Practical robotics

Fabrice LE BARS

- French version of the presentation (more up to date)
  - See https://www.ensta-bretagne.fr/lebars/robotique_pratique.pdf

Designing a robot

Designing a robot

- To be defined depending on robot purpose :
  - Mechanical structure
  - Actuators
  - Sensors
  - Embedded computer
  - Communication
  - Energy

- In this presentation, unmanned (without human onboard) **mobile robots between 50 cm and 4 m** will be mainly considered, often **inspired by RC community**

Designing a robot

- Vocabulary :
  - **Manned** : a human onboard controls the robot, otherwise it is **unmanned**

    UAV : Unmanned Aerial Vehicle, UGV : Unmanned Ground Vehicle, USV : Unmanned Surface Vehicle, UUV : Unmanned Underwater Vehicle
  - **Autonomous** : several possible levels of autonomy (e.g. with continuous monitoring by a human operator vs without any monitoring), the meaning depends on the context, in most cases it means a **robot moving without continuous low-level control by an operator**

    AGV, ASV, AUV

Designing a robot

- Off-the-shelf vs custom platform :
  - There are lots of existing off-the-shelf platforms, or parts of platforms
  - If the purpose of the robot is to test algorithms on a prototype, I can be a good idea to use off-the-shelf parts to minimize construction time
  - If the robot is expected to be significantly modified during its development, it is good to have a total control on its parts
  - Be careful to the reliability of the manufacturers, some parts might quickly disappear from their product line
  - RC community websites can be interesting for certain types of robots

Designing a robot

- Mechanical structure :
  - Can be significantly different from existing manned systems

    e.g. the shape or the type of moving pattern (vertical takeoff, reverse flight, strong accelerations, no need for breathable air, etc.)
  - Often need rapid prototyping methods

    e.g. 3D printers, see also CAD presentation using CATIA : https://moodle.ensta-bretagne.fr/mod/resource/view.php?id=32600

Designing a robot

- Actuators :
  - Might be the same as in manned systems, but very often electrical for small robots

Designing a robot

- Actuators :
  - Electrical motors : brushed vs brushless, step-by-step or not?

    Motors brushed DC :

    2 wires (**+ and -**, this can be used to identify them)

    DC = direct current, can be tested directly on a battery

    An electromagnet rotates in a static cage with permanent magnets, its power supply reversing with the axis rotation

    The power supply is on the axis, the **contact is made by 2 brushes** (+ and -)

    If it gets water inside, the contact might not be possible…

Designing a robot

- Actuators :
  - Electrical motors : brushed vs brushless, step-by-step or not?

    Brushless motors :

      3 wires (this can be used to identify them)

      Cannot be tested directly with a battery, need a motor controller/driver/ESC to generate the correct signals

      No electric contact between rotating part and fixed part

      The outside part is often rotating and axis is fixed (**outrunner**, invert is **inrunner**)

Designing a robot

- Actuators :
  - Electrical motors : brushed vs brushless, step-by-step or not?

    Brushless motors :

    We can put them inside a fluid without too much problems (however be careful to mechanical parts corrosion, current leaks if the fluid is conductive and electrolyze…)

    Waterproofness in sea water :

    In waterproof box filled with oil if high pressure

    Directly in the water if mechanical parts are resistant to corrosion and electromagnets protected with a resin

Designing a robot

- Actuators :
  - Electrical motors : brushed vs brushless, step-by-step or not?

    Brushless motors :

    Often more expensive than brushed but more efficient in terms of power consumption, torque, speed, durability for equivalent size, weight, especially thanks to the absence of brushes (less friction, usure, etc.)

Designing a robot

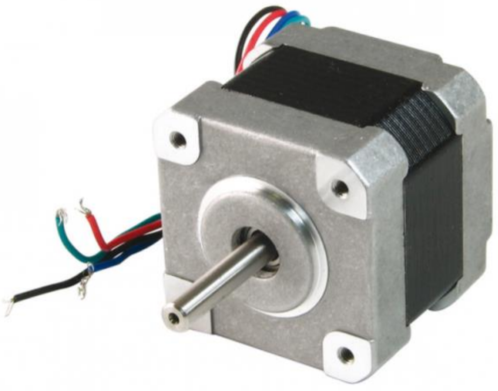
- Actuators :
  - Electrical motors : brushed vs brushless, step-by-step or not?
    - Brushless motors :
      - Often used in aerial RC drones
      - Important parameters (they have also similar meaning for other types of motors) :
        - KV : number of rotations per Volts or RPM/V
        - Corresponding propellers are often listed on the motor documentation, depending on the voltage input
        - If the propeller and/or the voltage is not correct, the maximum thrust might not be obtained, while the power consumption and the motor temperature will be higher…

Designing a robot

- Actuators :
  - Electrical motors : brushed vs brushless, step-by-step or not?
    - Brushless motors :
      - Empirical observations for propeller choice :
        - The higher the step, the faster the plane
        - The higher the diameter, the bigger the plane
        - The higher the step and the higher the diameter, the bigger the motor and the lower the Kv
        - For multirotors, the energy required to lift 1 gram is between 100 and 200 mW
        - See http://www.ecalc.ch/ to choose correctly the motors and propellers depending on the robot characteristics (aerial)

Designing a robot

- Actuators :
  - Electrical motors : brushed vs brushless, step-by-step or not?
    - Step-by-step motor :
      - >3 wires
      - Need an adequate driver board
      - Very accurate to be able to go to a specific angle (e.g. used in printers)



Designing a robot

- Actuators :
  - Electrical motors

    See FIPA 1 course :

    https://moodle.ensta-bretagne.fr/course/view.php?id=1162

Designing a robot

- Actuators :
  - Motor controller/driver/ESC
    - Enable to control the motors from control signals
    - Motors : high voltages and currents from the batteries
    - Control signals : low voltages and currents coming directly or indirectly from the embedded computer
    - Examples : PWM signals (the most used in RC community), I2C, etc.

Designing a robot

- Actuators :
  - Motor controller/driver/ESC
    - ESC for brushed motors: in decline in the RC community because of the brushless motors price drop…
    - ESC for brushless motors : often used in planes, quadrotors, helicopters in the RC community, so there is not always backward direction support…

Designing a robot

- Actuators :
  - Motor controller/driver/ESC

    Example : Robbe Rokraft (brushed motors)

Designing a robot

- Actuators :
  - Motor controller/driver/ESC

    Example : AfroESC (brushless motors)

PWM control signals,
Usual servomotor connector

To battery, PK3.5 connector

To motors

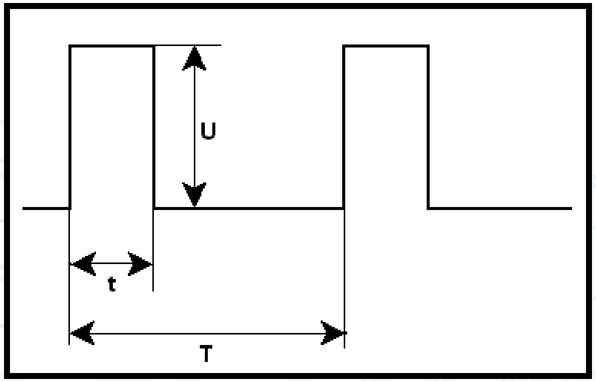
Designing a robot

- Actuators :
  - Motor controller/driver/ESC

    Functioning :

    Power sent to motors (and therefore their speed) depends on PWM control signal

    PWM = Pulse Width Modulation



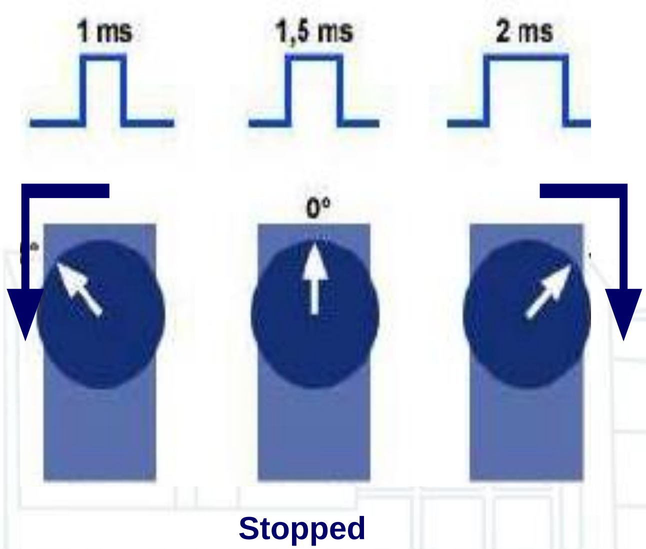
U : PWM voltage (5 V)
t : pulse width (between 1 and 2 ms)
T : period (20 ms)

Designing a robot

- Actuators :
  - Motor controller/driver/ESC

    Functioning :

      Correspondence between pulse width and rotation speed



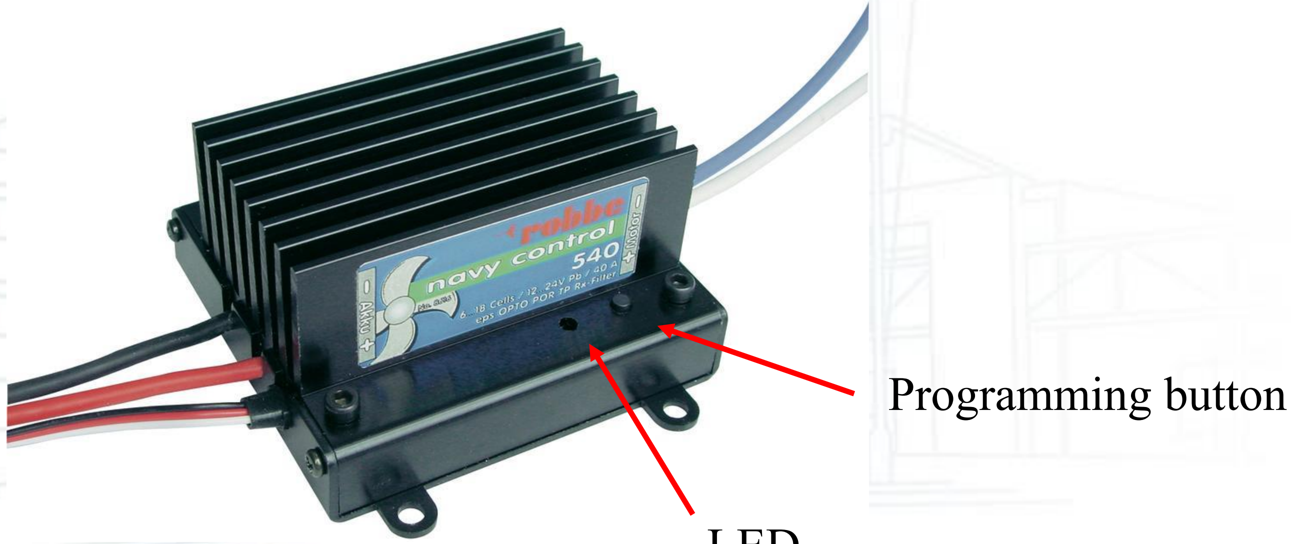
Motor state	Pulse width
Motor stopped	**1.5 ms**
Rotation in one direction, accelerating	**1.5 to 2.0 ms**
Rotation in reverse direction, decelerating	**1.0 to 1.5 ms**

Designing a robot

- Actuators :
  - Motor controller/driver/ESC

    Calibration :

    Some ESC must be calibrated : **max, min, neutral setup** depending on PWM signal, **check the documentation** (special button or starting procedure, might need a specific adapter to connect to computer with a specific configuration software…)



Designing a robot

- Actuators :
  - Motor controller/driver/ESC
    - BEC (Battery Elimination Circuit) :
      - On its « **servo connector** », there is usually **5 V (typ. 1-3 A)** available on the red wire (generated from the motor battery through an **internal voltage regulator**)
      - In the RC community, the purpose is to be able to provide power to the receiver as well as small actuators without needing to add a 5 V battery
      - On the contrary, some ESC (often those using opto/photo internal components) do not have a BEC and need to get 5 V on their red wire to work (they do not use the motor battery for their own power supply)

Designing a robot

- Actuators :
  - Motor controller/driver/ESC

    BEC (Battery Elimination Circuit) :

    Warning: if several ESC with BEC are on the robot, their BEC should not be all connected together => remove the red wire from the servo connector for all ESC except 1 (or all if 5 V is not needed)!

