LEONARDO

LEONARDO TECHNICAL TRAINING

Robotic Software
Lezione 1

Introduzione programmazione sistemi robotici e principali tool

Agenda

- Organizzazione delle lezioni
- Robotica e sistemi automatici
- Programmazione di sistemi robotici avanzati
  - Robotica Industriale
  - Robotica Mobile
- Principali problemi aperti
- Strumenti utilizzati
- Percorso di apprendimento
- Programmazione di sistemi robotici intelligenti

Organizzazione delle lezioni

- Durata lezione: circa 4 ore
- Ogni lezione consiste di una parte teorica e una parte pratica
- Teoria:
  - Descrizione degli elementi utilizzati nella parte pratica
- Pratica:
  - Analisi e studio di esempi pratici
  - Esercitazioni da svolgere in aula
- Materiale didattico:
  - Esempi e esercitazioni su github: https://github.com/robotic-software
  - Slide presentate a lezione
  - Riferimenti per ulteriori approfondimenti

Organizzazione delle lezioni

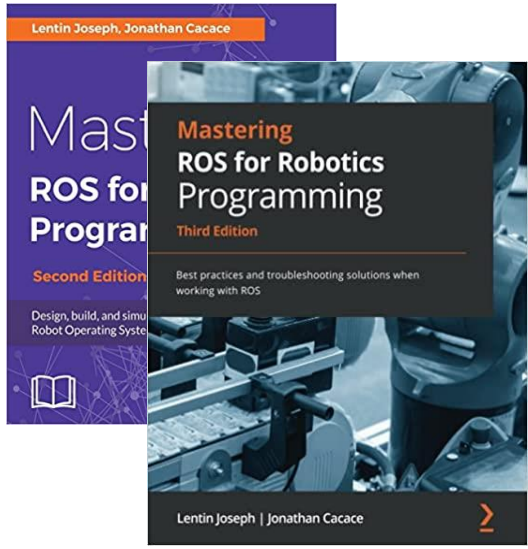
- Strumenti hardware:
  - Computer Host
    - Windows (necessario per utilizzare la piattaforma di training di Webex)
    - Utilizzato per avviare le applicazioni e modificare codice in maniera remota
  - Computer Remoto
    - Amazon Web Service (AWS)
      - Computer con sistema opertivo Linux Ubuntu 18.04 / 22.04 installato
      - Equipaggiate con schede video NVIDIA in modo da sfruttare il caclolo in GPU
  - Software di comunicazione
    - SSH: Secure Shell
    - RDP: Remote Desktop Protocol
- Strumenti software
  - Docker
  - Visual studio code
- Tutto il necessario sarà installato durante il corso

Organizzazione delle lezioni

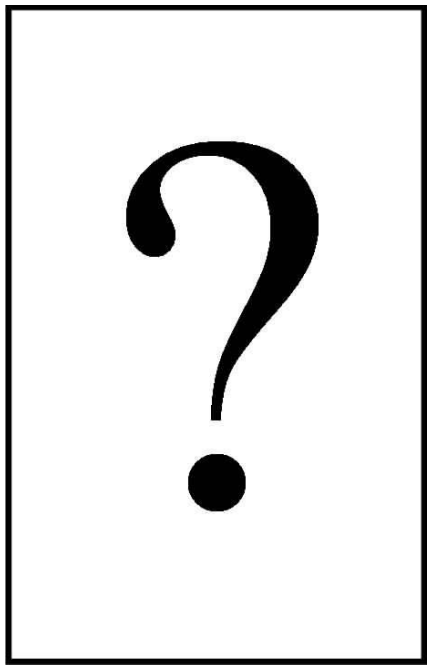
- Strumenti di programmazione:
  - Cpp
    - Tipicamente Cpp è utilizzato per la velocità di esecuzione
    - Nei sistemi robotici è importante reagire prontamente agli stimoli esterni
    - Applicazioni multi-threading
  - Python
    - La semplicità di utilizzo semplifica la programmazione di comportametni avanzati
  - Git
    - Deploy e aggiornamento del codice sorgente
  - Json e XML
    - Configurazione

Presentazioni

- Jonathan Cacace, PhD
  - Laurea triennale e magistrale in informatica
  - Dottorato di ricerca in ingegneria informatica e automatica
- Posizione lavorativa
  - Ricercatore a tempo determinato (RTD-A) presso l'università degli studi di Napoli Federico II
    - Ingegneria dell'automazione
      - Robotics Lab
    - Autonomous Vehicle Engineering
      - Mobile robots
  - Attività di ricerca su navigazione di robot aerei in interazione con l'ambiente
  - Attività di ricerca su robot manipolatori in interazione con l'ambiente

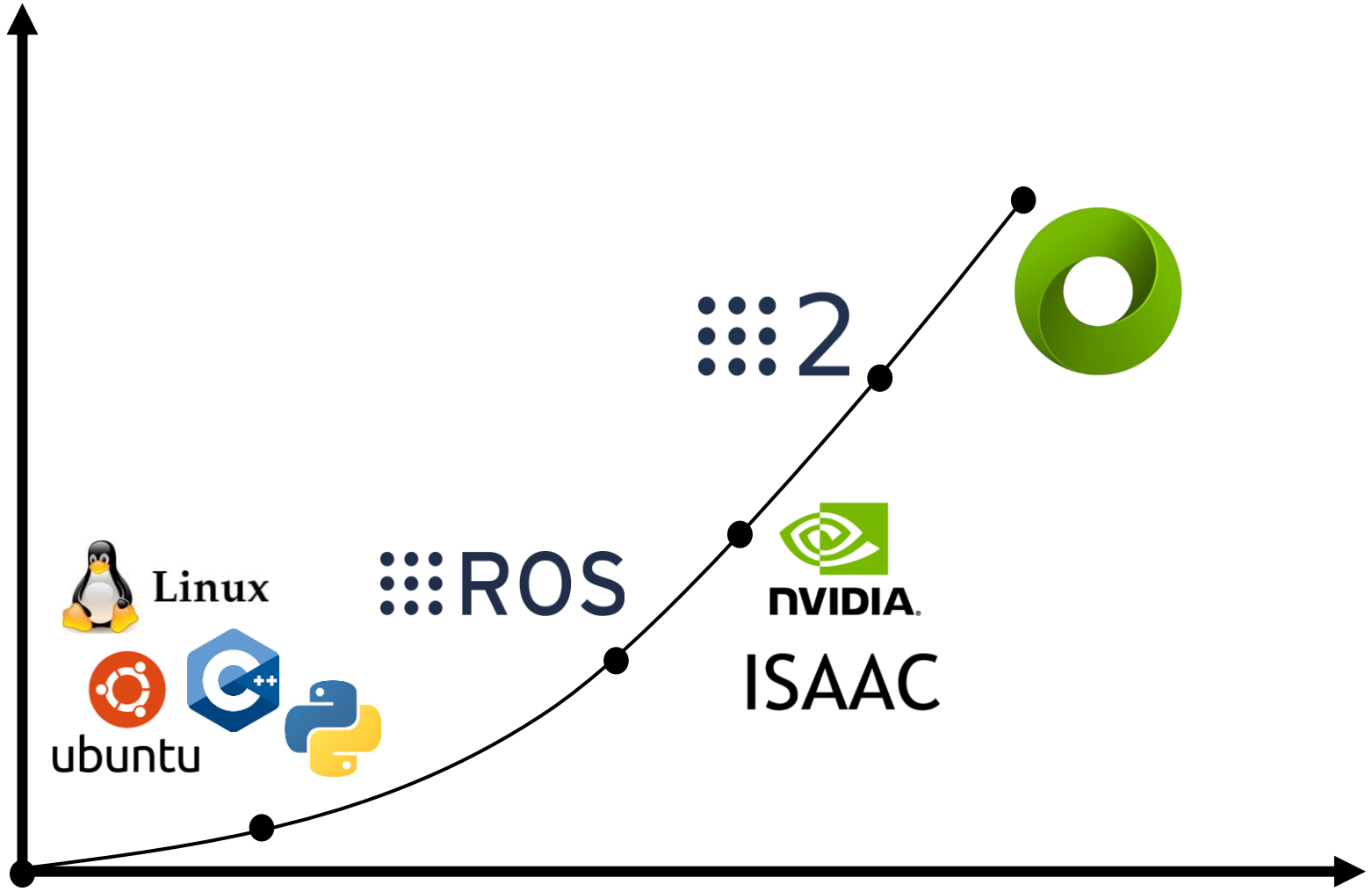


Presentazioni



Percorso di apprendimento

- Basic tools
  - Linux
    - Ubuntu
  - C++
  - Python
  - Git
- ROS
- NVIDIA Isaac SDK
- ROS + NVIDIA SDK
- ROS2
- NVIDIA SIM
- ROS + ROS2 + NVIDIA SIM



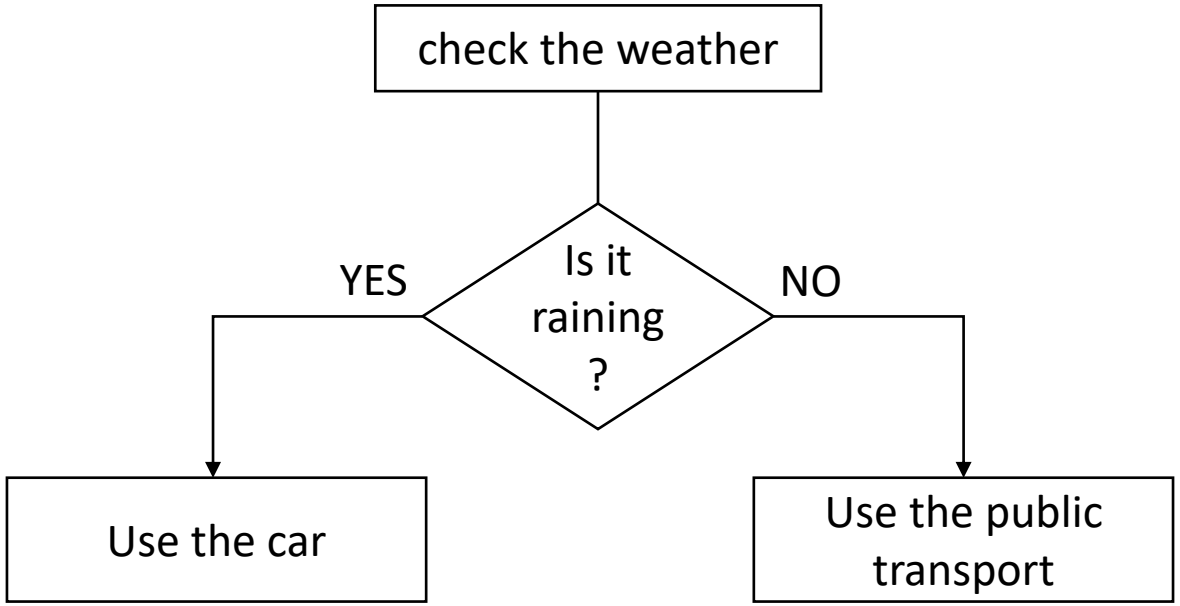
Percorso di apprendimento

Numero lezione	Argomento
1 – 26/09	Introduzione al corso e alla programmazione di sistemi robotici
2 – 27/09	ROS
3 – 04/10	ROS
4 – 06/10	ROS / ISAAC SDK
5 – 11/10	ISAAC SDK
6 – 13/10	ISAAC SDK + ROS / ROS2
7 – 18/10	ROS2
8 – 20/10	ROS2 / ISAAC SIM
9 – 25/10	ISAAC SIM
10 – 27/10	ISAAC SIM + ROS2

Introduzione alla programmazione di sistemi robotici

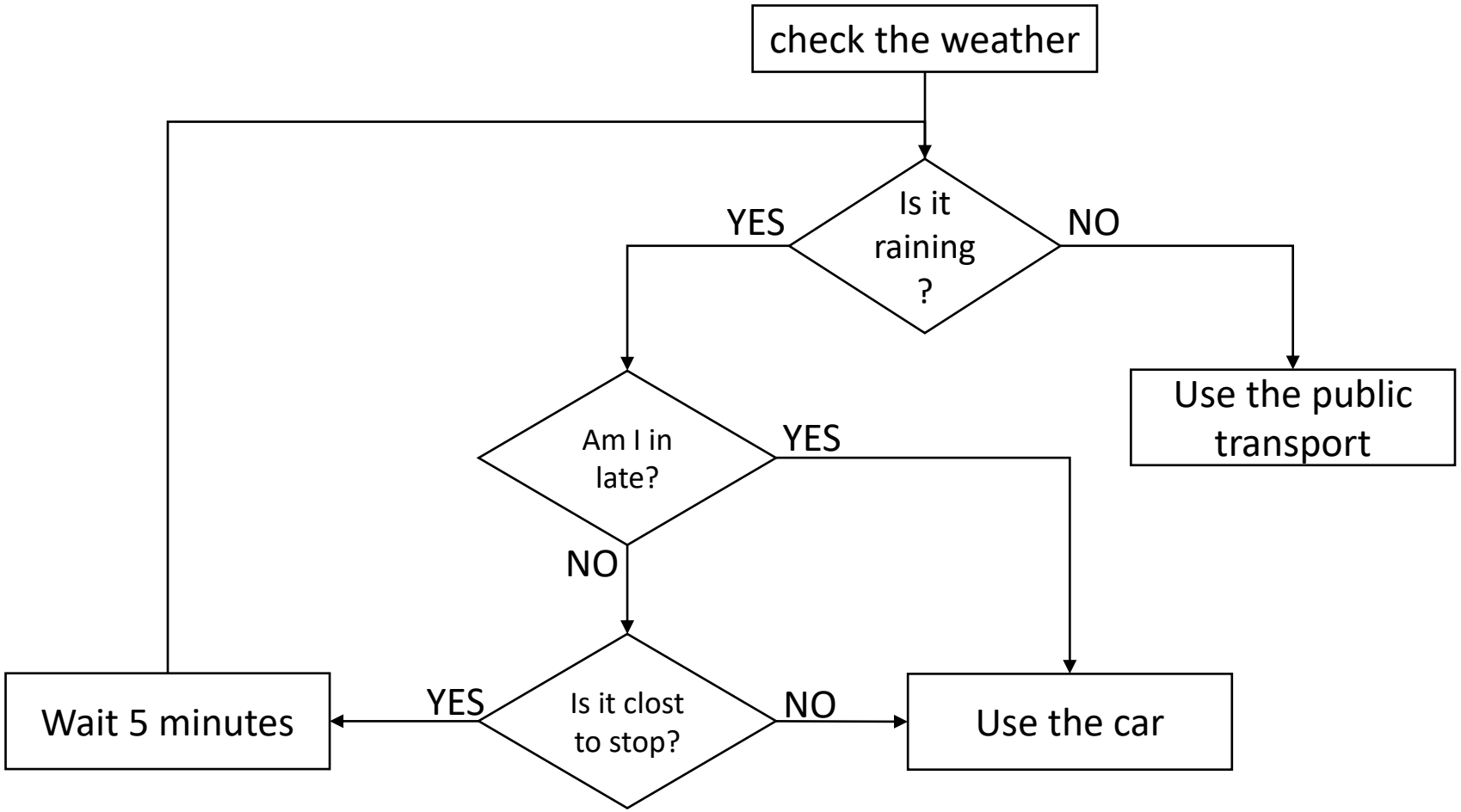
Paradigma: Sense-Plan-Act

- Un esempio simile alla vita reale
  - Un uomo che deve raggiungere il suo posto di lavoro.
    - Lui preferisce raggiungere il posto di lavoro utilizzando il trasporto pubblico, a meno che non piova.



