO5/RB7 |
| 14 | PGE3/ASDA1/RP9 ⁽¹⁾ CN27/PMD7/RB5 | 15 | PGE3/ASCL1/RP8 ⁽¹⁾ CN24/PMD6/RB8 |

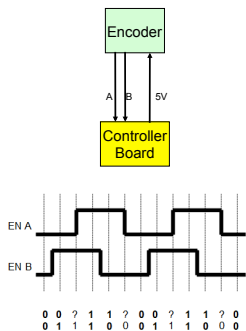
⁽¹⁾ = Pins are up to 5V tolerant

Up to 40 MIPS operation Program Memory (KB) 128
 Architecture 16-bit RAM Bytes 16,384



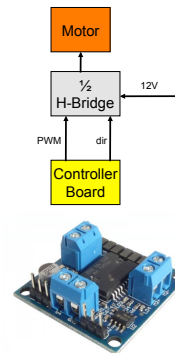
MARRtino: Encoders

- Each encoder has two signals (A, B) and requires a 5V voltage supplied by the controller board
- The encoders are managed by the Quadrature Encoder Module (QEI) of the controller, that takes care of counting ticks and direction



MARRtino: H Bridge

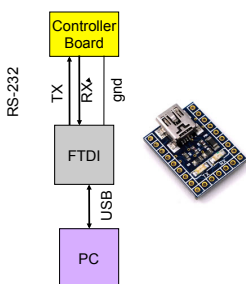
- The motor is connected to the H Bridge, that provides the necessary voltage and current to drive it.
- The H bridge requires 12V power directly from the battery
- The controller board controls the H bridge by*
 - A square wave whose duty cycle is proportional to the voltage applied to the motor, that controls the speed (PWM)
 - A direction pin, that reverts the voltage when asserted, causing the motor to rotate in the opposite direction



* Other modes are possible, please refer to the wiki for details

MARRtino: PC connection

- The robot communicates with the PC through an RS232 interface at TTL levels (0-5V)
- The TTL-RS232 is converted in USB through an FTDI chip
- The device is visible on Linux as /dev/ttyUSB0



MARRtino: Firmware

- The controller runs an event driven C program that
 - Executes PID controllers on both wheels
 - Integrates the odometry
 - Communicates with the PC
 - Implements a watchdog
- The PC periodically sends to the controller the desired translational/rotational velocities
- The controller periodically sends to the PC state packets that include the integrated odometry since the last message and the battery state
- If the controller does not receive any message from the PC for a while it sets the speed to 0 (safety measure)

MARRTino: Wiki

- The wiki describes all the steps to:
 - obtain the firmware code
 - flash the firmware on the board
 - Obtain the ROS node
 - Use the ROS node

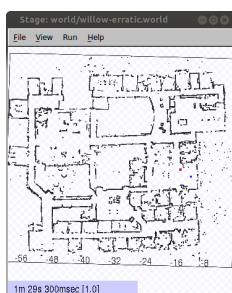
- www.dis.uniroma1.it/~spqr/MARRtino

Simulation: Stage

- There are many simulators integrated in ROS, but in this lecture we will focus on Stage.
- Stage is a 2D simulator that allows to operate a large number of simple robots and simulates their sensors
- To work, stage requires a configuration file (*.world*), that describes the configuration and the position of each robot.
- The *.world* includes a bitmap, that represents the environment where the robot is operating

Stage

- To launch stage
 - `$> rosrun`
 - `$> roscd stage`
 - `$> rosrunc stage stageros words/willow_erratic.world`
- With `rostopic` you will see that there is a `/cmd_vel` argument. Publishing on this topic allows you to set the the robot speed
- The robot sends you the odometry feedback by the `/odom` topic, and potentially some additional state packet.



Homework

- Do a simple program that listens to the keyboard or the joystick and allows to control the robot by sending `/cmd_vel` messages
- First try it in simulation (needed to pass the homework)
- Start the viewer (`rviz`), and try to visualize the laser scan
- Once this is running, test it on MARRtino.