    Always be careful when there are several power sources on the robot!

    Be careful also to **+ loops or - loops** often possible when using SPS (high power relays), **check** if **common ground** is possible/impossible/necessary, etc.

Designing a robot

- Actuators :
  - Motor controller/driver/ESC

    Important parameters :

    ESC supported **current** must be **>** what the **motor** might need

    ESC supported **voltage** must be **>=** max voltage of the **battery** and must agree with what the motor accepts and where it is the most efficient (max speed and torque for a reasonable power consumption and temperature)

Designing a robot

- Actuators :
  - Servomotor = small motor + ESC
    - Controlled by PWM
    - Power supply 5 V (usually < 2 A)
    - 2 types of servomotors :
      - Position/angle control : turn e.g. from -40 to +40°
      - Speed control

Designing a robot

- Actuators :
  - Interface board
    - Connects computer part with electronics part (sensors, actuators)
      - Computer part : « brain » through programs on the PC
      - Electronics part : sensors, actuators

Designing a robot

- Actuators :
  - Interface board

    Example : IOIO board for Android smartphone/tablet, Windows/Linux PC

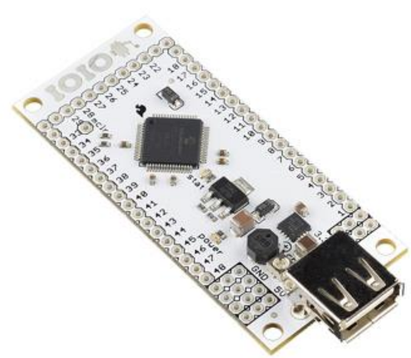
    Connected on the smartphone/computer USB port and is controlled by programs running on the smartphone/PC

    Can generate PWM signals, I2C

    Can generate and read digital signals

    Can read small voltages (from analog sensors such as telemeters, odometers, compass...)

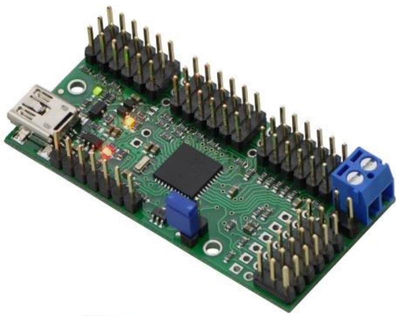
    …

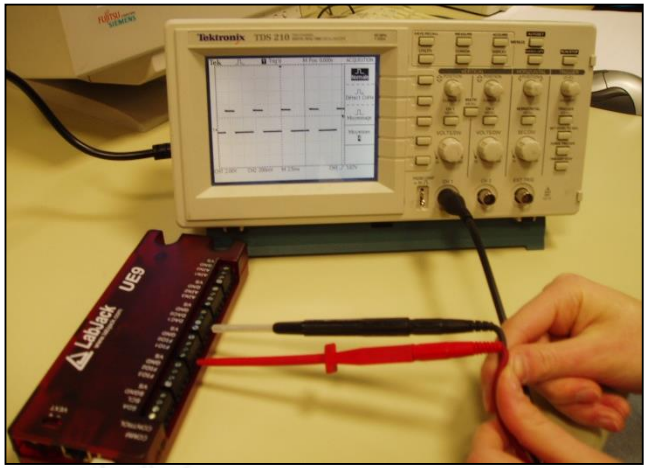


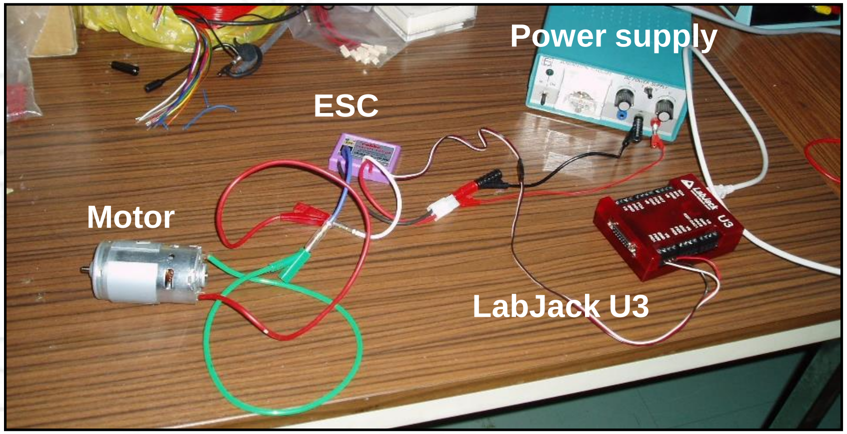
Designing a robot

- Actuators :
  - Interface board

    Other examples : Pololu Maestro, SSC-32, LabJack for PC







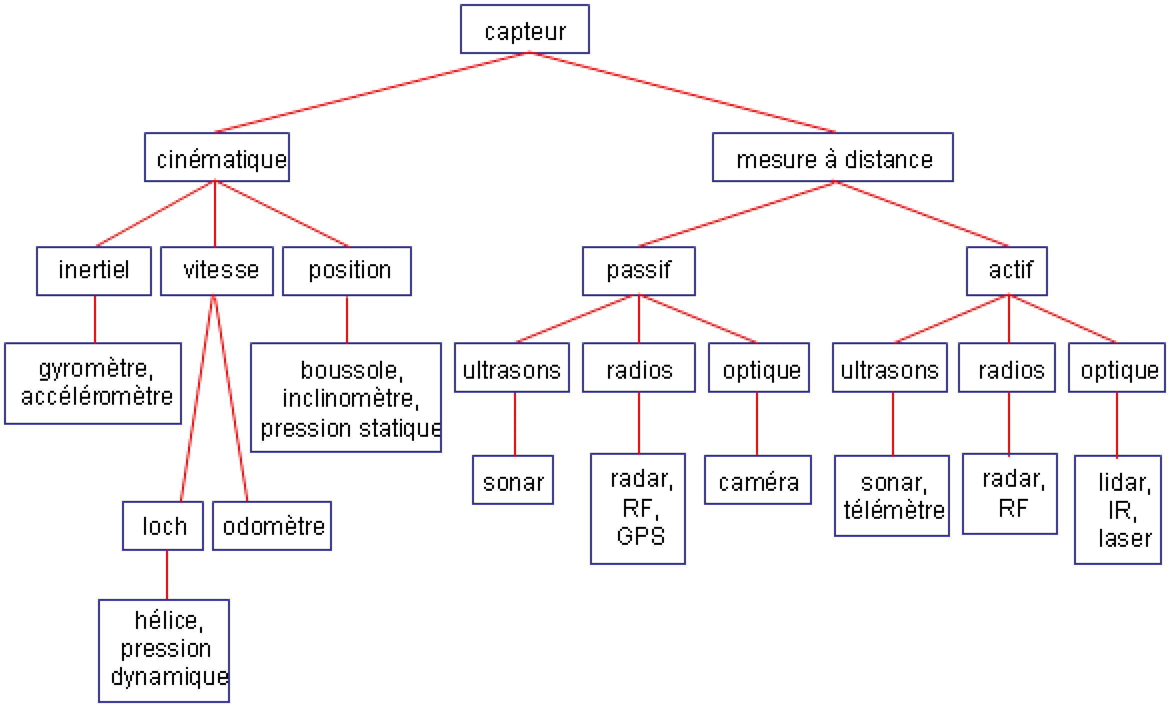
Designing a robot

- Sensors :
  - **Proprioceptive** : provide information about current robot **internal state**, w.r.t. and internal reference
  - **Exteroceptive** : provide information about robot **state w.r.t. its environment**
  - **Exproprioceptive** : combination of proprioceptive and exteroceptive

capteur	sous-marin	aérien	proprioceptif	extéroceptif	exproprioceptif
boussole	+	+	+		
magnétomètre	+	+	+		
inclinomètre	+	+	+		
gyromètre	+	+	+		
accéléromètre	+	+	+		
GPS		+		+	
IMU/AHRS	+	+	+		
INS	+	+			+
odomètre		+	+		
altimètre barométrique		+	+		
capteur de pression	+		+		
loch hélice/pression	+	+	+		
DVL	+			+	
caméra	+	+		+	
sonar	+			+	
télémètre ultrason		+		+	
télémètre infrarouge		+		+	
télémètre laser		+		+	
radar		+		+	
lidar		+		+	
récepteur radios		+		+	

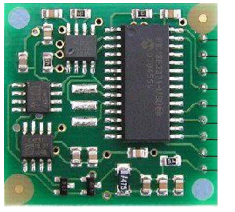
Designing a robot

- Sensors :



Designing a robot

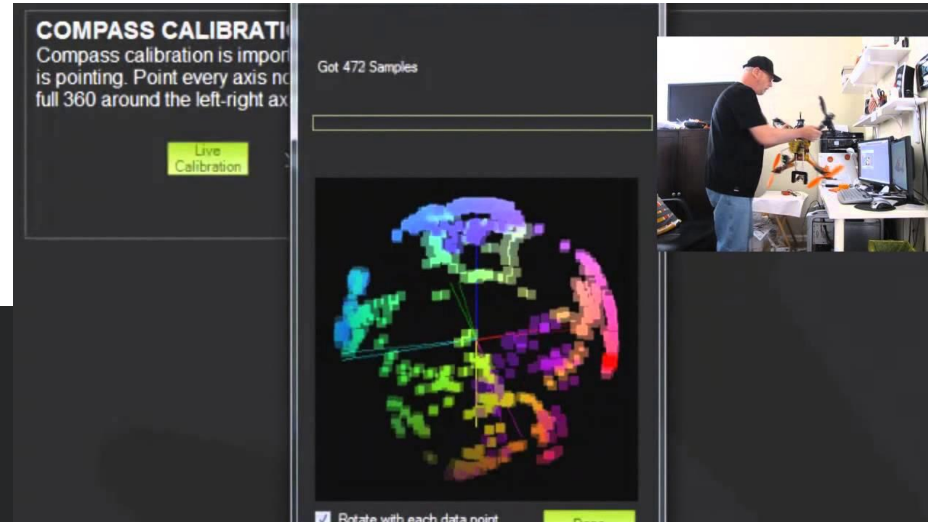
- Sensors :
  - **Compass** : angle w.r.t. magnetic North.