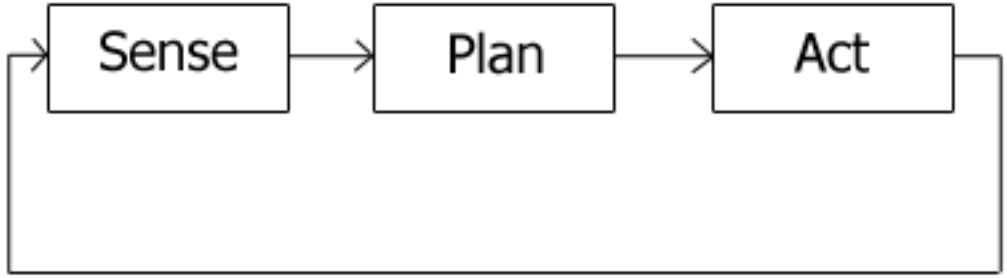
Paradigma: Sense-Plan-Act

- Un esempio simile alla vita reale
  - Un uomo che deve raggiungere il suo posto di lavoro.
    - Lui preferisce raggiungere il posto di lavoro utilizzando il trasporto pubblico, a meno che non piova.



Paradigma: Sense-Plan-Act

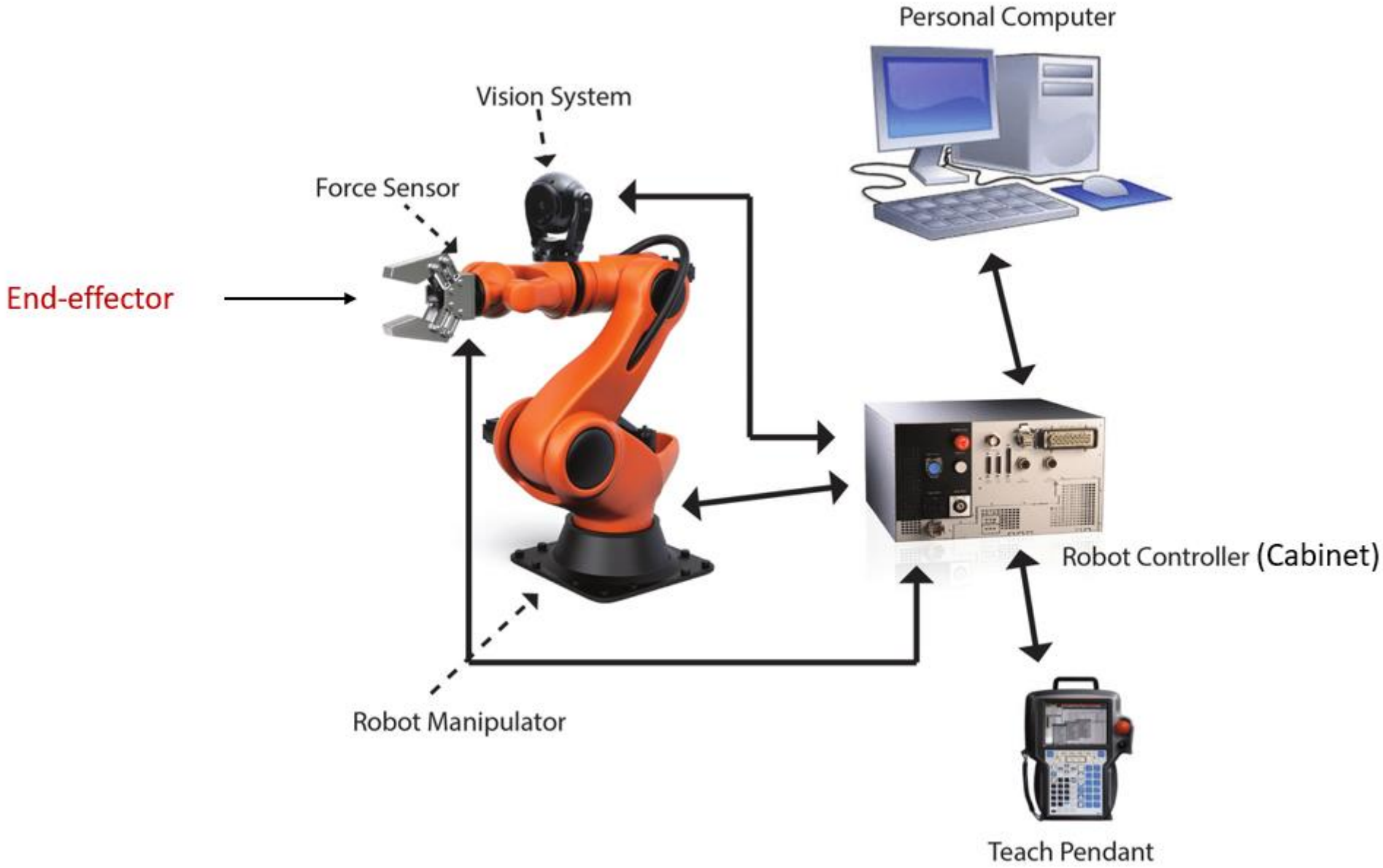
- Un esempio simile alla vita reale
  - Un uomo che deve raggiungere il suo posto di lavoro.
    - Lui preferisce raggiungere il posto di lavoro utilizzando il trasporto pubblico, a meno che non piova.
  - Questo esempio è molto simile a un caso di programmazione di un sistema robotico
- Programmare un sistema robotico avanzato vuol dire implementare il paradigma: SENSE-PLAN-ACT
  - Questo paradigma è utilizzato iterativamente:
    - Dopo la fase di azione, di pianificazione e di azione, l'intero ciclo è ripetuto
  - SENSE
    - Il robot deve avere l'abilità di percepire l'ambiente e i suoi cambiamenti
  - PLAN
    - Il robot deve avere la possibilità di decidere le proprie azioni, in base ai dati dei sensori
  - ACT
    - Il robot deve essere in grado di eseguire le azioni pianificate



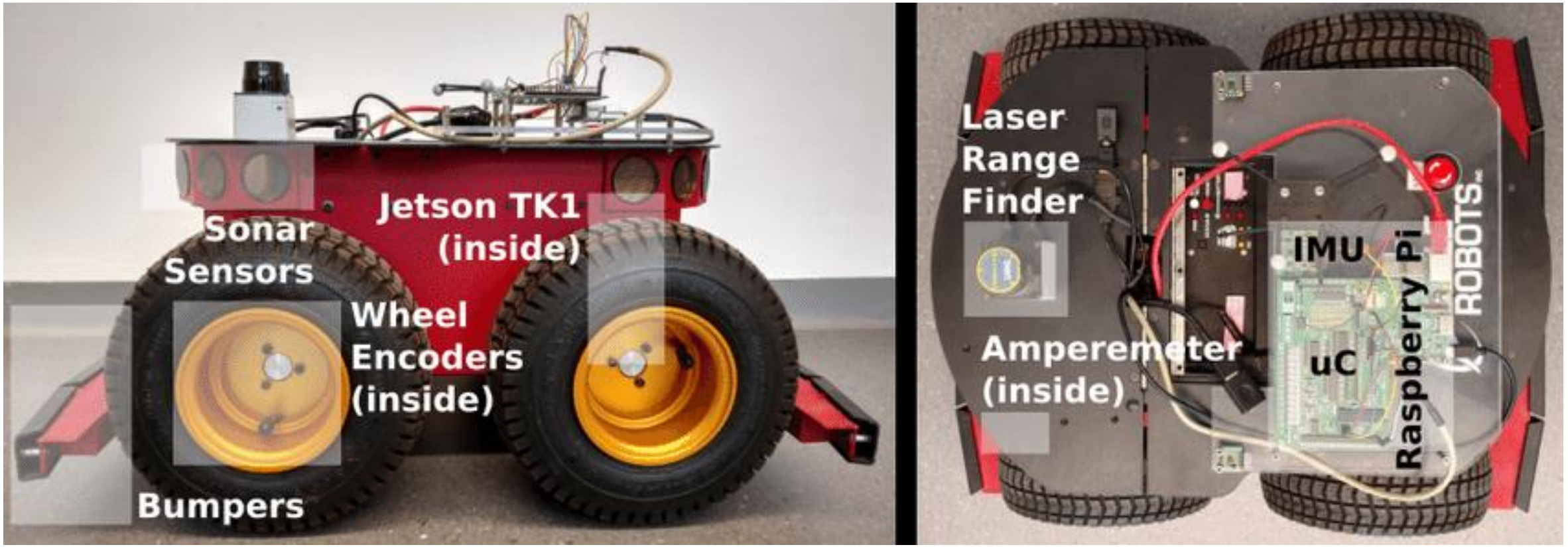
Classificazione

- La Robotics Institute of America classifica i robot nelle seguenti categorie
- Variable-sequence robot: un dispositivo che esegue in maniera progressiva una serie di compiti predeterminati. Questi compiti possono modificati prima della loro esecuzione
  - Principalmente utilizzati nella robotica industrial
  - Variable-sequence: macchinari che possono essere programmati
- Robot avanzati intelligenti: un robot che è in grado di interpretare l'ambiente il proprio stato che abbiano l'abilità di completare i propri compiti anche se questo ambiente cambia
- Un ulteriore classificazione di sistemi robotici può essere fatta in:
  - Robot industriali: manipolatori
  - Robot mobili: sistemi in grado di navigare l'ambiente
    - Si possono ancora classificare in base alla loro locomozione
      - Terrestri
      - Aerei
      - Legged

Hardware robot industriali



Hardware robot mobili



Cosa è un robot?

- Un sistema robotico è un apparato meccanico/elettronico/informatico in grado di svolgere un compito ben preciso in autonomia
- Elementi di un sistema Robotico
  - Ambiente di lavoro
  - Struttura meccanica
  - Sensori
  - Unità di elaborazione
  - Attuatori
- Lavatrice... È un robot?
  - Ha un obiettivo ben preciso:
    - lavare i panni!
  - Lo fa *da sola*
  - Opera in un *ambiente di lavoro*
  - Ha una struttura meccanica
  - Possiede opportuni sensori
    - (temperatura, peso, etc.)
  - Possiede una unità di elaborazione che comanda il ciclo di lavaggio
  - Possiede degli attuatori (motore cestello, pompa, resistenza di riscaldamento, etc.)

Cosa è un robot?

- Un sistema robotico è un apparato meccanico/elettronico/informatico in grado di svolgere un compito ben preciso in autonomia
- Ha un obiettivo ben preciso:
  - togliere la polvere dal pavimento della stanza!
- Lo fa *da solo*
- Opera in un *ambiente di lavoro* (la casa)
- Ha una struttura meccanica
- Possiede opportuni sensori (posizione, ostacoli, scalini, sporcizia, etc.)
- Possiede una unità di elaborazione che comanda le operazioni legate alla gestione dei percorsi ed all'attivazione della pompa
- Possiede degli attuatori (motori locomozione, aspiratore, etc.)

Cosa è un robot?

- Il veicolo autonomo costituisce la prossima sfida nell'ambito dei sistemi di trasporto
- E' un sistema robotico
- Ha un obiettivo: portarvi sani e salvi verso una destinazione
- Ha dei sensori: GPS, velocità, ostacoli, ambiente, etc.
- Ha un'unità di elaborazione, che stabilisce come pilotare la guida sulla base del percorso voluto e le informazioni dai sensori
- Ha degli attuatori: motore, sterzo, freni, etc.