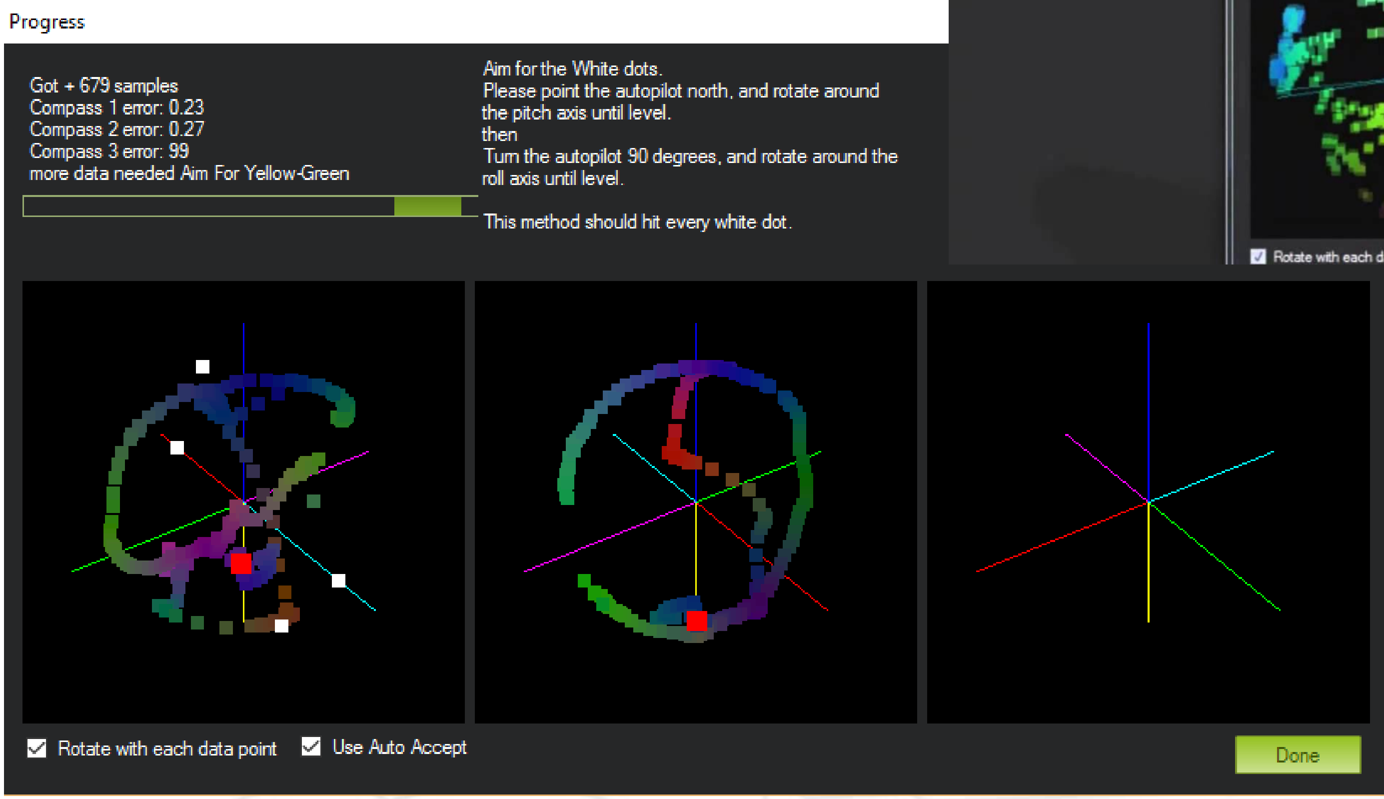


  - 3 axis **Magnetometer**/ 3D compass : measure the 3 Earth magnetic field components. Be careful to **perturbations** due to metal parts, electrical currents…
    - **Magnetic perturbations** due to **surrounding environment metal parts** : **difficult to correct** but we could make a cartography of the environment magnetic field
    - **Perturbations** due **robot parts** (may depend on motors speed…) : **constant perturbations** can be taken into account by following a **magnetic calibration procedure**

Designing a robot

- Sensors :
  - Magnetic calibration





Designing a robot

- Sensors :
  - Inclinometer : angle w.r.t. horizon
  - Gyrometer : rotating speed
  - Accelerometer : acceleration in translation
  - GPS : latitude, longitude, altitude (not precise), speed

    Do not work inside nor underwater (needs a good direct « view » of the satellites in the sky)

    Might give outliers when it is at the limit of not getting the signal or if there are multiples reflections on obstacles

    Startup time (« fix ») of several minutes depending on conditions

    Sometimes, 2 GPS can be used to estimate heading

Designing a robot

- Sensors :
  - IMU (Inertial Measurement Unit) : combination of acceleros, gyros, magnetos

  - AHRS (Attitude and Heading Reference System) : IMU with a global digital processing of the different sensors data

  - INS (Inertial Navigation System) : AHRS with the addition of positions and speeds outputs

    Often comes with an integrated GPS because getting this values only from inertial data causes an accumulation of errors dues to integrations

Designing a robot

- Sensors :
  - Check the documentation to determine if any calibration is possible/necessary
  - If necessary, a simple calibration procedure for heading measurement using a compass/IMU/AHRS can be followed : see http://www.ensta-bretagne.fr/lebars/these/thesis_2011-11-23.pdf (p64)
  - **Bad heading measurements are very common problems for autonomous robots**

Designing a robot

- Sensors :
  - Communication types :
    - Analog outputs (e.g. 0..5 V)
    - Digital outputs :
      - P2P :
        - 0 or 1 on one or several bits in parallel
        - RS232/UART serial link
      - Bus :
        - RS485/RS422
        - CAN
        - I2C/SMBus
        - SPI
        - USB
      - Network :
        - Ethernet, Wi-Fi

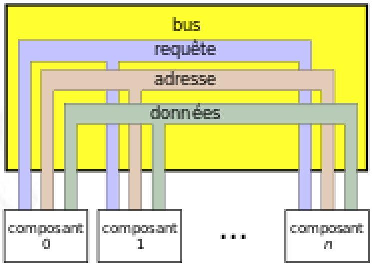
Designing a robot

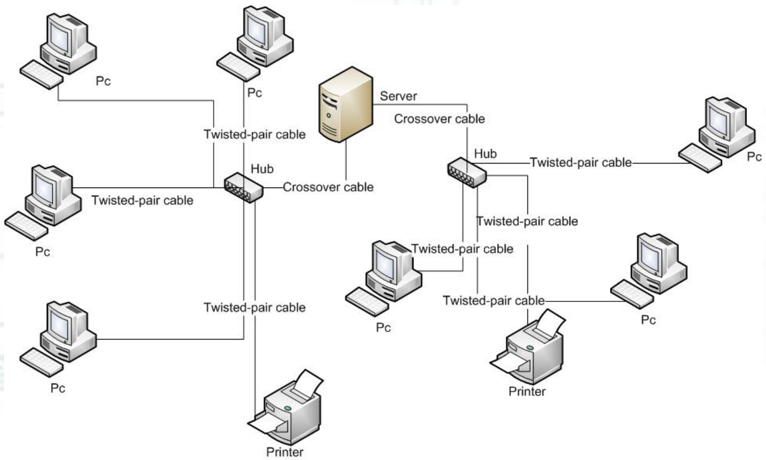
- Sensors :
  - Communication types :

    Point to point link (P2P) : 2 devices are in exclusive communication

    Bus : communication channel shared between several devices, often with concepts masters and slaves

    Network : several devices and several channels



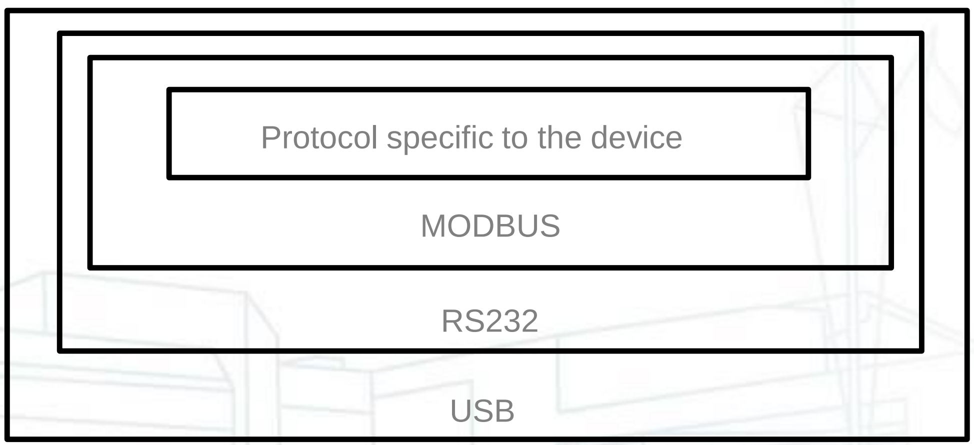


Designing a robot

- Sensors :
  - Communication types :

    Often enclosing layers of links types, protocols, see OSI model...

    Example : Keller P33x pressure sensor



Use as much as possible official software provided with sensors/actuators to first check the behavior in the nominal conditions defined by the manufacturer!

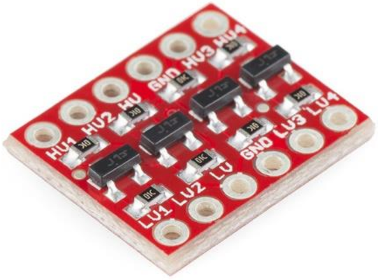
Designing a robot

- Sensors :
  - Communication types :

    Warning : the max sensor power supply voltage might be different from the max voltage on its communication pins!

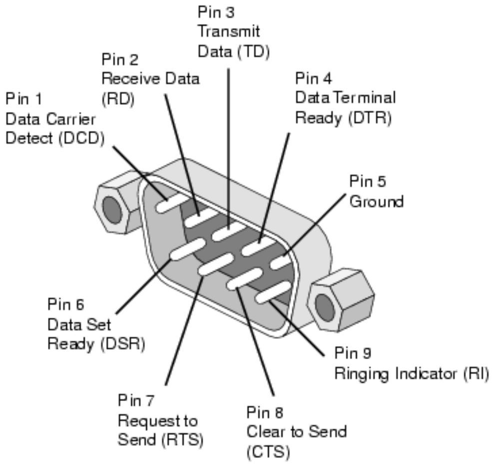
    Example : several sensors are powered by 5 V while their UART has voltage levels between 0 and 3.3 V

    See https://learn.sparkfun.com/tutorials/bi-directional-logic-level-converter-hookup-guide



Designing a robot

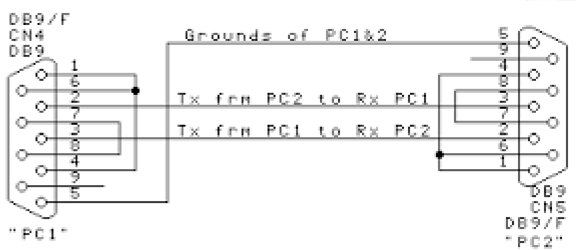
- Sensors :
  - Communication types :
    - RS232/UART serial link
      - Originally, DB9 port directly on the PC
      - Now, through USB converters
      - Standard PC voltage levels : -12..+12 V
      - On embedded boards and small sensors : often TTL 0..5 V or 3.3V levels, to check
      - USB-RS232 PC adapters : FTDI, Prolific
      - USB-RS232 TTL adapters : FTDI
      - RS232 PC-RS232 TTL adapters : MAX232



See also https://learn.sparkfun.com/tutorials/terminal-basics

Designing a robot

- Sensors :
  - Communication types :
    - RS232/UART serial link
      - Main signals : **GND, TX (Transmit), RX (Receive)**
      - **The TX from the PC should be put on the RX from the device, and conversely!**



      - If the sensor only send data (e.g. GPS), it might not be necessary to connect its RX, same for actuators with TX
      - For device firmware update, RTS, CTS, DTR, etc. might be also necessary
      - Conversely, sometimes these signals might cause problems and should be left disconnected…

Designing a robot

- Sensors :
  - Communication types :
    - RS232/UART serial link
      - Depending on the device protocol, it might be possible to receive/send easily data using **HyperTerminal** (Windows, see http://www.ensta-bretagne.fr/lebars/Share/hypertrm.zip) or **minicom**, **gtkterm** (Linux)
      - See also : http://www.ensta-bretagne.fr/lebars/tutorials/Serial%20port%20C.pdf

Designing a robot

- Sensors :
  - Communication types :
    - Bus RS485/RS422
      - Signals : **A+, A-, B+, B-**, sometimes called also RX+, RX-, TX+, TX-
      - No GND since voltage levels are measured between the – and + : called symmetric/differential signal, contrary to RS232 where it is called asymmetric since the reference is GND for TX and RX
      - **Differential signal** => more robust to interferences
      - **Full-duplex** : has a separated reception and transmission line (needs **4 wires**)
      - **Half-duplex** : only use the same line for reception and transmission, therefore must commute between transmitter and receiver mode each time (needs **2 wires**)

Designing a robot

- Sensors :
  - Communication types :

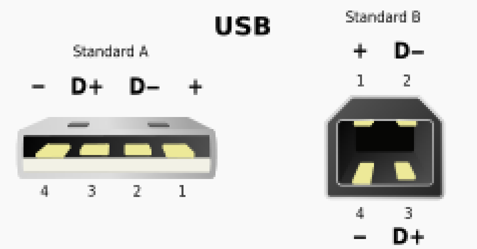
    USB 2.0 bus

    Signals : **GND, D+, D-, +5 V**

    High variety of **different connectors**, to check

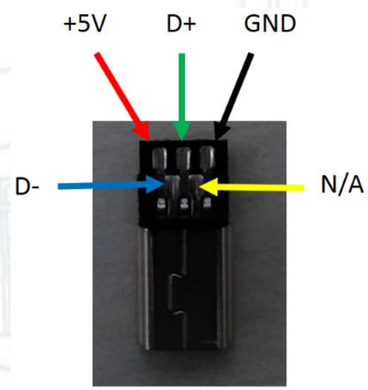
    Use **high quality cables** (shielded, etc.), avoid cutting the cables or pass through non-standard connectors, this type of signal is highly fragile in practice…

Pin out

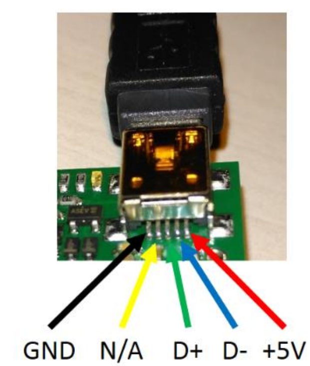


The type-A plug (left) and type-B plug (right)

Pin 1	V_{BUS} (+5 V)
Pin 2	Data−
Pin 3	Data+
Pin 4	Ground



Adafruit Slim MiniB DIY #1825



Mini USB port on
Pololu Micro Maestro 6

Designing a robot

- Sensors :
  - Communication types :
    - USB 2.0 bus
      - **USB host** vs **USB device** : PC=host, sensor=device in general
      - Smartphones/tablets : sometimes can support both modes
      - Power supply of a device :
        - In general provided by host
        - If consuption<100 mA no problem in general
        - Otherwise, a negociation happens between host and device to obtain up to 500 mA
        - A non-powered USB hub cannot provide more than 100 mA on each of its ports

Designing a robot

- Sensors :
  - Communication types :

    USB 2.0 bus

    Plugging a device asking for excessive power can damage the computer!

    Warning : PCs often already have integrated hubs, this might also have consequences on expected max bandwidth (480 Mbits/s for a « real » USB 2.0 port)!

    Warning : USB 3.0 to USB 2.0 backward compatibility do not always work well, try to use USB 2.0 ports for USB 2.0 devices when possible…

Designing a robot

- Sensors :
  - Communication types :

    USB bus

    It is a **complex bus** with its version, bandwidth, power supply management, etc. => we never directly need to code with this protocol, there should be always a more simple protocol on top and **drivers and specific libraries** to use (when it is not just a conversion to RS232, RS485)…

    e.g. : Webcams (with OpenCV), Labjack UE9 (with libUSB, see manufacturer), Kinect 2 (with Kinect SDK)

Designing a robot

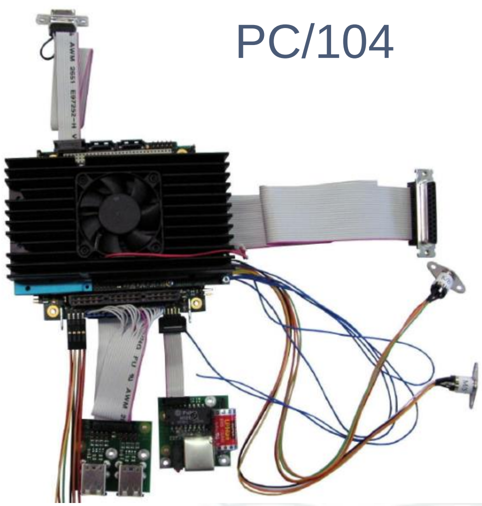
- Sensors :
  - Communication types :
    - I2C/SMBus
      - 3 signals: **GND, SDA (data) and SCL (clock)**
      - On Linux, a driver, commands, and an API enable to use this bus if available on the computer

Designing a robot

- Embedded computer :
  - Robot intelligence
    - Contains programs defining robot behavior
    - In general, all sensors, actuators, communication devices are directly or indirectly connected on it
    - Used to save the data, even if it is often safer to send them also to the control station in case of the loss of the robot…
    - If not powerful enough, we can try to make some computations on the control station
  - « Computer », « PC », « brain », « intelligence » are often used as a synonym…

Designing a robot

- Embedded computer :
  - Examples : PC i.e. computer with Intel x86 compatible processor with BIOS/UEFI, etc.
    - HTPC (Home Theater PC)
    - EeePC 901 (netbook)
    - Mini ITX
    - PC/104



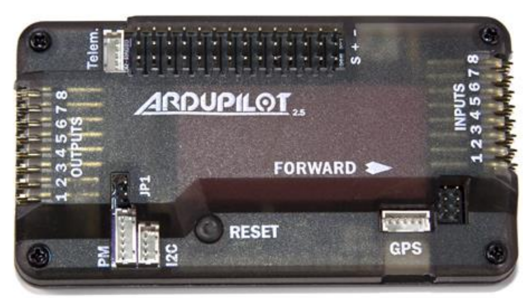
Computer form factors	
Name	Size (mm)
NUC	116.6 x 112 x 34.5
Compute Stick	103.3 x 12.5 x 37.6
Zotac Pico	66 x 19.2 x 115.2
eeePC 901	226 × 175.3 × 22.9
Mini TX	170 × 170
Nano ITX	120 × 120
Pico ITX	100 × 72
PC/104	96 × 90

Designing a robot

- Embedded computer :
  - Examples : computers with ARM processor
    - Smartphone / tablet
      - Smartphone Samsung Galaxy S with Android (with GPS, compass, camera, Wi-Fi, etc. already integrated)
    - Raspberry Pi

Designing a robot

- Embedded computer :
  - Other : Arduino, autopilots (some have often already PWM outputs, etc.)…



Designing a robot

- Embedded computer :
  - **Warning** : even if computers and other embedded boards are more and more small and cheap, take into account the **size and price of the connectors** and potentially necessary **adapters**!

    e.g. Raspberry Pi Zero (5$) + hub USB (5$) + micro USB adapter (1$) + mini HDMI adapter (5$) + Wi-Fi dongle (5$) + keyboard/mouse (5$)…

Designing a robot

- Communication :
  - Actuators control

    RC receiver on the robot, remote for the operator

    No need for a PC, designed to be directly connected to the ESC and/or servos

    Typ. 4-8 PWM

    Usual frequencies : nowadays 2.4 GHz, 5.8 GHz, 433 MHz, before 27, 35, 40, 41 MHz

    Various modulation/coding type (FM, AM, SSS, SPCM, PPM), various channels around specific frequencies (check quartz if any)

Designing a robot

- Communication :
  - Telemetry : orientation, position, speed, battery, etc. info
    - Autopilot and/or specific module on the robot, specific module for the operator PC or other
    - Usual frequencies : 2.4 GHz, 5.8 GHz, 433 MHz

  - FPV (First Person View) : remote camera
    - Camera with specific module on the robot, glasses or screen for the operator
    - Usual frequencies : 2.4 GHz, 5.8 GHz

Designing a robot

- Communication :
  - Other : bidirectional communications

Wi-Fi

3G

XBee

Satellite (e.g. Iridium)

AIS (on boats)

Acoustic modems (submarines)

Designing a robot

- Communication :
  - **Range** vs **bandwidth** vs **transmit power** vs **frequency** vs **antenna size**
    - The **higher** the **frequency**, the **more** with can expect to get a high **bandwidth**
    - The **lower** the **frequency**, the **more** with can expect to get a **long range** with a **low transmit power** (i.e. low power requirement)
    - Antenna size decrease when frequency increase
    - Use directional antennas to maximize range, however, need an accurate control towards the right direction

Designing a robot

- Communication :
  - Perturbations :
    - Even an unused antenna might cause perturbations
    - A powerful Wi-Fi can perturbate a GPS, even if they are not using the same frequency…
    - Some obstacle types (metal fence) can perturbate significantly
    - Antennas should be as far as possible from the ground…
    - Wi-Fi channels : it can be sometimes better to be on the **same channel** as others, since **protocols are designed to handle this situation**, while if we are on different but close channels unexpected radio interferences not related to the protocol will occur…

Designing a robot

- Communication :
  - Air vs water :

    Electromagnetic waves do not propagate much underwater :

    In **still water**, we can expect to communicate on **a few m** at frequencies of **tens of MHz** while in sea water, there is more attenuation

    Similarly, visibility is limited underwater, therefore optical communications are also limited :

    In general we detect better yellow or orange objects since they contrast with the water color

    However, it is more the blue-green color that is absorbed the least in the water, unfortunately it does not contrast with the water color

Designing a robot

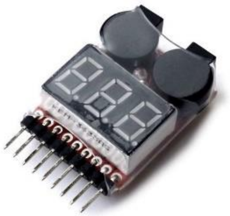
- Energy :
  - Usual battery types : **Li-Po**, Li-Ion, **NiMH**, Pb
  - Energy (in Wh) = Voltage (in V) * Capacity (in Ah or often mAh, often noted C)
    - It is this value that should be used to compare 2 batteries
  - Max charge/discharge current
    - e.g. 30C/5C : if e.g. C=2500 mAh, the max discharge current would be 30*2500 mAh and max charge current would be 5*2500 mAh

Designing a robot

- Energy :
  - Cells : 3S2P => 2 parallel blocks of 3 cells in serial (S for serial, P for parallel)
  - Cell voltage depends on battery type

    e.g. 3.7 V for Li-Po, 1.2 V for NiMH

    Warning : if the voltage of a cell becomes too low (e.g. < 3.5 V for Li-Po), it can be forever! => Use a checker/alarm!