Ambiente di lavoro (ambiente fisico)

- Sistemi non-robotici
- L'ambiente dove opera il software è genericamente virtuale, sottoposto al nostro controllo
- Le chiamate a funzione
  - ...funzionano sempre
  - ... restituiscono un'indicazione di successo/fallimento
  - Fallimento: risultato immediatamente, rispetto all'eventuale esecuzione dell'azione (es. invio di un pacchetto in rete)
  - La probabilità di fallimenti è (in genere) molto bassa e
    - Legata a eventi eccezionali
    - L'azione è persistente

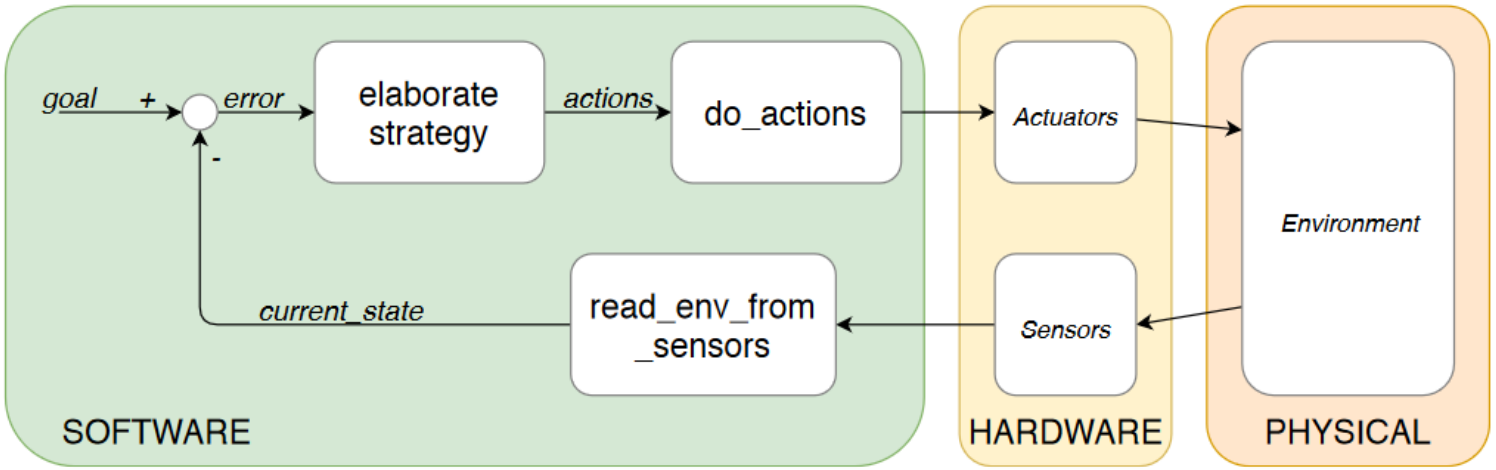
Ambiente fisico

- Sistemi robotici
- L'ambiente dove opera il software è fisico ed evolve secondo sue proprie regole che non siamo in grado di controllare
- Una chiamata di funzione che agisce sull'ambiente:
  - Può fallire anche con probabilità elevata
  - L'indicazione di fallimento può non essere immediata
  - L'azione (o la sua conseguenza) non è garantito sia persistente

  - Nei sistemi fisici, effettuare un'azione non implica la sua persistenza!
    - Esempio: percorrere un rettilineo in auto richiede che lo sterzo sia dritto
    - Impostare lo sterzo e non toccarlo più, è sufficiente per assicurare un percorso corretto?
      - No! Noi verifichiamo continuamente la nostra "consegna" (percorso rettilineo) e "aggiustiamo" (controlliamo) lo sterzo sulla base di errori

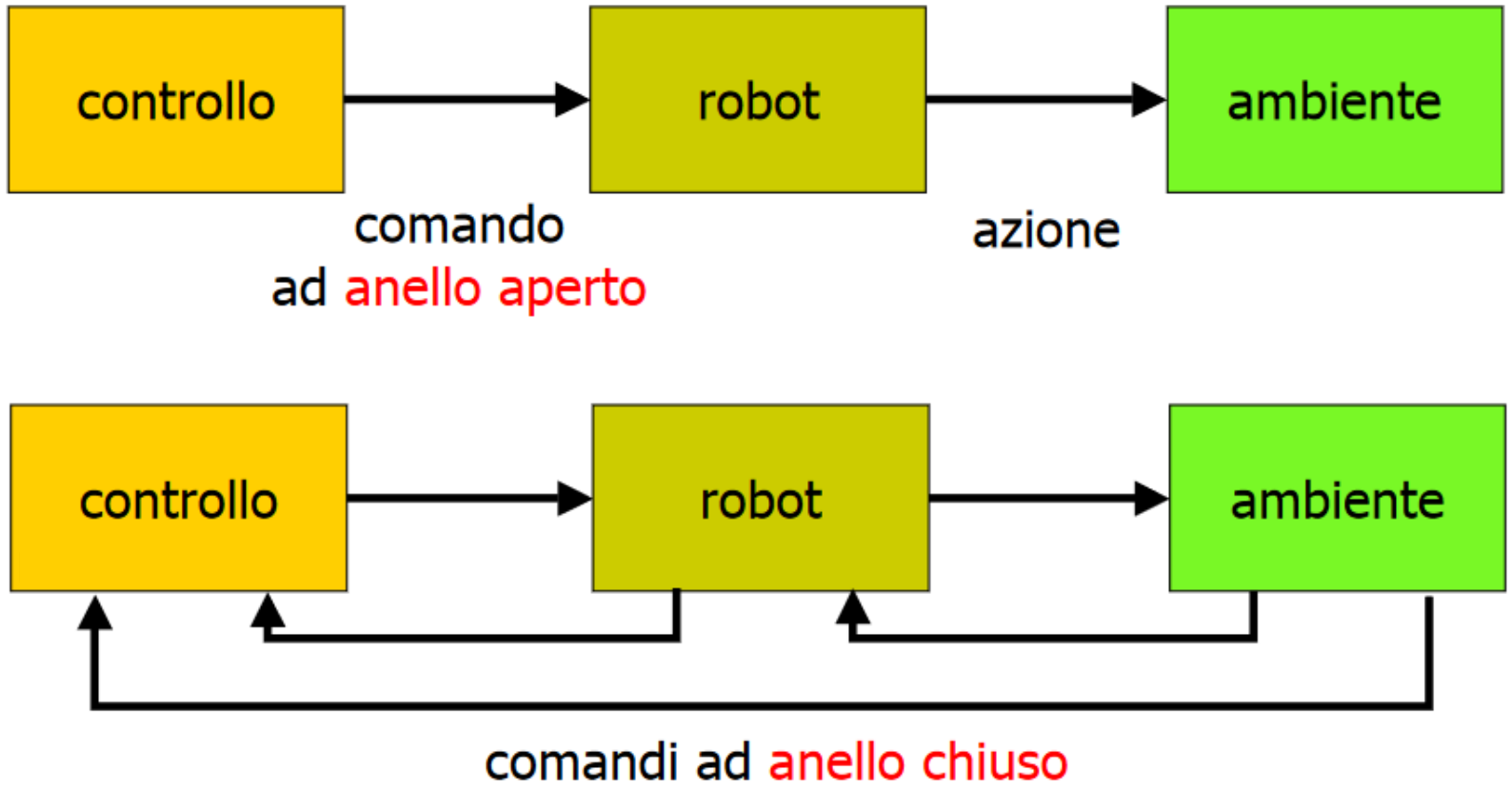
Controllo di robot

- Sistemi robotici
- L'ambiente dove opera il software è fisico ed evolve secondo sue proprie regole che non siamo in grado di controllare
- Feedback
  - Sense-Plan-Act
  - Data una grandezza fisica da controllare (velocità, posizione, forza, corrente, ..., distanza)
  - Avendo a disposizione un sensore in grado di misurare tale grandezza, anche con un certo grado di errore
    - Si acquisisce il dato sulla grandezza effettiva tramite il sensore
    - Confronta il valore effettivo con quello desiderato
    - Sulla base di questo errore, il sistema di controllo attua i motori di conseguenza



Controllo di robot

- L’azionamento di un robot è caratterizzato alcuni loop di controllo
  - Loop di velocità
  - Loop di posizione
  - Loop di coppia



Controllo di robot: attuatori

- Tipi di controllo
  - Dipende anche dal tipo di attuazione
  - Controllo in coppia
    - Si specifica direttamente la coppia da imprimere all'attuatore
    - Tale coppia dipende da diverse cose:
      - Quanta forza è necessaria per vincere l'inerzia del motore più la parte meccanica ad essa attaccata?
      - Quanda forza è necessaria per vincere la gravità
    - Utilizzato quando si vuole specificare direttmante una coppia/forza deisderata per compiere un compito
  - Controllo in velocità
    - Si specifica la velocità itpicamente in RPM (rotazioni per minuto) da far eseguire al motore
  - Controllo in posizione
    - Si specifica la posizione che si vuole raggiungere con il motore

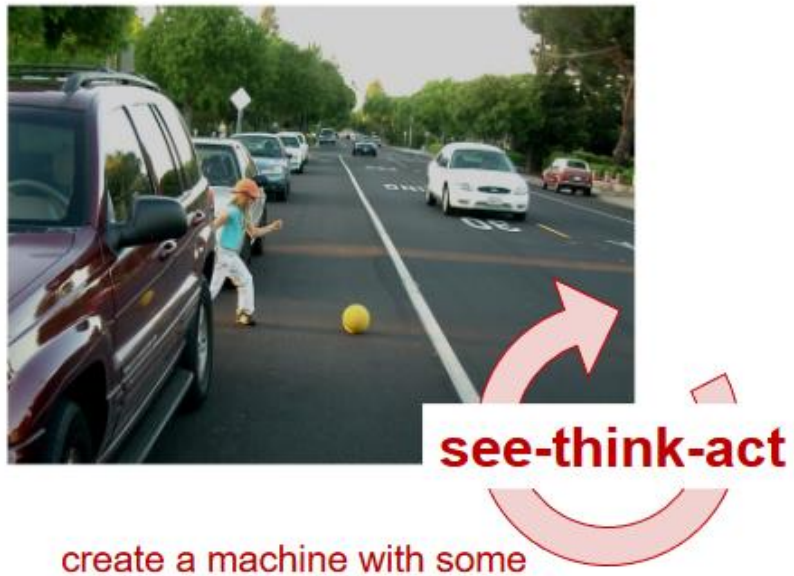
Controllo di robot: attuatori

- Non tutti gli attuatori permettono il controllo in forza
- Tipi di controllo
  - Smerigliatura: controllo in forza
    - Anche se gli attuatori non permette il controllo in coppia, si può utilizzare un sensore di forza e coppie
  - La frequenza di controllo cambia a seconda del loop di controllo
    - Il motore sarà sempre controllato in coppia anche se noi possiamo specificare la posizione e la velocità
  - Un controllore elettrico dedicato tradurrà l'input in posizione in output di coppia

Controllo di robot: sensori

- Elaborare i dati generati dai sensori in robotica è parte del problema della percezione
- Questi problemi sono davver difficili da risolvere, sopratutto quando si lavora in ambienti non controllati (non strutturati)
- In robotica così come nei problemi di intelligenza artificiale, problem semplice da risolvere per noi umani diventano molto difficili, e viceversa

beating the world's chess master: EASY

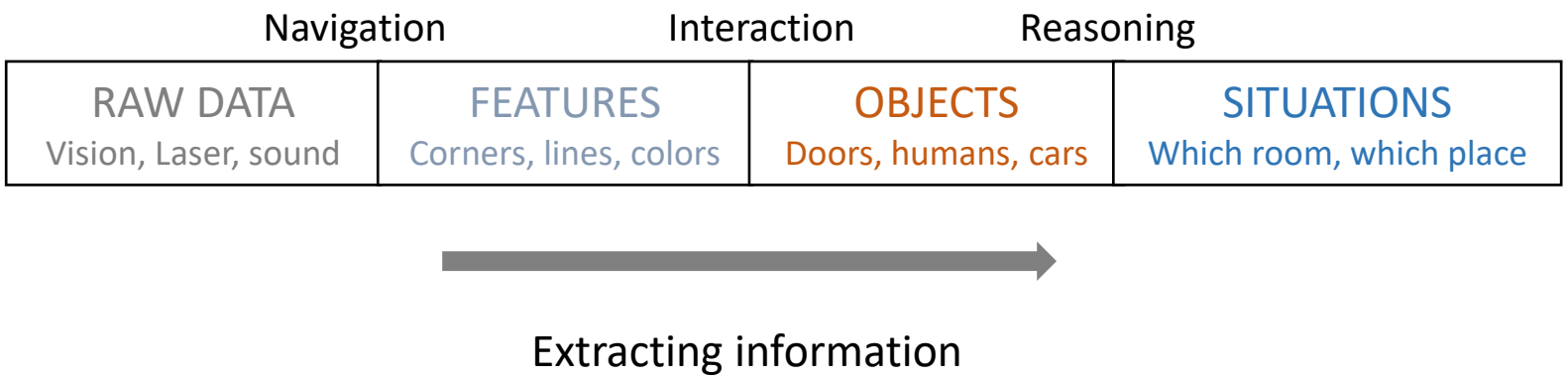


create a machine with some "common sense": very HARD

Controllo di robot: sensori

Controllo di robot: sensori

- Perception pipeline
  - Problemi di navigazione: permettono al robot di navigare liberamente l'ambiente
  - Problemi di interazione: permettono al robot di interagire con l'ambiente
  - Problemi di ragionamento: permettono al robot di capire l'ambiente



Controllo di robot: sensori

- Sensori propriocettivi: misurano dati interni al robot
  - Velocità dei motori
  - Carico di peso su una ruota
  - Posizione angolare di un giunto angolare
  - Carica della batteria
- Sensori esterocettivi: acquisiscono informazioni sull'ambiente in cui opera il robot
  - Distanza dagli ostcaoli
  - Intensità della luce
- Sensori passive: misurano i dati senza emettere alcun tipo di energia nell'ambiente
  - Temperature probes
  - Microphones
  - CCD or CMOS cameras.
- Sensori attivi: emettono energia nell'ambiente per misurare la sua reazione
  - Ultrasonic sensors
  - Laser rangefinders

Controllo di robot: sensori

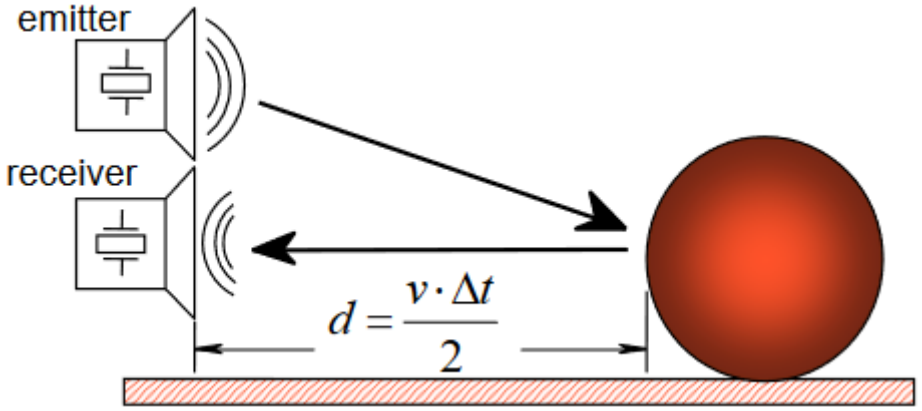
- Esempi di sensori
  - Sensori tattili o bumbper: sono in grado di identificate un contatto fisico
  - GPS: sistemi di posizionamento globale
    - Utilizzati principalmente per la navigazione outdoor
  - IMU: sensori inerziali
    - Utilizzati per calcolare l'orientamento di un robot
  - Encoder: misurano la rotazione di un giunto prismatico
    - Calcolo dell'odometria
    - Possono essere relative, o assoluti
  - LIDARs: sensori di distanza ottici
    - Utilizzati per la navigazione a l'obstacle avoidance
    - Possono essere 3D e 2D
  - Videocamera: fornisce informazioni visive sull'ambiente
    - Utilizzati per estrarre informazioni generiche sull'ambiente