    Always unplug it after use since it always draw a little bit of energy!

  - See http://www.ensta-bretagne.fr/lebars/robots_batteries.pdf

Designing a robot

- Energy :
  - Most of the batteries must be charged with **intelligent chargers** that follows specific current/voltage charge curves that depend on battery type and its state

    https://youtu.be/vzAOG9ctZRU
  - **Voltmeter** or checker : **can prove** that a battery is **discharged**, but not **reliable** to say that it is **charged**!
  - In particular, **Li-Po** built from several cells usually have a **balance connector** that enables the charger to adjust the charge of each cell independently
  - Mechanical stress, overcharge, short-circuit = danger!

    https://www.youtube.com/watch?v=gz3hCqjk4yc

    https://www.youtube.com/watch?v=-DcpANRFrI4

Designing a robot

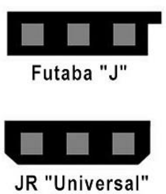
Designing a robot

- Typical RC connectors :
  - Power :

T/Deans

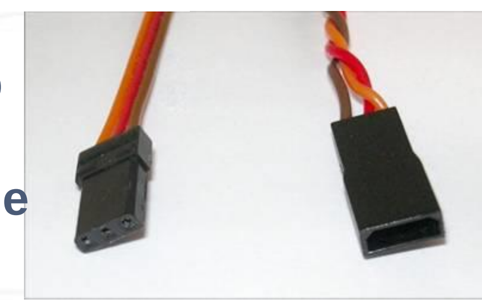
XT60

PK3.5 (designed for brushless motors)



  - Control or low power :

Servo (warning, sometimes different brand types do not connect well) : **3 wires** with in general **black/brown GND**, **red +5V**, **white/yellow/orange PWM signal**



BEC JST : 2 wires, GND and often +5V (to check)

Designing a robot

- Typical RC connectors :
  - Usual conventions :

    http://www.ensta-bretagne.fr/lebars/conventions_connecteurs.pdf

  - See also :

    http://www.dronetrest.com/t/wires-connectors-and-current-what-you-need-to-know-as-a-drone-builder/1342

Designing a robot

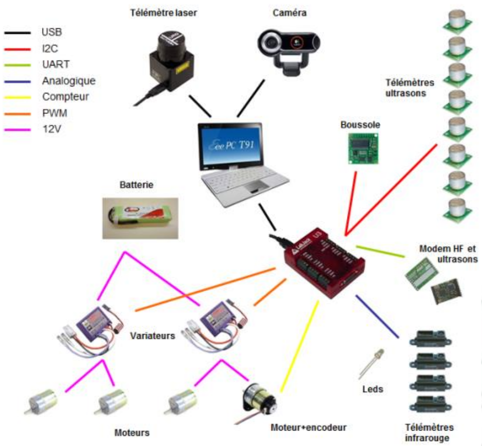
- Cables :
  - Check the **cables diameter** and the **connectors quality** when the **currents** are **high** (e.g. > 5 A)
  - Voltage decreases with the cable length (e.g. -1 V every 10 m depending on cable resistance, current, etc.)
  - If the currents are high, **magnetic perturbations** may occur for the robot compass => **twisting cables** can sometimes limit those perturbations
  - Some digital signals should be also carried using **twisted pairs** : e.g. Ethernet, RS232,…
  - For **antennas cables extensions**, **coaxial cables and connectors** are necessary, check what are their max **frequency and impedance**
  - The **ground cables set-up** should be **in star** rather than with loops…

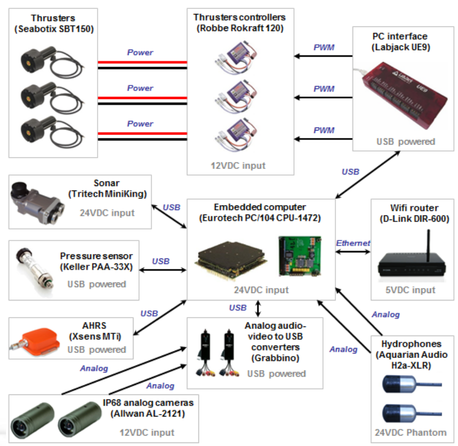
Designing a robot

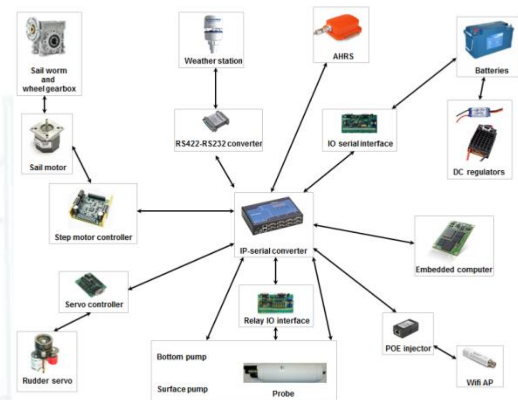
- Manipulations, interventions, tests :
  - **Unplug the battery** for any **technical intervention**!
  - **Only plug the minimum** when you are not sure of what you are doing/or about the hardware state **to limit damages** in case of a problem!
  - Check if your workspace contains **metal objets**: **short-circuits can occur**!
  - **Take pictures and videos** at each step of any mounting/unmounting/test
  - Make a user, test, etc. checklist.