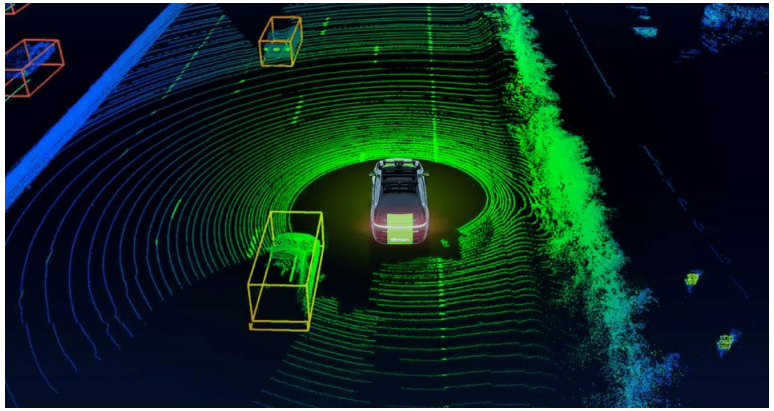
Controllo di robot: sensori

- Sensori GPS:
  - Rappresentano un sistema di posizionamento globale che aiuta la navigazione sulla terra
  - La posizione di un dispositivo GPS è determinate attraverso il tempo di ritorno di un segnale dai satelliti all'utente
  - La posizione finale è calolata tramite la triangolazione di più satelliti
    - Almeno 3 satelliti occorrano per avere una stima della posa
  - Le coordinate GPS sono espresse in angoli:
    - Latitudine
    - Longitudine

Controllo di robot: sensori

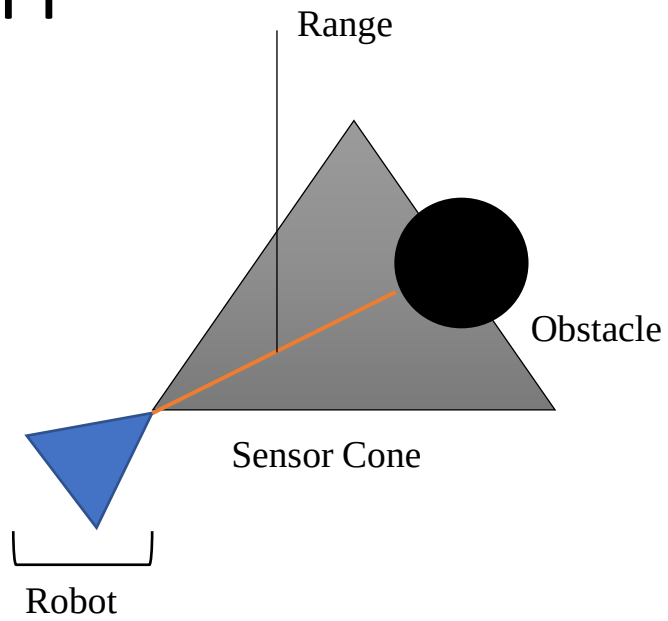
- I sensori di prossimità sono utilizzati per calcolare la distanza tra il robot e gli ostacoli che lo circonadno
  - Basati sugli infrarossi: calcolano la distanza dagli ostacoli misurando la quantità di luce riflessa considerando una serie di fasci di luce infrarossi
    - Ostacoli “trasparenti”?
  - Basati sugli ultrasuoni: misurano la distanza dagli ostacoli considerando il tempo di ritorno di un suono ultrasonico
  - La qualità di misura è profondamente influenzato dal tipo di sensore adottato



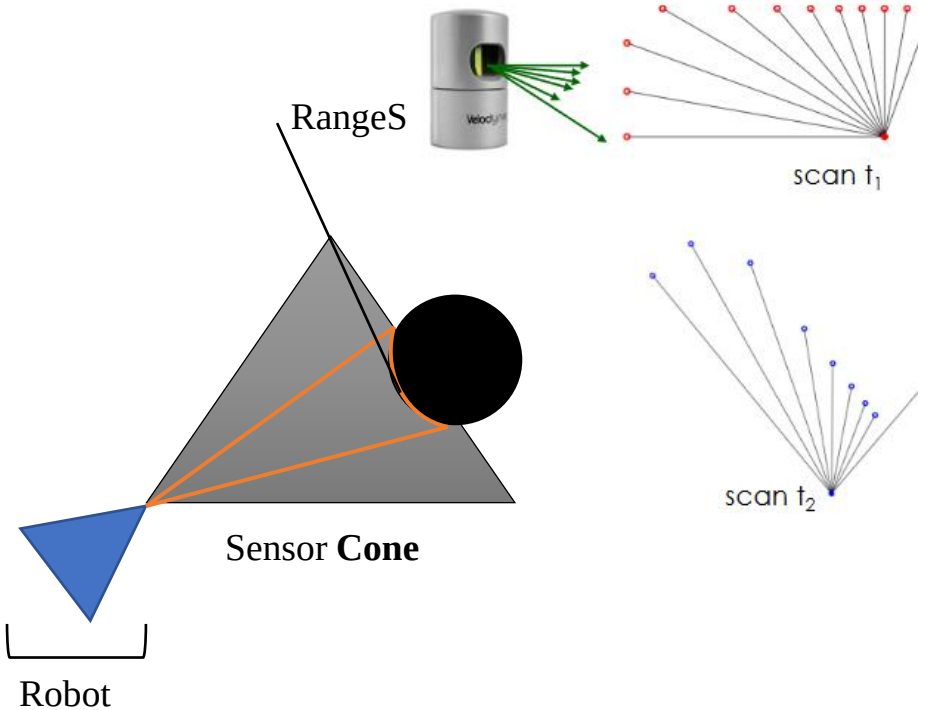


Controllo di robot: sensori

- Sensori a ultrasuoni
  - Applicazioni
    - Misura di distanza
    - Collisioni
    - UAV: stima dell'altitudine

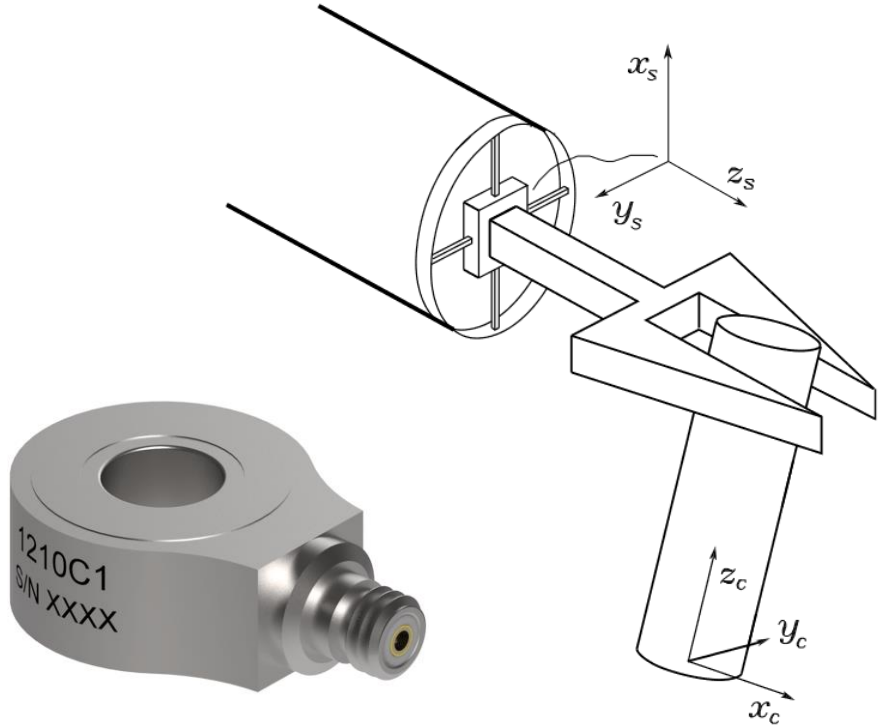


- Sensori a infrarossi
  - LIDARs (light detection and ranging)
  - Possono avere una circonferenza di 360°
  - Possono misurare la distanza anche nel 3d
    - Ancora molto costosi per applicazioni di tipo generale



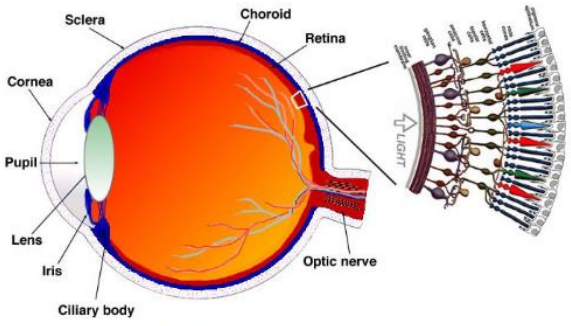
Controllo di robot: sensori

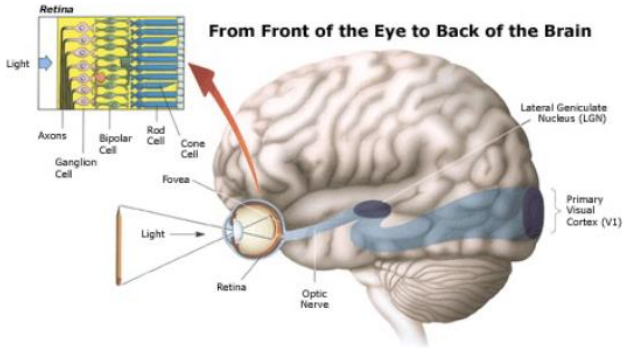
- Sensori di forza e coppia
- Misura le tre componenti di forza e le tre componenti di
- Momento di contatto tra manipolatore e ambiente
- Utilizzati per stimare le forze di contatto tra il manipolatore (l'organo terminale) e l'ambiente



Controllo di robot: sensori

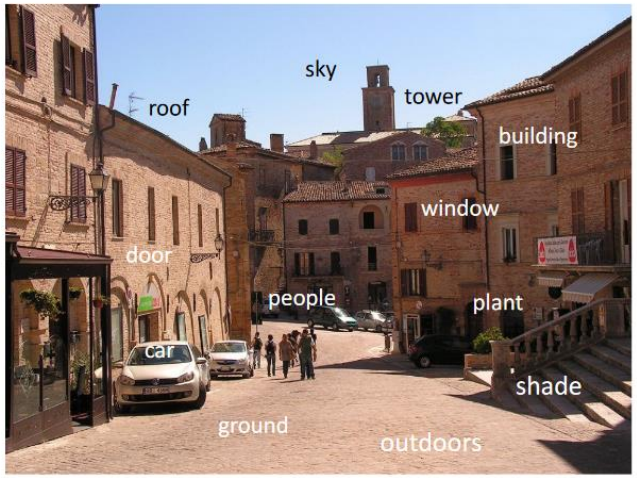
- I sensori di visione sono i sensori che forniscono il maggior numero di informazioni sull'ambiente rispetto gli altri
  - La retina contiene milioni di fotorecettori
  - Fornisce informazioni per un ammontare di 3 GBytes/s circa
- Gran parte dell'attività celebrale è dedicata all'elaborazione di queste informazioni
- Sensori di visione
  - Standard webcam
    - Bassa velocità
    - Ottiche standard
    - Poca possibilità di variazione di parametri
  - Camera industriali
    - Alta velocità di trasmissione
    - Possibilità di modificare le ottiche
    - Possibilità di commutare i parametri



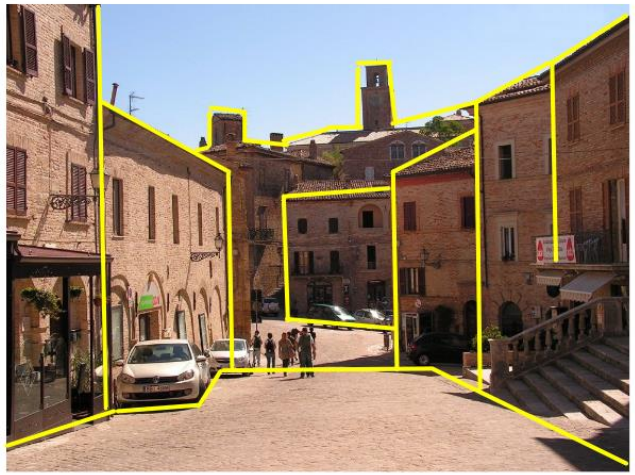


Controllo di robot: sensori

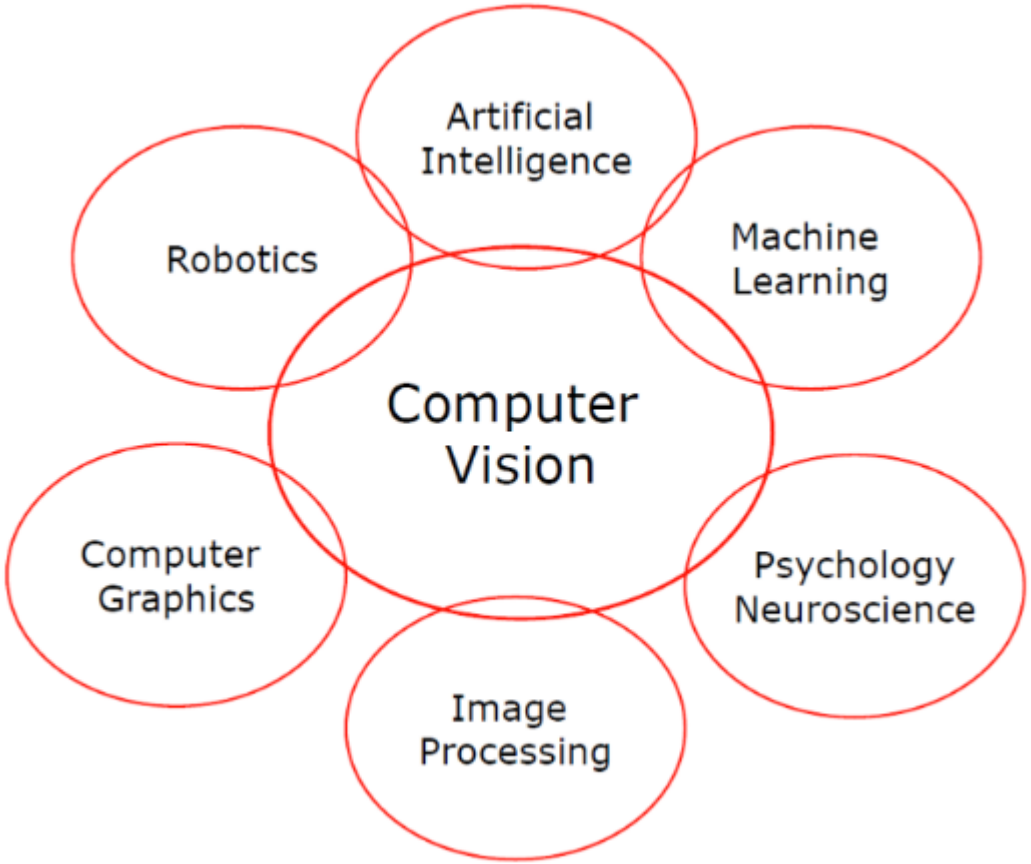
- Elaborazione delle imamgini
  - Computer vision
    - Campo di lavoro interdisciplinare
    - Automatizzare l'estrazione di informazioni salient da una o più immagini
  - Operazioni comuni
    - Thresholding
    - Binarization
    - Features detection
    - Visual odometry
  - Campo di applicazione del machine learning



Semantic information

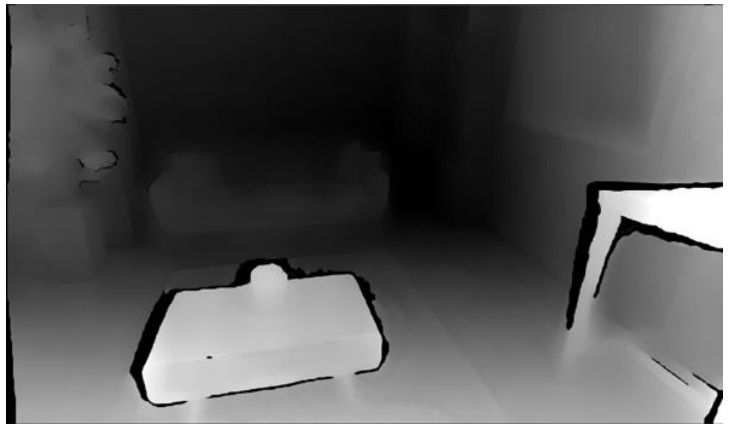


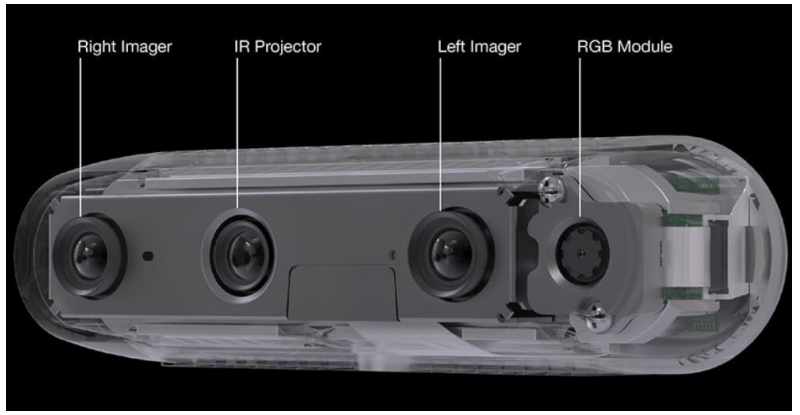
Geometric information



Controllo di robot: sensori

- Per eseguire task di robotica, tipicamente una componente fonramentale è la distanza degli oggetti dagli attuatori o dal corpo del robot
- Con una sola camera non è possibile stimare la distanza a meno di tecniche di trangolazione
- Alcuni sensori associano lidear e camera per fornire informazioni 3d sui pixel della camera
  - Sensori depth: RGB-D
  - La distanza è fornita direttamente in metri
  - Un sensore RGB-D fornisce direttamente la mappa di profondità:
    - Immagine monocromatica di 8bit
    - Pixel chiari: punti vicini
    - Pixel scuri: punti lontani





Controllo di robot: sensori

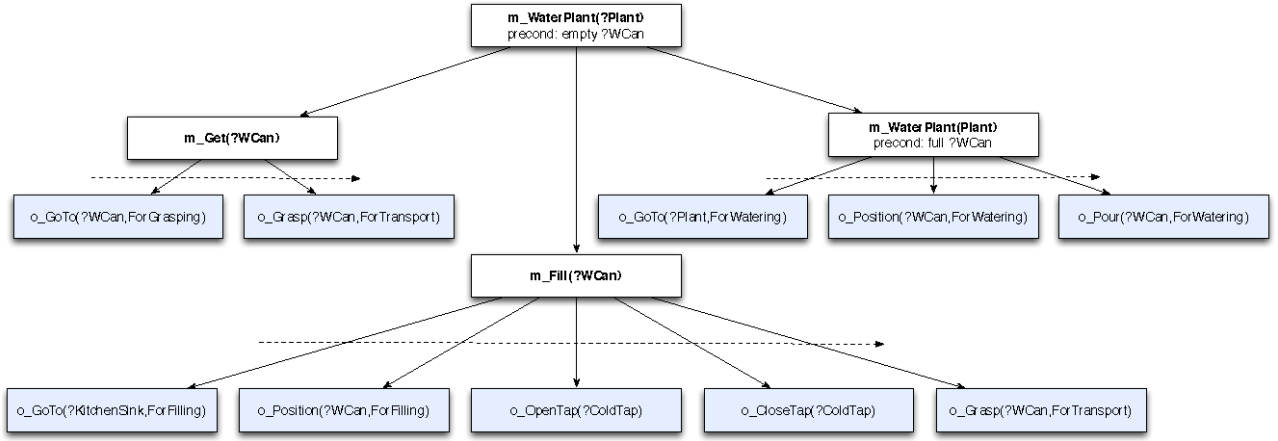
- La scelta del sensore di pende da diversi fattori
  - Il task da svolgere
  - Il tipo di segnale adatto al task
  - Il costo
  - La velocità di aggiornamento
  - L'errore di misura (un errore di tipo casuale)

Controllo di robot: problemi aperti

- Problemi di alto livello
  - Pianificazione di task
  - Percezione:
    - Object recognition / detection
    - Riconoscimento semantico
- Problemi di medio livello
  - Pianificazione di percorsi
  - Localizzaione
  - Mapping
  - Localizzazione e mapping
  - Manipolazione: grasping
- Problemi di basso livello
  - Pianificazione del moto
  - Controllo cinematico diretto e inverso
  - Controllo a dinamica diretta e inversa
  - Controllo robusto

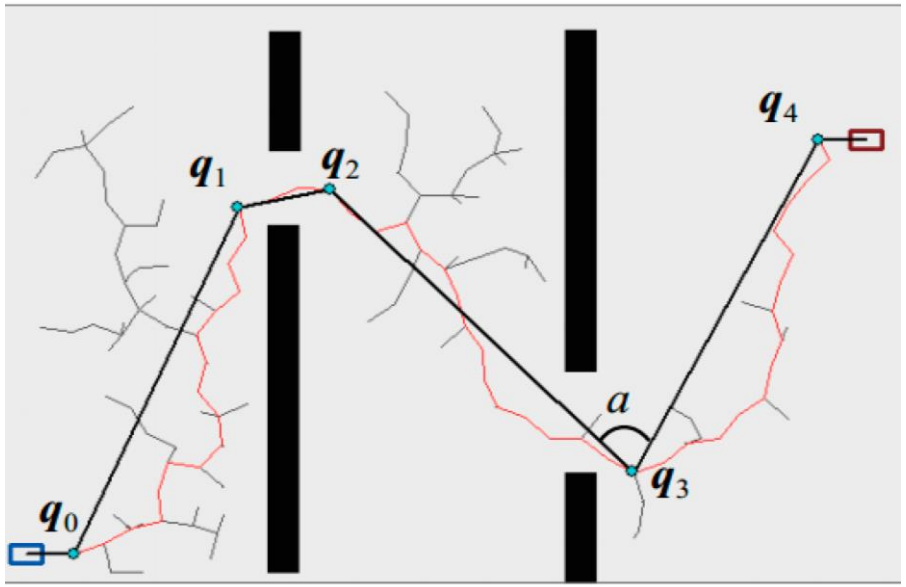
Controllo di robot: problemi aperti

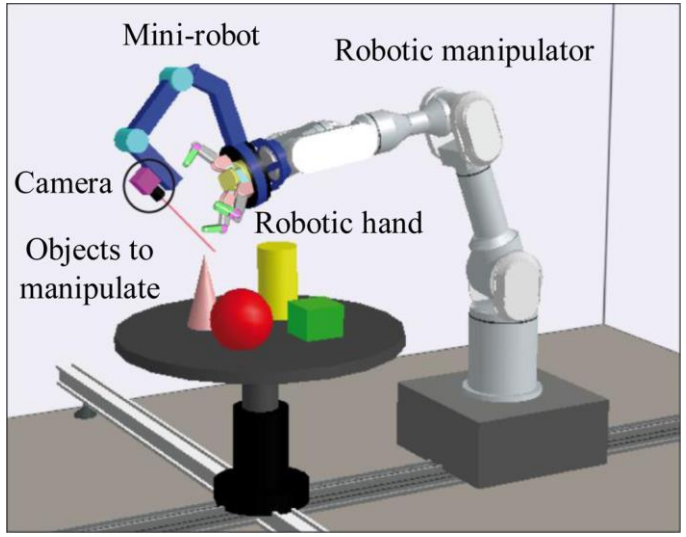
- Problemi di alto livello
  - Pianificazione di task
    - Considerando tutte le possibili azioni eseguibili da un robot, il suo stato e l'effetto delle azioni, trovare la migliore sequenza per portare a termine un determinato compito
  - Percezione:
    - Object recognition / detection
      - Considerando i sensori di visione o depth
        - Riconoscere uno specifico oggetto
      - Riconoscimento semantico
        - Capire che tipo di oggetto è

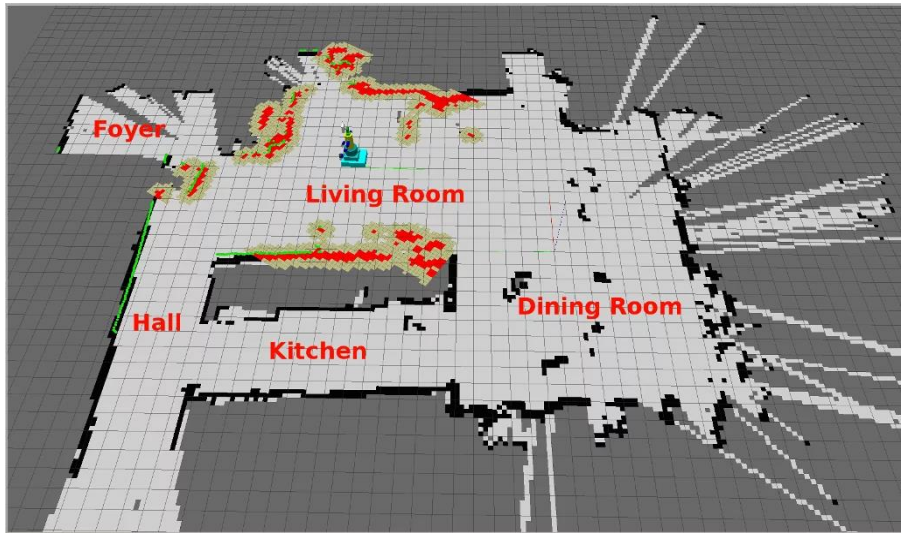


Controllo di robot: problemi aperti

- Problemi di medio livello
  - Pianificazione di percorsi
  - Localizzaione
  - Mapping
  - Localizzazione e mapping
  - Manipolazione: grasping