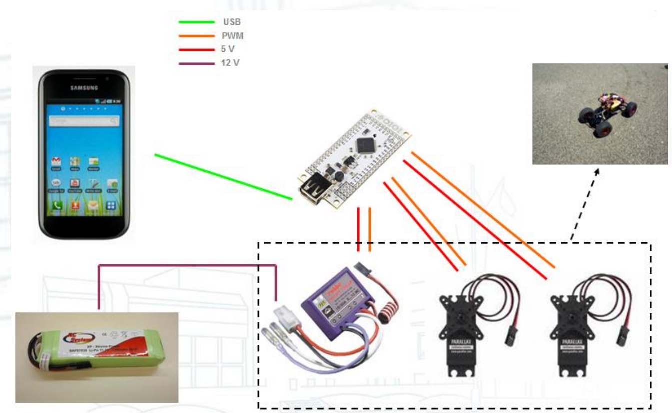
Designing a robot

- Design examples for different types of robots

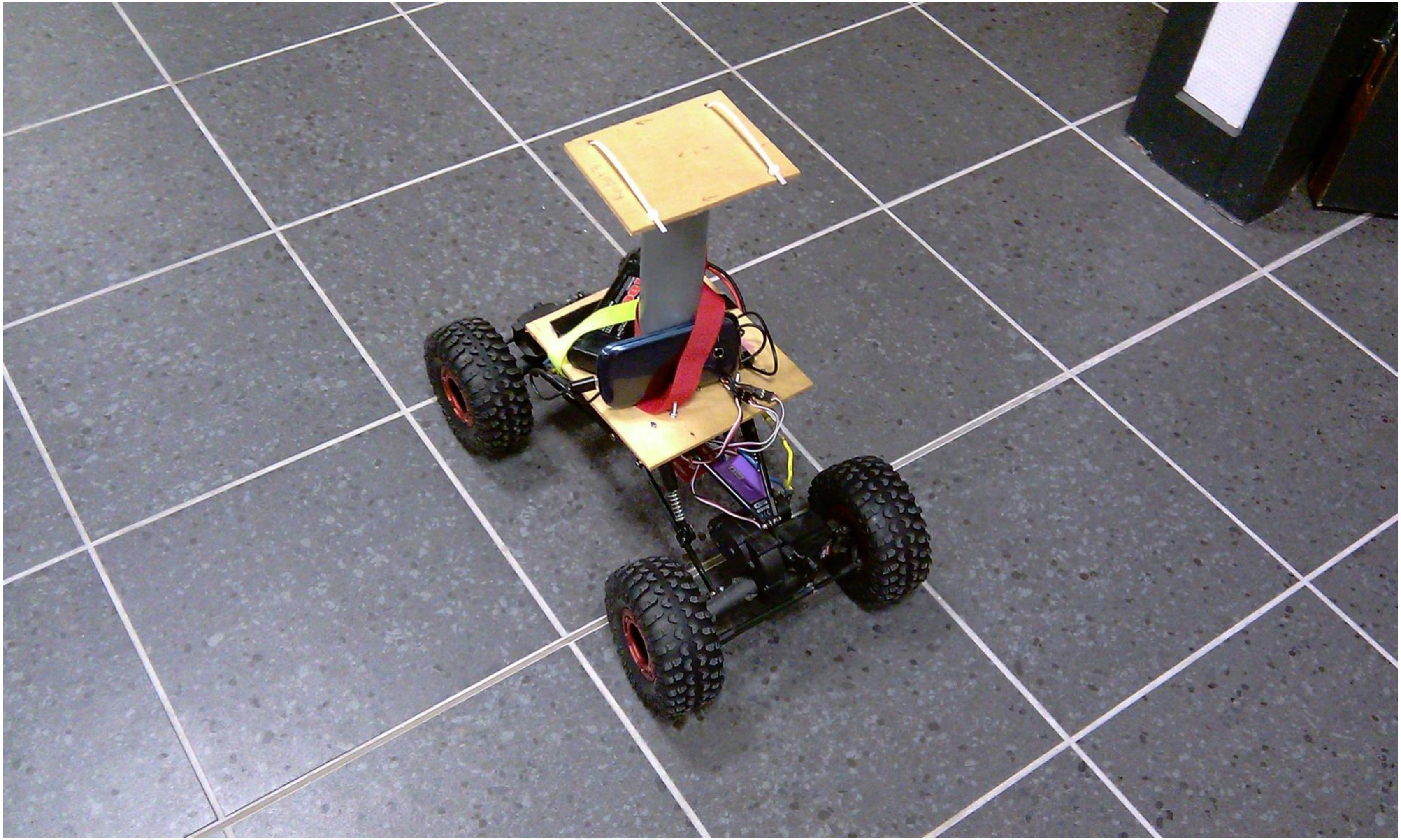




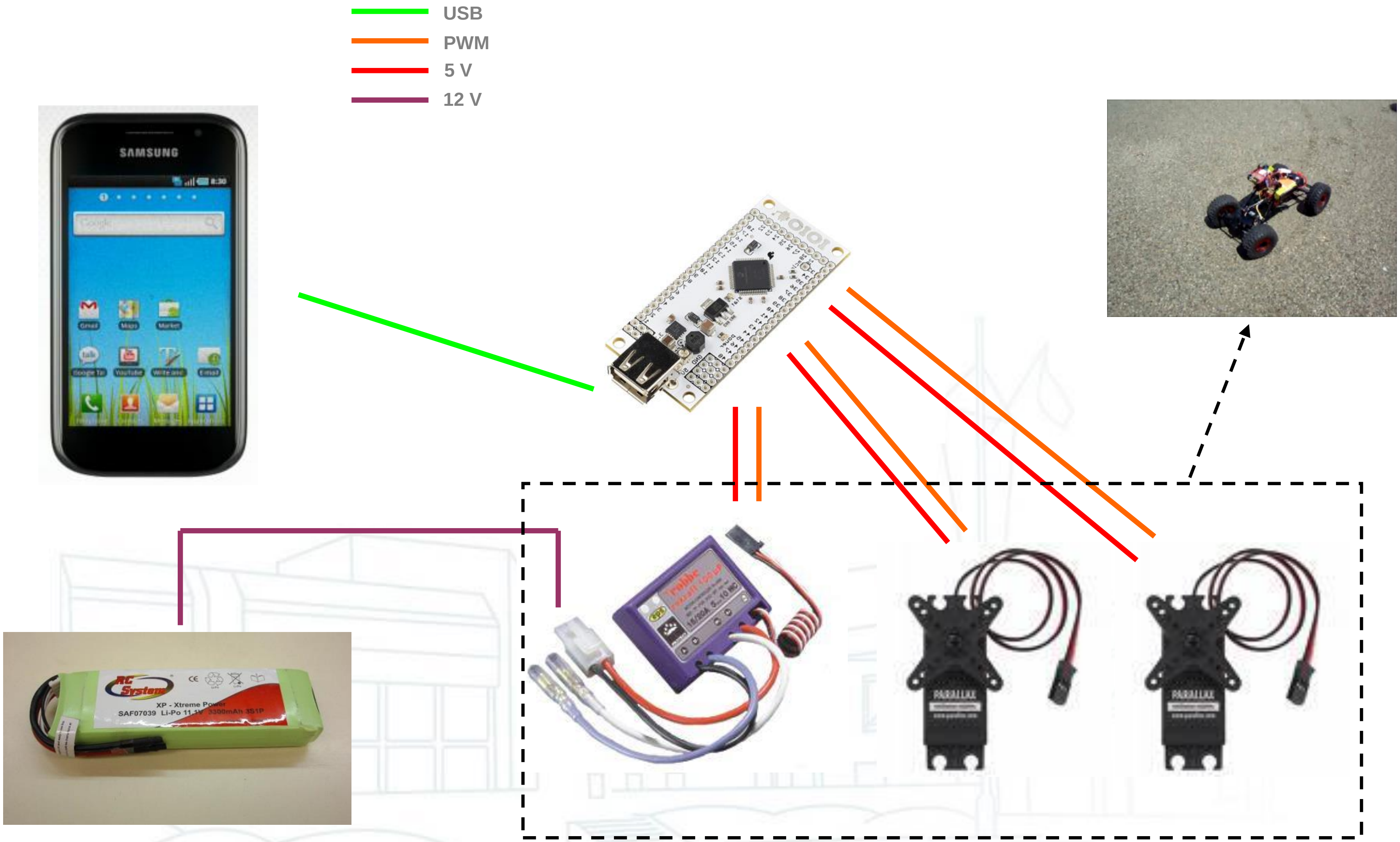




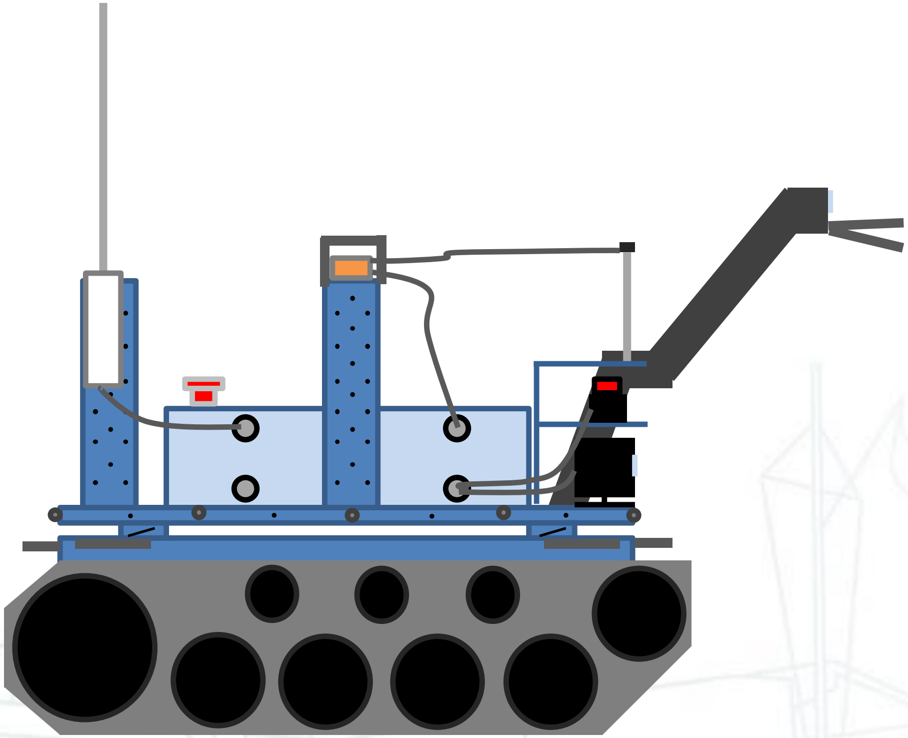
AGV Buggy (2012)



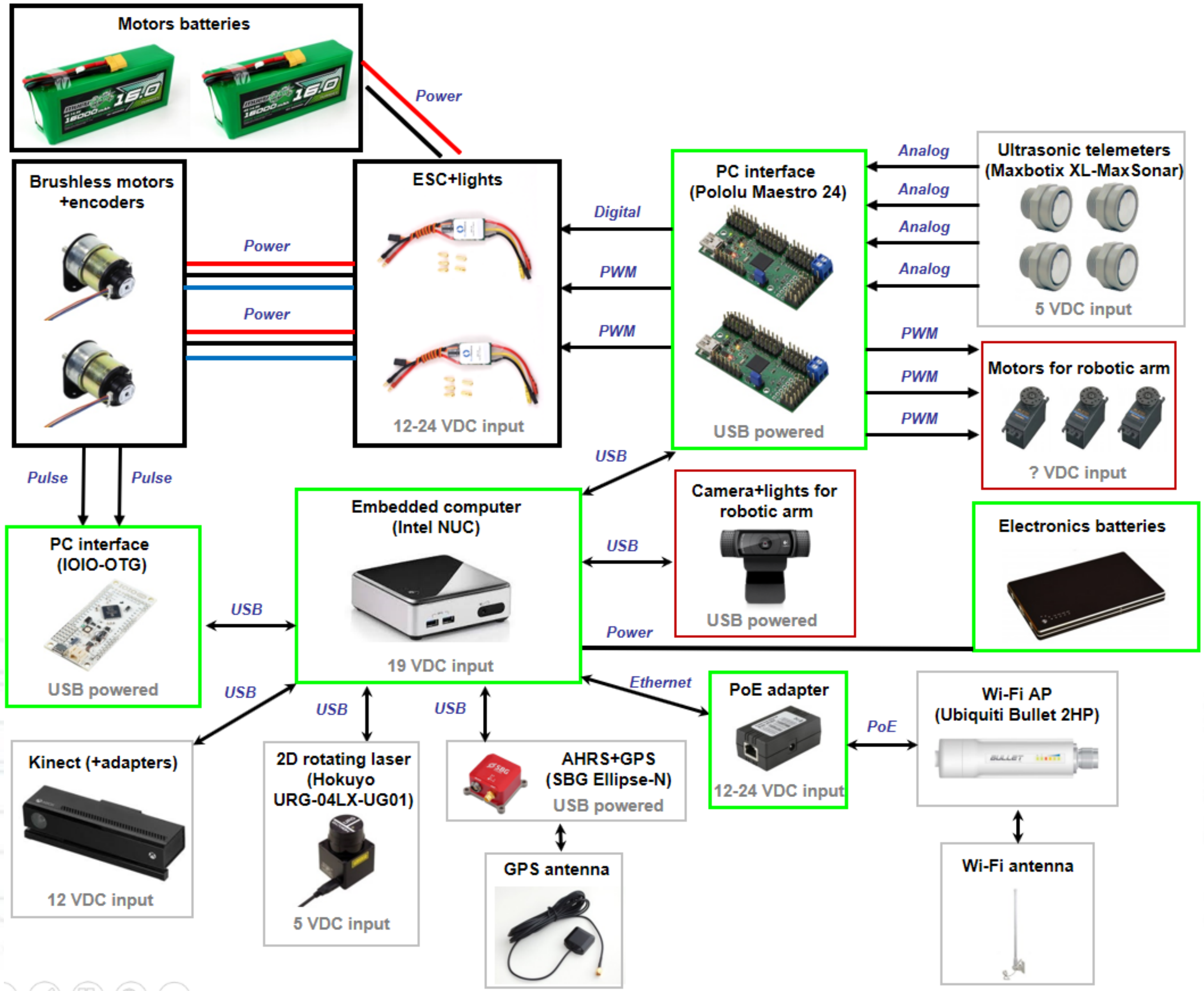
AGV Buggy (2012)



AGV euRathlon (future)