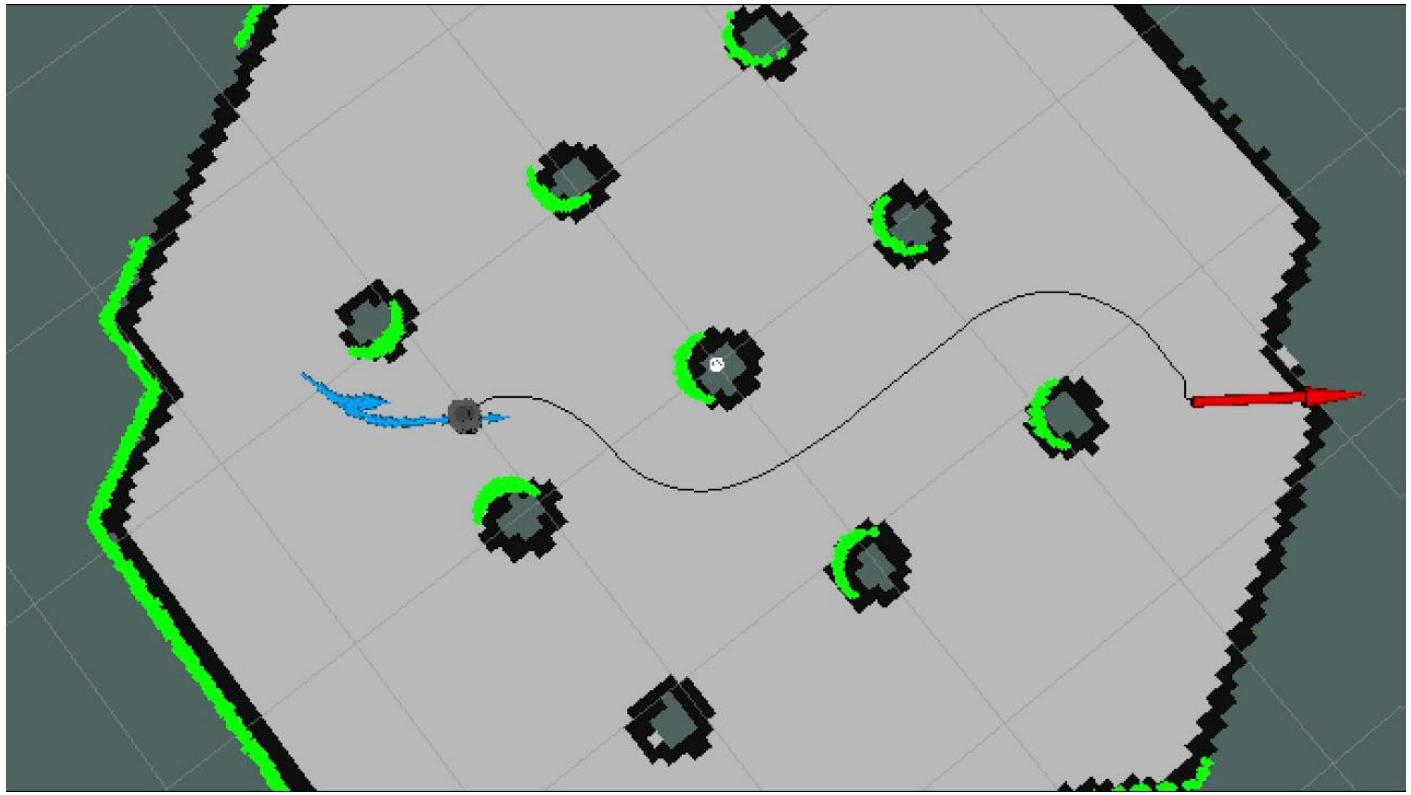






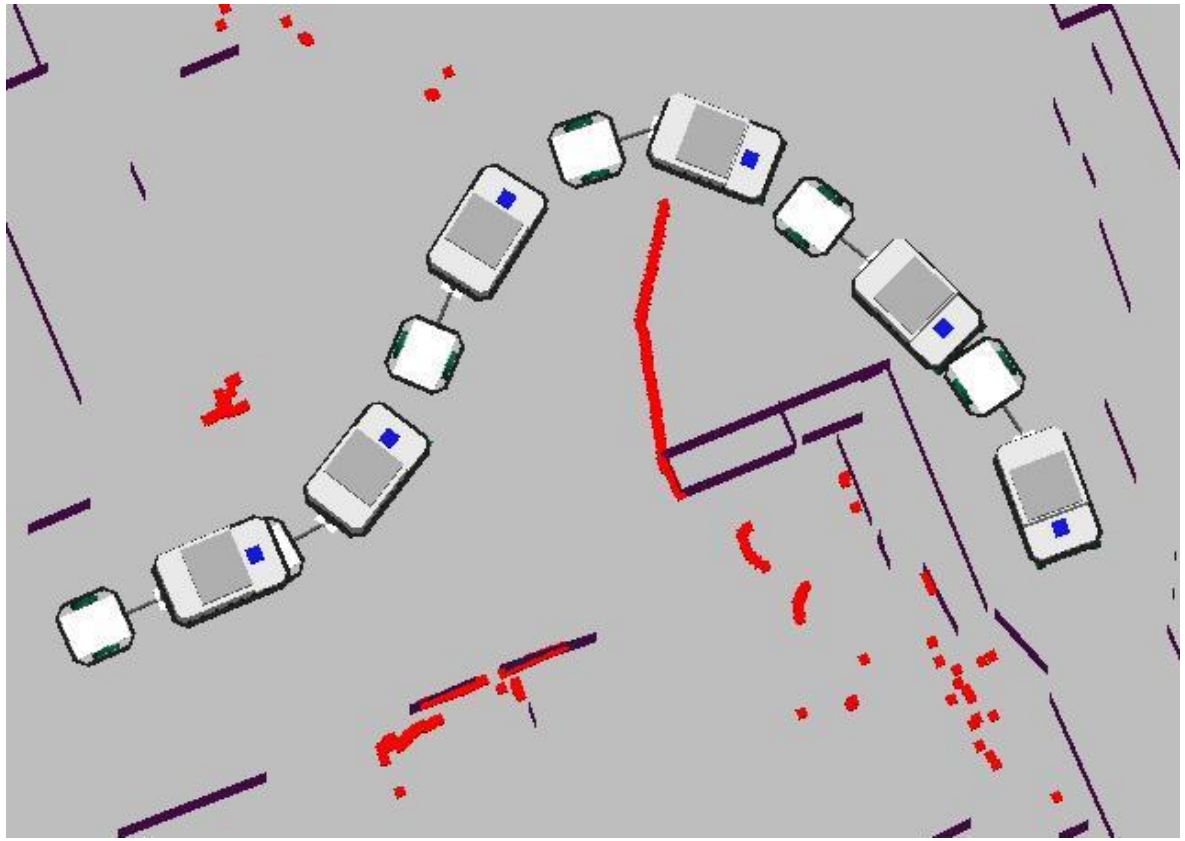
Controllo di robot: problemi aperti

- Robotica mobile
  - Pianificazione di traiettorie



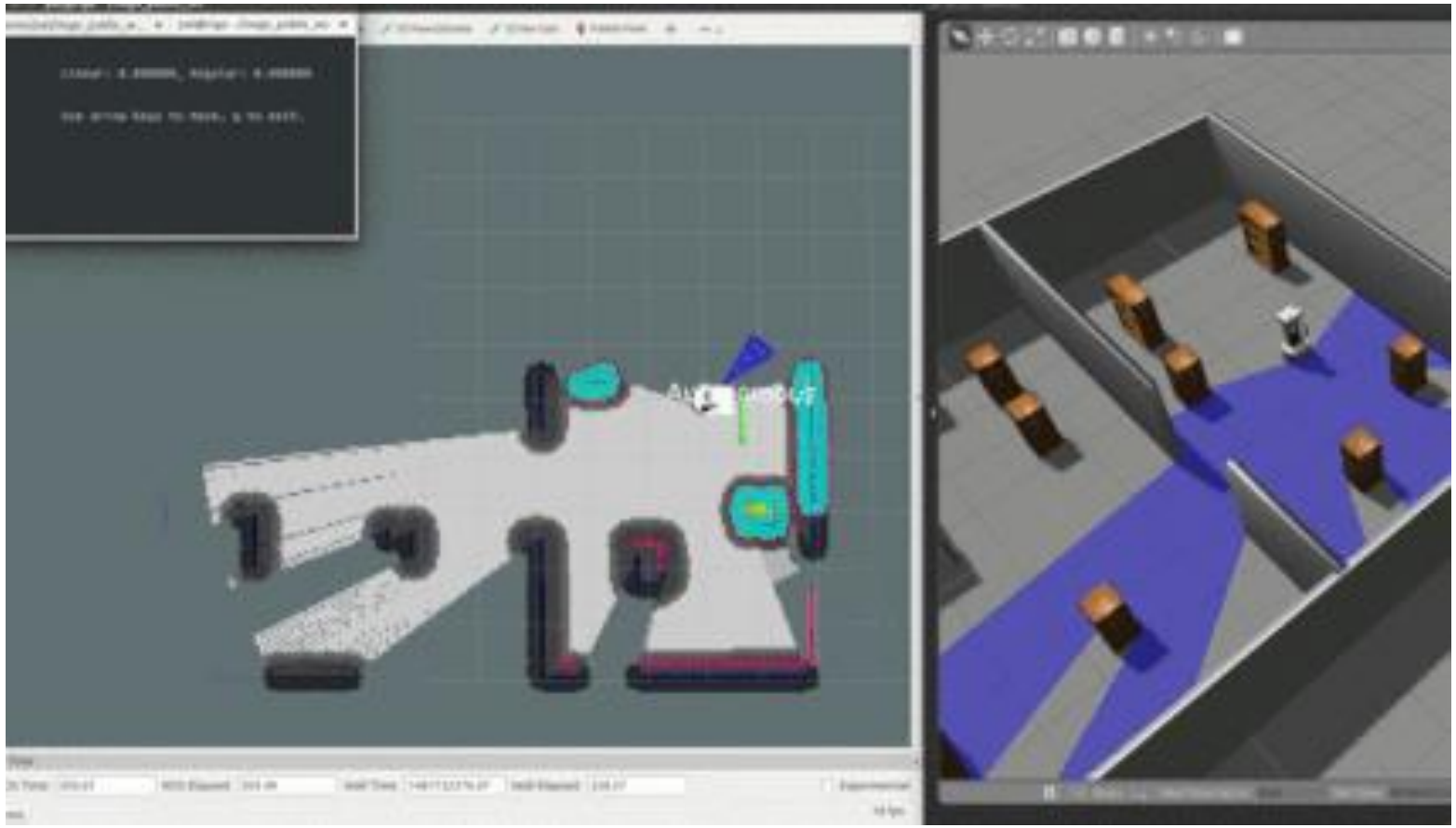
Controllo di robot: problemi aperti

- Robotica mobile
  - Pianificazione di traiettorie
  - Pianificazione di moto
  - Motion Planning



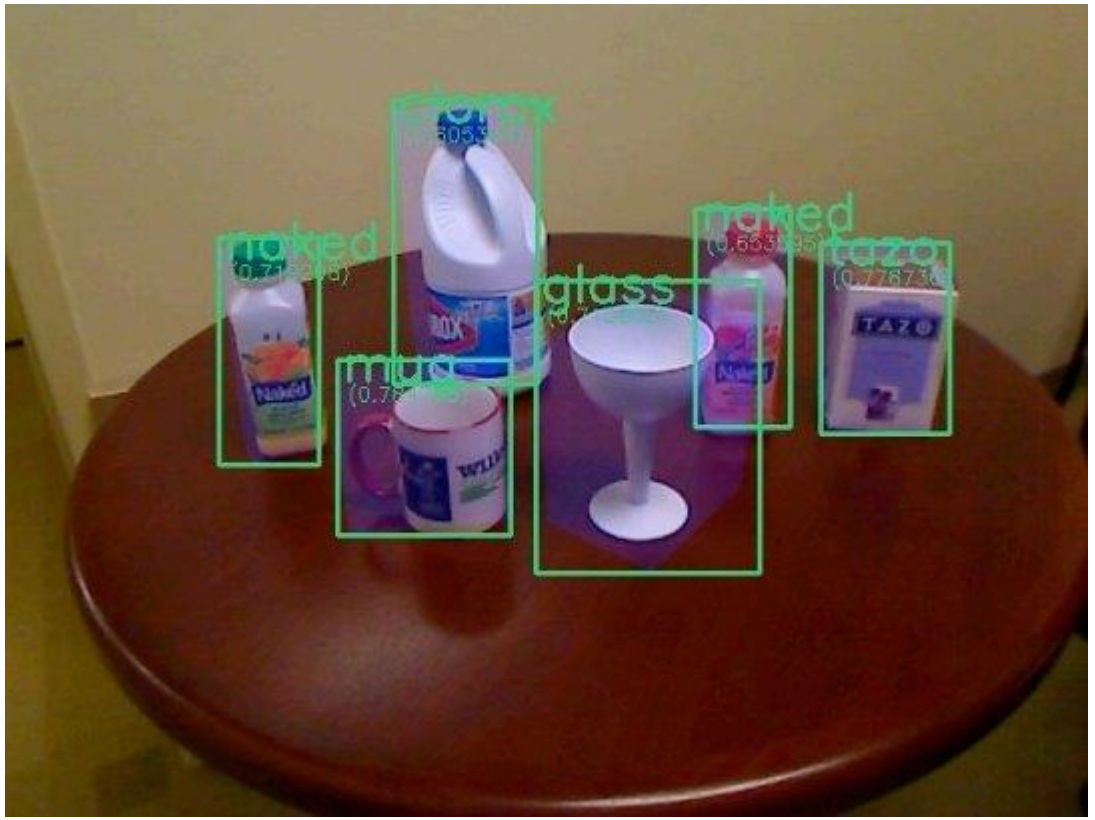
Controllo di robot: problemi aperti

- Robotica mobile
  - Pianificazione di traiettorie
  - Pianificazione di moto
  - Localization and Mapping (SLAM)



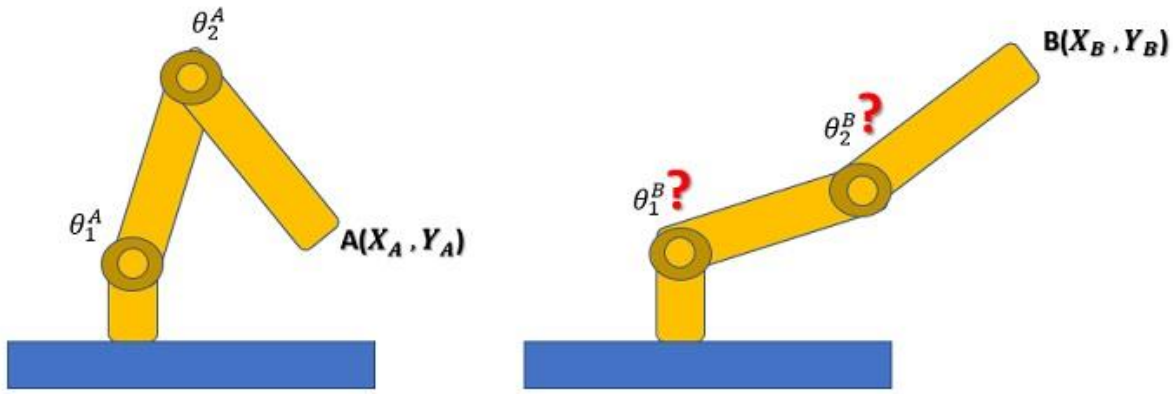
Controllo di robot: problemi aperti

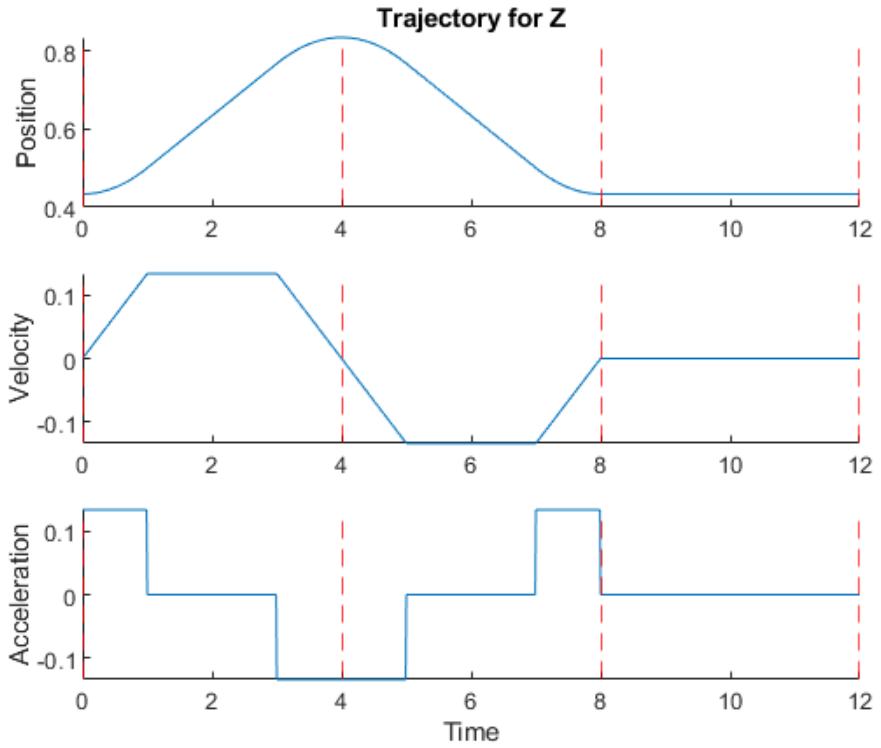
- Manipolazione
  - Object recognition and detection



Controllo di robot: problemi aperti

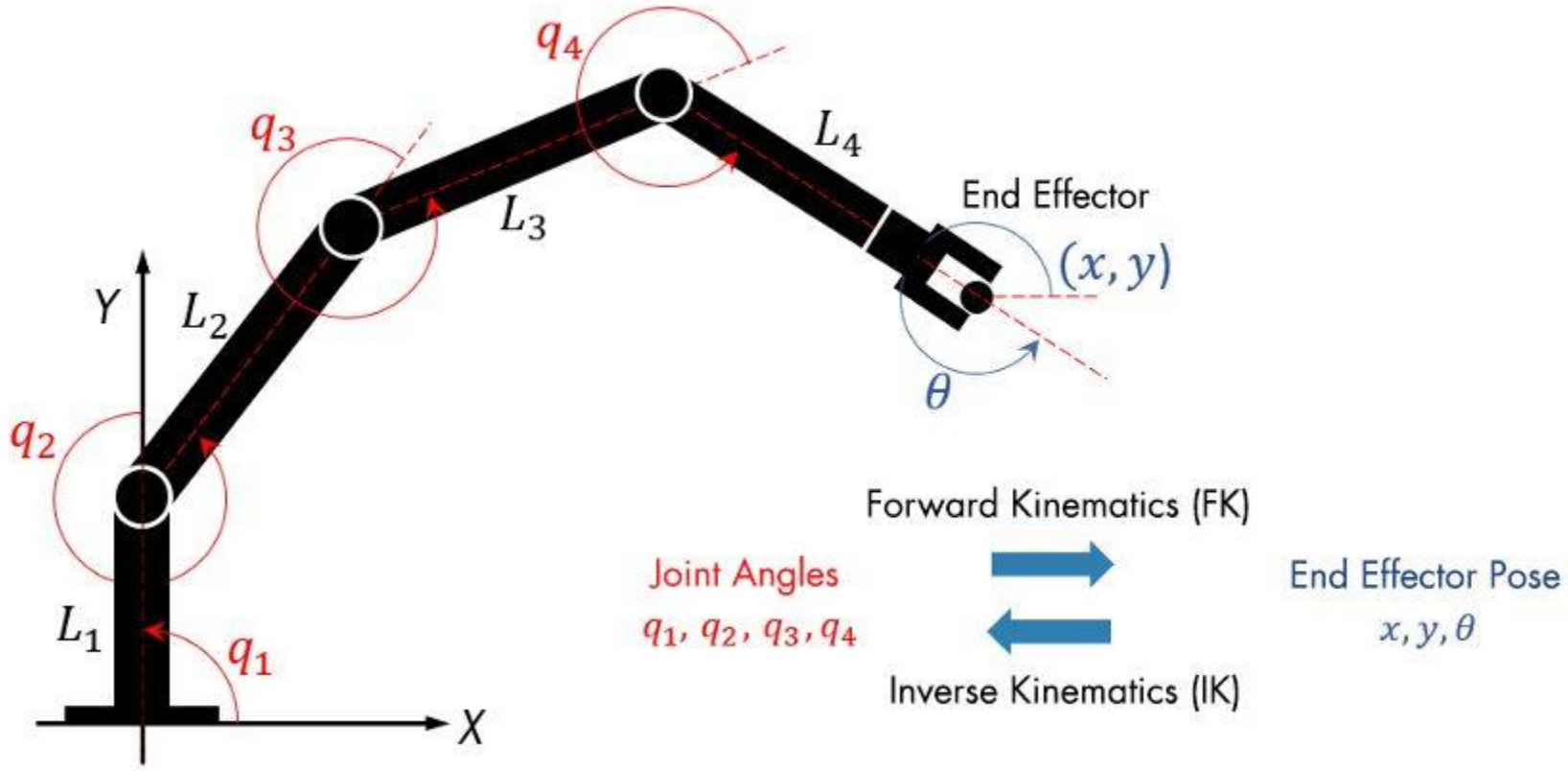
- Problemi di basso livello
  - Pianificazione del moto (traiettorie)
  - Controllo cinematico diretto e inverso
  - Controllo a dinamica diretta e inversa
  - Controllo robusto





Problemi comuni in robotica

- Manipolazione
  - Object recognition and detection
  - Forward-Inverse kinematics



Controllo di robot: conclusioni

- Programmare robot industriale per task industriali è un problema risolto
- Programmare sistemi robotici avanzati è un problema ancora aperto e un campo di lavoro eterogeneo
- Conoscenze da molteplici settori
  - Matematica
  - Fisica
  - Algebra
  - Informatica
  - ...
- Progressi in questo settore sono garantiti dalla possibilità di utilizzare librerie che risolvono problemi «aperti»
  - Ci si può focalizzare solo sul compito di interesse
  - Si può velocizzare la programmazione del robot

Take a break: 15 minutes

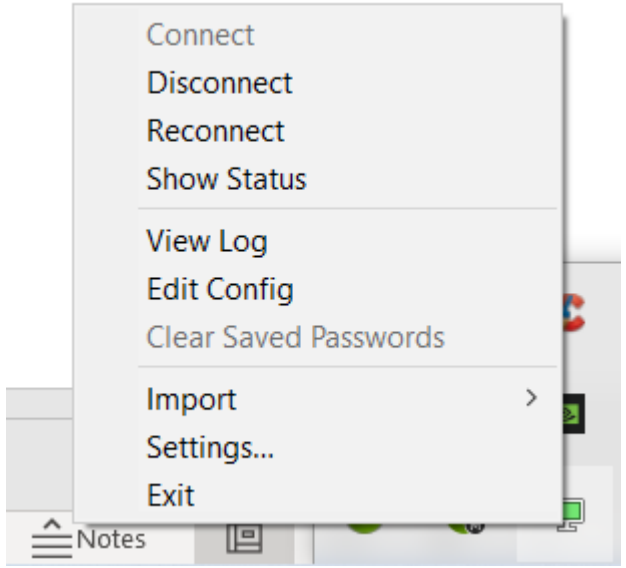
Introduzione ai sistemi operativi Linux e command shell linux

Ambiente di lavoro

- I robot industriali si programmano attraverso linguaggi di programmazione tipicamente nativi:
  - KRL: kuka robot language
  - Kuka Sunrise
  - RoboDk
- Sviluppare applicazioni robotiche per sistemi complessi non è un compito semplice
  - Troppi problemi aperti
  - Conoscenze eterogenee necessarie per sviluppare semplici applicativi
- E’ necessario utilizzare linguaggi di programmazione standard, in modo da sfruttare librerie esterne nelle proprie applicazioni

Ambiente di lavoro

- Collegamento alla macchina virtuale di amazon (AWS)
  - OpenVPN per entrare nella rete della AWS
    - Installare OpenVPN
      - https://swupdate.openvpn.org/community/releases/OpenVPN-2.5.7-I602-amd64.msi
      - Import file -> selezionare file .ovpn
      - Connettersi alla vpn
  - Remote Desktop per utlizzare la macchina virtuale
    - La macchina virtuale ha un indirizzo ip della classe: 42.2.10.XX
      - 10.42.2.10 – Jonathan
      - 10.42.2.11 – Alessio
      - 10.42.2.12 – Giovanni
      - 10.42.2.13 – Chiara
      - 10.42.2.14 – Davide
      - secsi

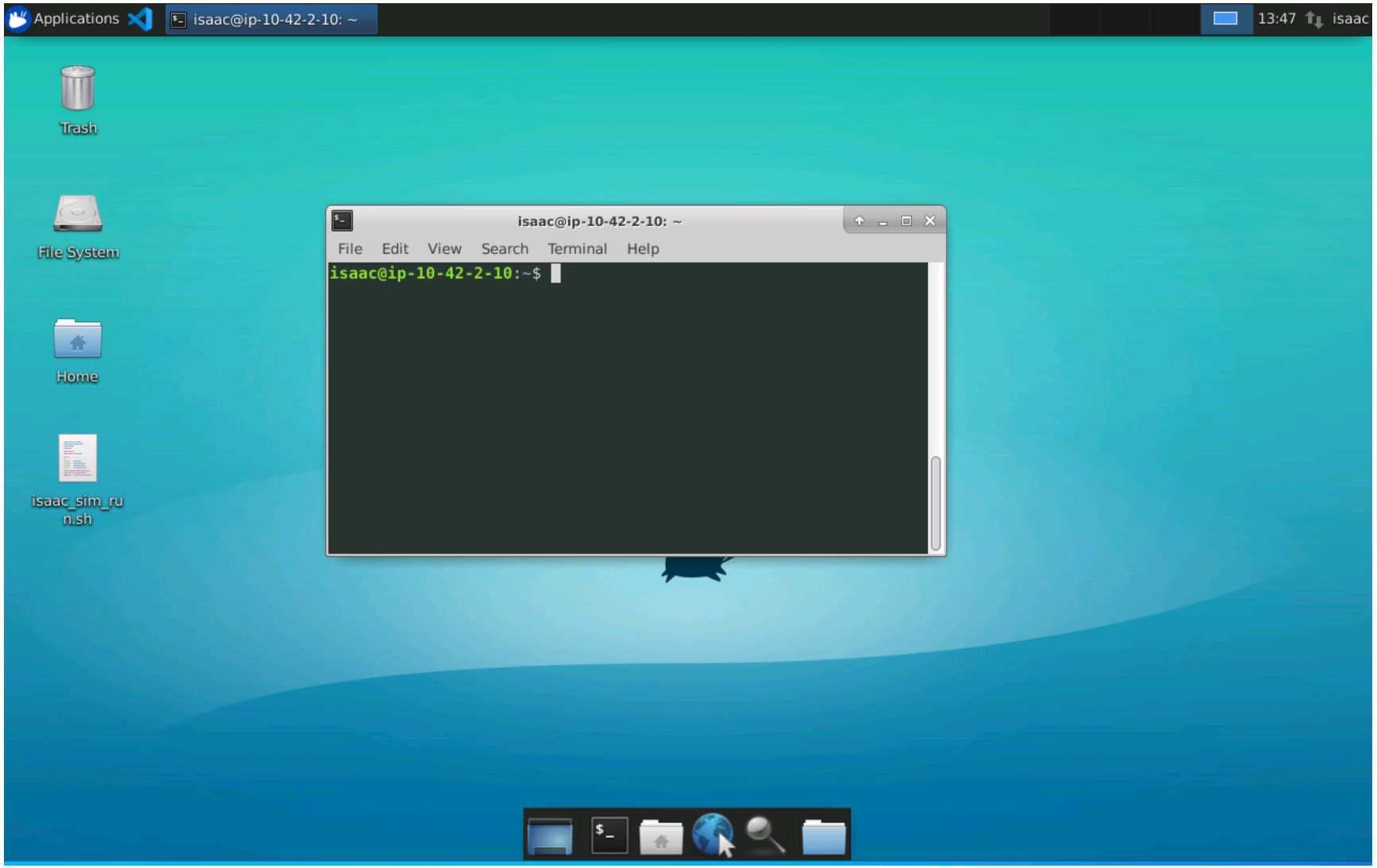


Ambiente di lavoro

- Setup tipico per la programmazione di robot:
  - OS: Sistema operativo bassato su unix
  - Linguaggio di programmazione: C++, python
  - C++: tempi di esecuzione rapidi e versatilità
  - Python: linguaggio di alto livello molto pratico per l'elaborazione di sensori e il machine learning
- Il termine Linux si riferisce a una famiglia di sistemi operativi open source basati sul kernel unix.
  - Ubuntu è oggigiorno una delle distribuzioni più diffuse
  - Una nuova versione di Ubuntu è rilasciata ogni 6 mesi
  - Ogni due anni viene rilasciata una nuova versione in forma di LTS: Long-Term Support
  - Attualmente la 22.04 è la LTS più recente

Ambiente di lavoro

- Interfaccia: basata su xfce



Comandi linux

- $ man
- $ pwd
- $ ls
- $ cd
- $ mkdir e $ rmdir
- $ rm
- $ cp
- $ mv
- $ locate
  - $ updatedb

Comandi linux

- $ echo
- $ cat
- $ nano
- $ touch
- $ sudo
- $ chmod
  - umask per i permessi di scrittura, lettura e esecuzione
- $ ping
- $ grep

0	None
1	only execute
2	only write
3	write/execute
4	only read
5	read/execute
6	read/write

Linux filesystem

- E’ un tipo di filesystem ad albero
- La radice del filesystem è la root: /
  - Gli utenti semplici non possono operare modifiche a questo livello
  - Altri directory appartenenti all’amministratore di sistema sono
    - /bin
    - /etc
    - /dev
    - /user
    - ….
- Lo spazio utente è rilegato nella sua home directory
  - /home/USERNAME

Linux environment

- La shell linux (terminale) è chiamato bash
- Quando un nuovo terminale si apre, una serie di file di conigurazione sono elaborati
- In particolare, vengono eseguiti i file /etc/bash.bashrc a poi quello posto nella directory home `~/.bashrc`
- Se si vuole modificare la configurazione di sistema di un utente, è possibile includere la modifica richiesta nel bashrc dell'utente in modo da renderla permanente.
- Esercizio 1.1:
  - Aggiungere un messaggio di benvenuto ogni qual volta si apre un nuovo terminale