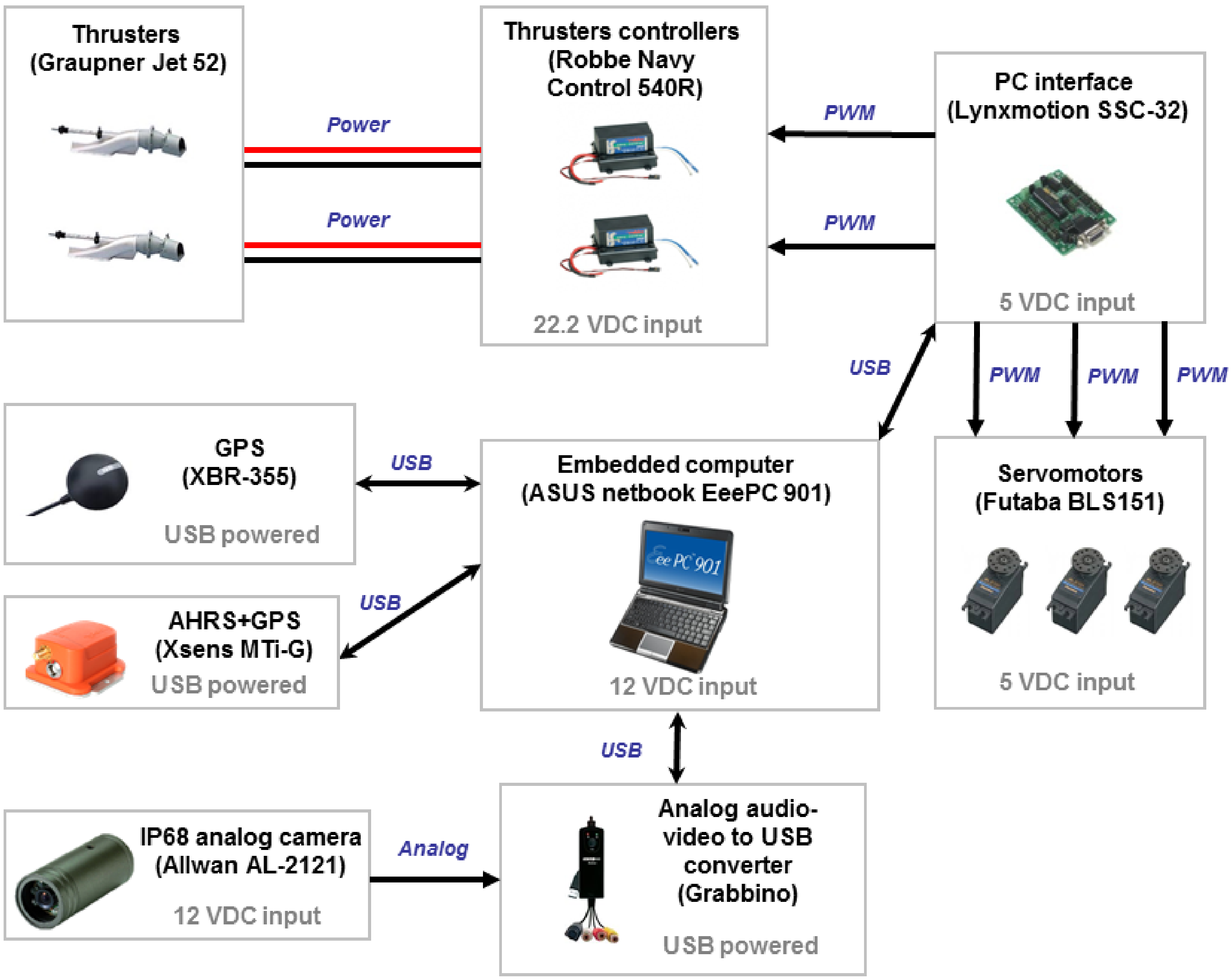


AGV euRathlon (future)



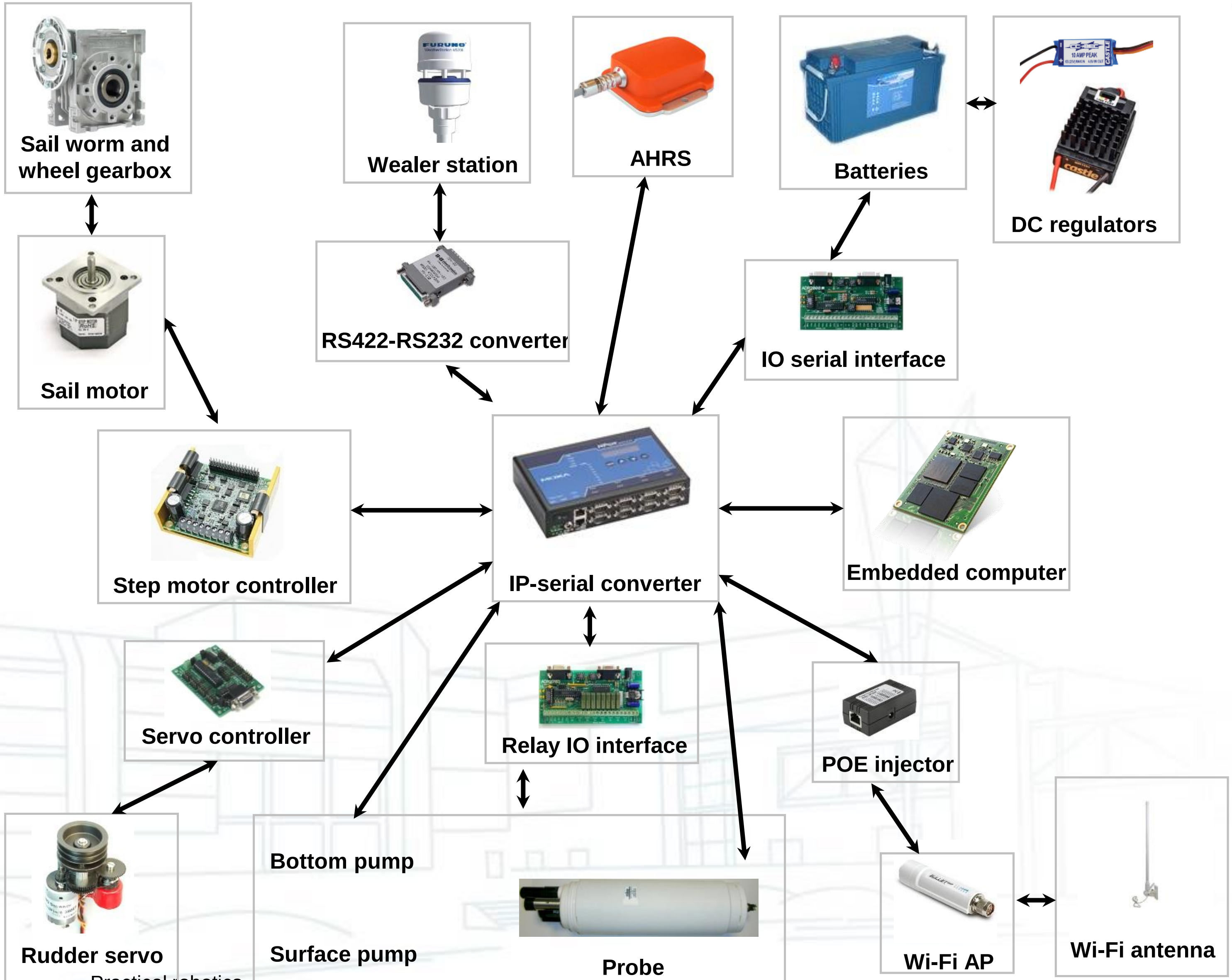
ASV Motorboat (2013)

ASV Motorboat (2013)



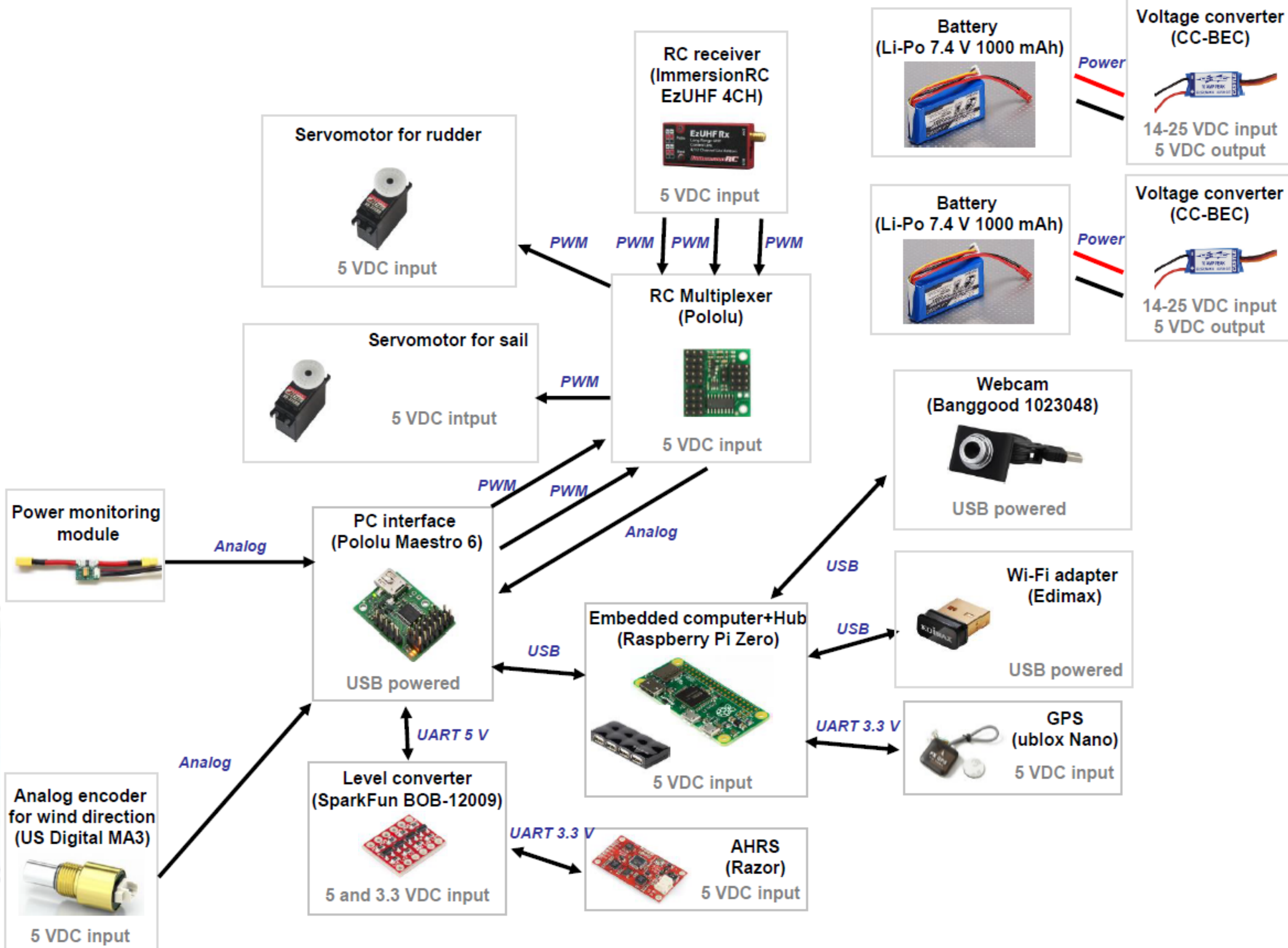
ASV VAIMOS (2011)

ASV VAIMOS (2011)