Linux environment

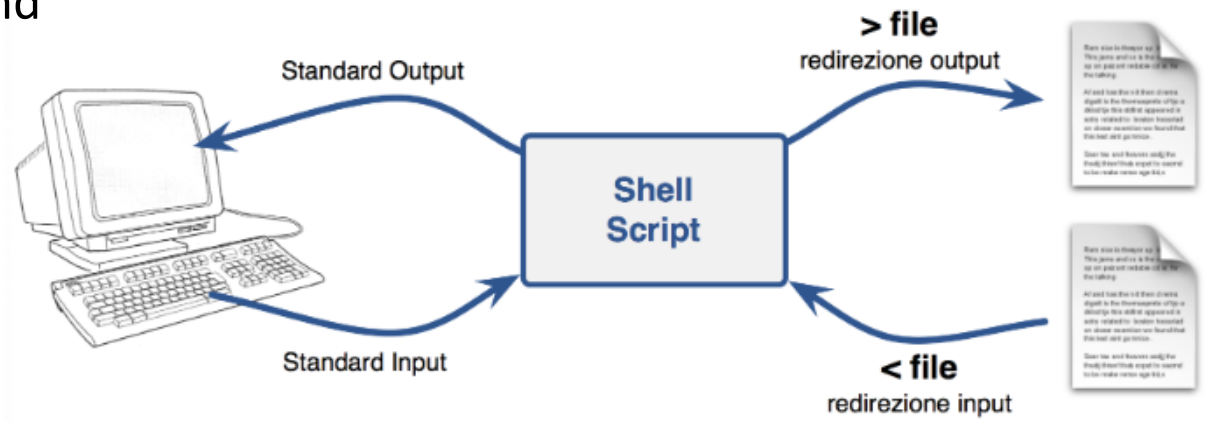
- Un elemento importante dell'ambiente linux è caratterizzato dalle sue variabili di ambiente.
- Una variabile può essere esportata con il comando $ export
- Il valore di una variabile può essere visualizzato con il comando $ echo
  - Alcune variabili di ambiente sono determinati per configurare il corretto modo di compilazione o esecuzione di alcuni eseguibili
  - LD_LIBRARY_PATH
  - PATH
  - ...

Manager di pacchetti: APT

- Ubuntu mette a disposizione un pratico manager di pacchetti: APT
- APT permette di installare, aggioranre e disinstallare software in maniera automatica
- I pacchetti software sono inoltre certificati dalla community
- apt-get install
- apt-get update
- apt-get upgrade
- APT scova i pacchetti da una repository software
  - E’ possibile aggiungere repository software
- E’ possibile cercare pacchetti:
  - $ apt-cache search

Redirezione I/O

- Ogni processo in linux può essere legato a 3 diversi stream
  - Stdin
  - Stdout
  - Stderr
- L'output di un comando o di un processo può essere rediretto in modo da intercettare lo stream e utilizzarlo nei propri programmi
  - Salvare l'output su un file
  - Usare il contenuto di un file come input di un comando (processo)
  - `>` Redirezione dell'output
  - `>>` Redirezione dell'output in modalità append
  - `<` Redirezione dell'input
  - `2>` Redirezione di messaggi di errore



Redirezione I/O

- `$ echo "ciao a tutti" > file`
- `$ more file`
  - `ciao a tutti`
- `$ echo "ciao a tutti" >> file`
- `$ more file`
  - `ciao a tutti`
    `ciao a tutti`
- `$ wc < file`
  - `2 6 26`

Redirezione I/O

- La pipe, realizzata per mezzo del metacarattere `|`, è una catena di montaggio, serve cioè per comporre `n` comandi “in cascata”, in modo che l’output di ciascuno sia fornito in input al successivo
- L’output dell’ultimo comando è l’output della pipeline
- Permette di implementare una ricerca ricorsiva utilizzato assieme al comando grepA

Esercizio 2.1

- Partendo dalla propria home directory:
  - Creare una directory chiamata: Robotica.
    - E’ vuota?
  - Creare una directory Eserc
  - Spostarsi in Robotica
  - Rinominare Eserc in EsShell
  - Copiare EsShell in Robotica
  - Rimuovere la copia originale di EsShell
  - Creare, in Robotica/EsShell un file README, contenente la stringa «Esercizi di robotica»
  - Aggiungere a README una seconda riga: Introduzione ai comandi linux
  - Stampare nel terminale il contenuto del file
  - Tornare nella propria home.

- Tempo: 10 minuti

Script bash

- I comandi eseguibili da shell possono essere inclusi in speciali file chiamati script bash
- .sh è la tipica estensione di questi script
- Utili per sequenze di comandi ripetitive/automatiche
- Per eseguire uno script
  - Editare un file di testo
  - Rendere eseguibile il file tramite il comando `chmod`
    - Digitare il nome del file (previa raggiungibilità via path)
  - In alternativa, senza permessi in esecuzione, è possibile utilizzare i caratteri ./ per richiedere l'esecuzione

Script bash

- ```
  #!/bin/bash
  ```

- ```
  a=13
  echo $a
  b=$a
  echo $b
  ```

- ```
  A=‘echo Hello!‘
  echo $a
  a=‘ls -l‘
  ```

- ```
  echo $a
  exit 0
  ```

Git

Version control software (VCS)

- Version control (also known as revision control, source control, or source code management) is a class of systems responsible for managing changes to
  - computer programs
  - documents
  - large web sites
  - …other collections of information.
- Version control is a system for tracking the state of files and folders

- Git is a Version Control System (VCS) designed to make it easier to have multiple versions of a code base, sometimes across multiple developers or teams
- It allows you to see changes you make to your code and easily revert them.
- The most famous implementation of Git resides on github

Version control software (VCS)

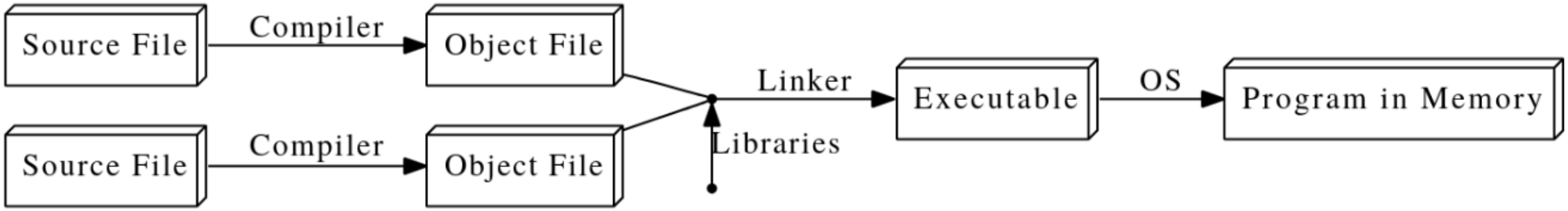
- Github page for the material of this course
  - https://github.com/robotic-software
  - Source code
  - Configuration file
- Provide me your username or e-mail address to invite you to the team
- To use Git on windows:
  - https://git-scm.com/download/win

Esempi di programmazione Cpp

Compilazione con CMakeLists.txt

- Per eseguire un codice C++, dobbiamo generare un eseguibile
  - Il processo di compilazione sfrutta il complatore Gcc

```
GCC -> make -> CMake
```



Compilazione con CMakeLists.txt

- Per eseguire un codice C++, dobbiamo generare un eseguibile
  - Il processo di compilazione sfrutta il complatore Gcc

```
GCC -> make -> CMake
```

Sample program **hello.c**.

```
1 #include <stdio.h>
2 int main() {
3     printf("Hello, world!\n");
4     return 0;
5 }
```

```
$ gcc hello.c
$ chmod a+x a.out
$ ./a.out
```

To specify the output filename, use -o option:

```
$ gcc -o hello.exe hello.c
```

```
all: hello

hello: hello.o
    gcc -o hello hello.o

hello.o: hello.c
    gcc -c hello.c

clean:
    rm hello.o hello
```

```
1  # Specify the minimum version for CMake
2  cmake_minimum_required(VERSION 2.8)
3  # Project's name
4  project(hello)
5  # Set the output folder where your program will be
       created
6  set(CMAKE_BINARY_DIR ${CMAKE_SOURCE_DIR}/bin)
7  set(EXECUTABLE_OUTPUT_PATH ${CMAKE_BINARY_DIR})
8  set(LIBRARY_OUTPUT_PATH ${CMAKE_BINARY_DIR})
9  # The following folder will be included
10 include_directories("${PROJECT_SOURCE_DIR}")
1  add_executable(hello ${PROJECT_SOURCE_DIR}/hello.c)
```

Compilazione con CMakeLists.txt

- Esempio 1.1: compilare un codice di esempio «hello world» utilizzando gcc, make e cmake

GCC -> make -> CMake

Sample program **hello.c**.

```
1 #include <stdio.h>
2 int main() {
3     printf("Hello, world!\n");
4     return 0;
5 }
```

```
$ gcc hello.c
$ chmod a+x a.out
$ ./a.out
```

To specify the output filename, use -o option:

```
$ gcc -o hello.exe hello.c
```

```
all: hello

hello: hello.o
    gcc -o hello hello.o

hello.o: hello.c
    gcc -c hello.c

clean:
    rm hello.o hello
```

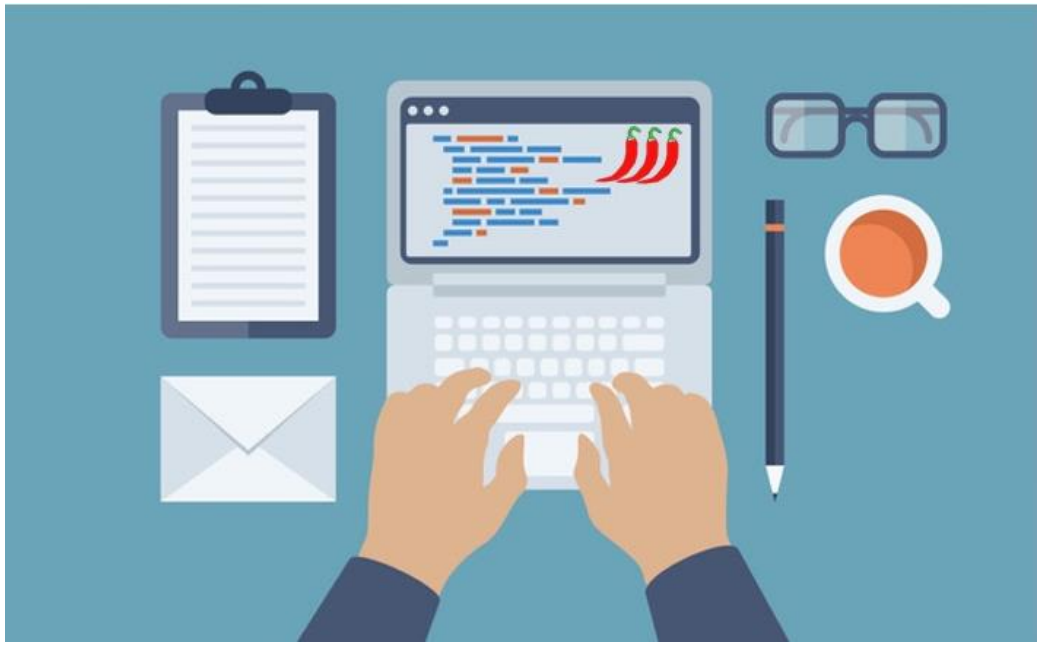
```
1  # Specify the minimum version for CMake
2  cmake_minimum_required(VERSION 2.8)
3  # Project's name
4  project(hello)
5  # Set the output folder where your program will be
       created
6  set(CMAKE_BINARY_DIR ${CMAKE_SOURCE_DIR}/bin)
7  set(EXECUTABLE_OUTPUT_PATH ${CMAKE_BINARY_DIR})
8  set(LIBRARY_OUTPUT_PATH ${CMAKE_BINARY_DIR})
9  # The following folder will be included
10 include_directories("${PROJECT_SOURCE_DIR}")
1  add_executable(hello ${PROJECT_SOURCE_DIR}/hello.c)
```

OOP con librerie boost

- I sistemi robotici sono multithreading
  - La parte di percezione non può fermarsi attendendo l'azione o la pianificazeione
- La classe Boost.Thread permette l'utilizzo di diversi processi in grado di condividere dati in maniera safe
- Boost.Thread implementa la classe per la creazione condivisione e sincronizzazione delle risorse che vivono nei thread
- Esempio 2.1: creare un codice cpp che lanci 10 thread utilizzando una funzione di classe. Ogni thread prende in ingresso un intero che stampa a video prima di uscire dalla funzione. Il numero in ingresso sarà sequenzialmente incrementato a seconda del thread
- Esempio 3.1: implementare un controllore PID in un sistema multi-threaeding

Esercizio 3.1:

- Creare un codice cpp che consiste di due funzioni eseguite parallelamente.
  - La prima funzione attende una stringa inserita da tastiera.
  - Dopo aver inserito la stringa, la seconda funzione inserisce questa stringa in un vettore dinamico (std::vector<string>) e stampa il contenuto di tutte le stringe ricevute fino a quel momento
- Tempo: 10 minuti



Esempi di programmazione Python

Esempi in python

- Esempio 4.1: creare uno script per la somma di 2 numeri, prima inseriti nel codice, poi richiesti da tastiera
- Esempio 5.1: creare uno script per la definizione di una classe Persona che abbia 3 attribbuti: nome, età e location
- Esempio 6.1: creare uno script multi-threading. Considerare una funzione che accetta un numero che indentifica l'attesa di quella funzione prima di concluderne l'esecuzione

Esercizio 4.1:

- Scrivere un programma utilizzando due funzioni parallele.
  - Entrambe le funzioni accettano due numeri in ingresso e stampano a video i numeri interi in ordine crescente.
  - La prima funzione stamperà solo i numeri pari, fino a raggiungere il secondo numero in ingresso, la seconda funzione, stamperà solo numeri dispari, fino a raggiungere il secondo numero in ingresso.
- Tempo: 10 minuti

Referenze

- Esercizi e documentazione al link:
  - https://github.com/robotic-software/L1

Fine lezione 1