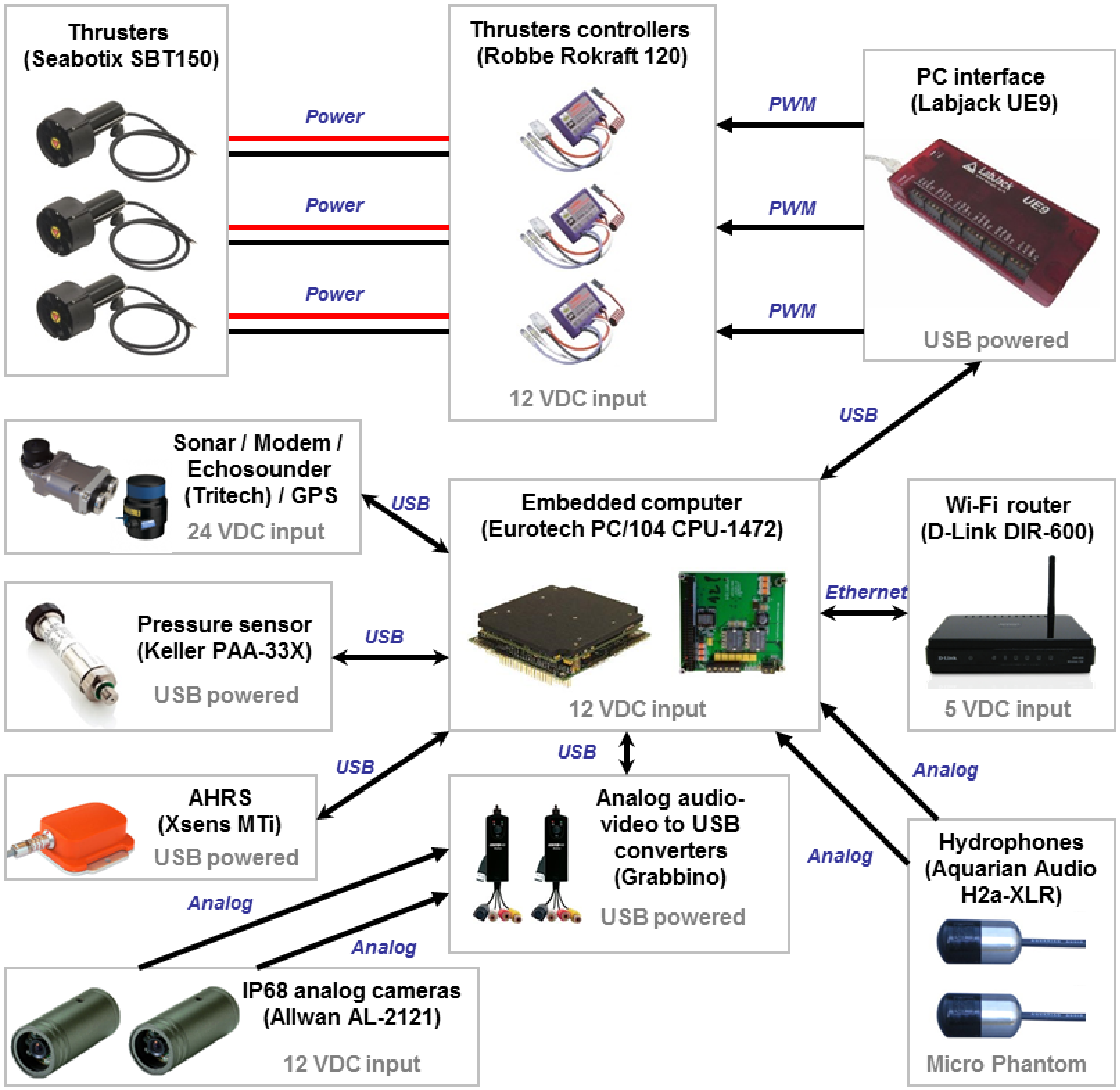
ASV White Sailboat (2016)

ASV White Sailboat (2016)

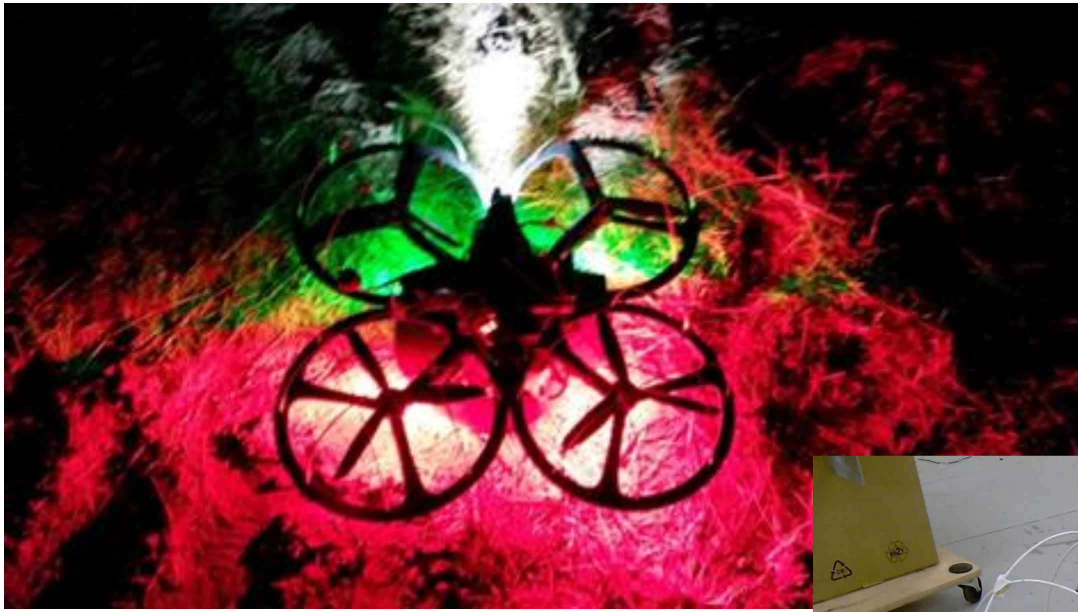


AUV SAUC'ISSE (2014)

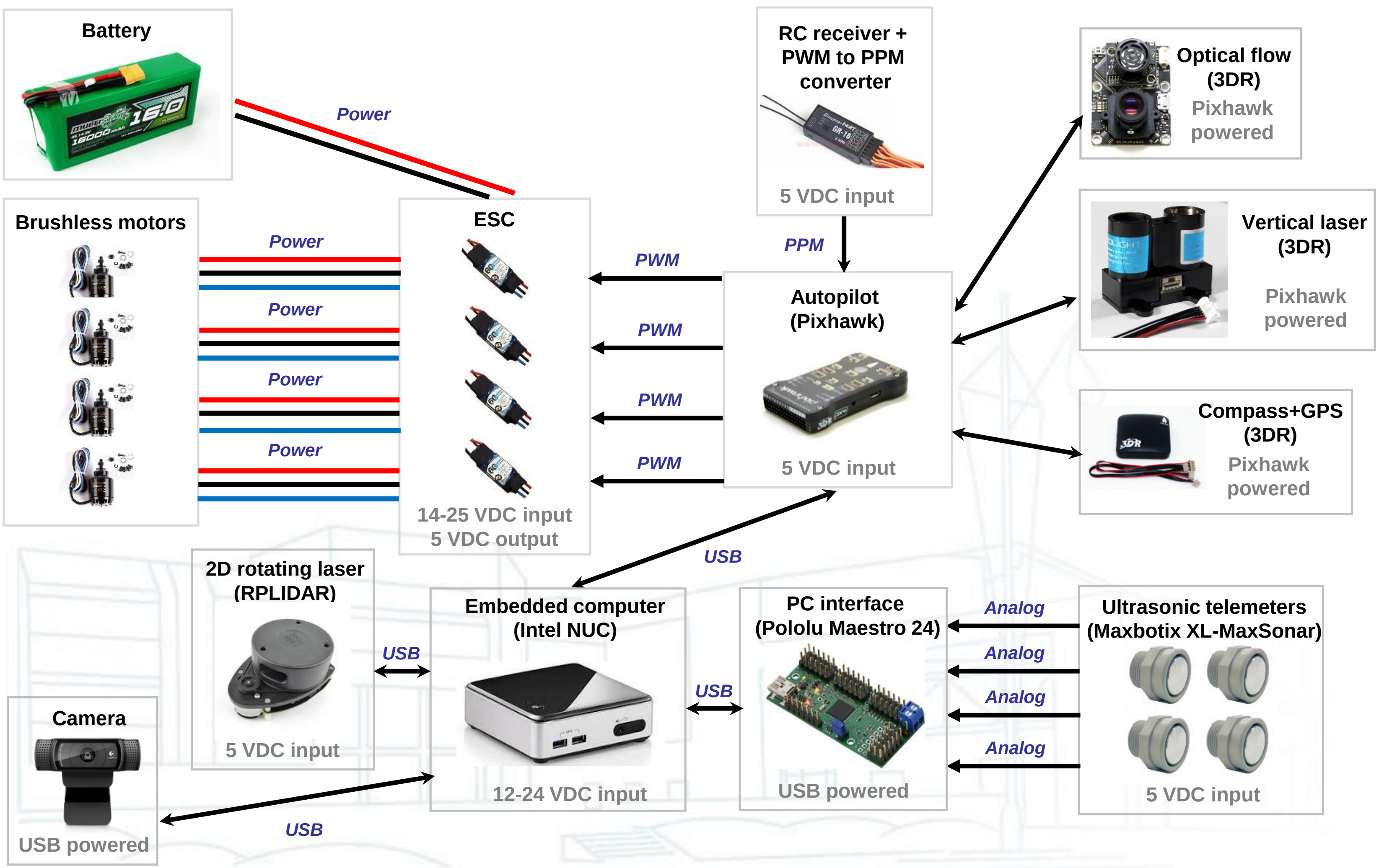
AUV SAUC'ISSE (2014)



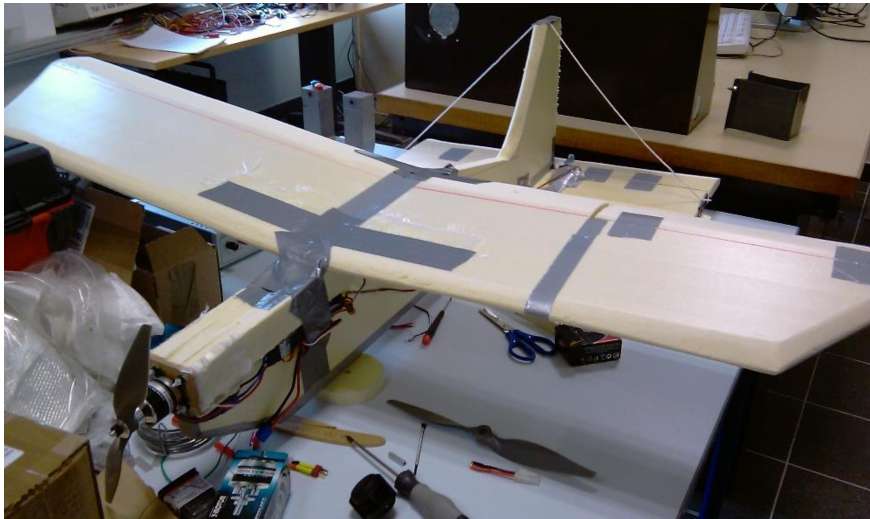
Indoor Quadrotor euRathlon (future)



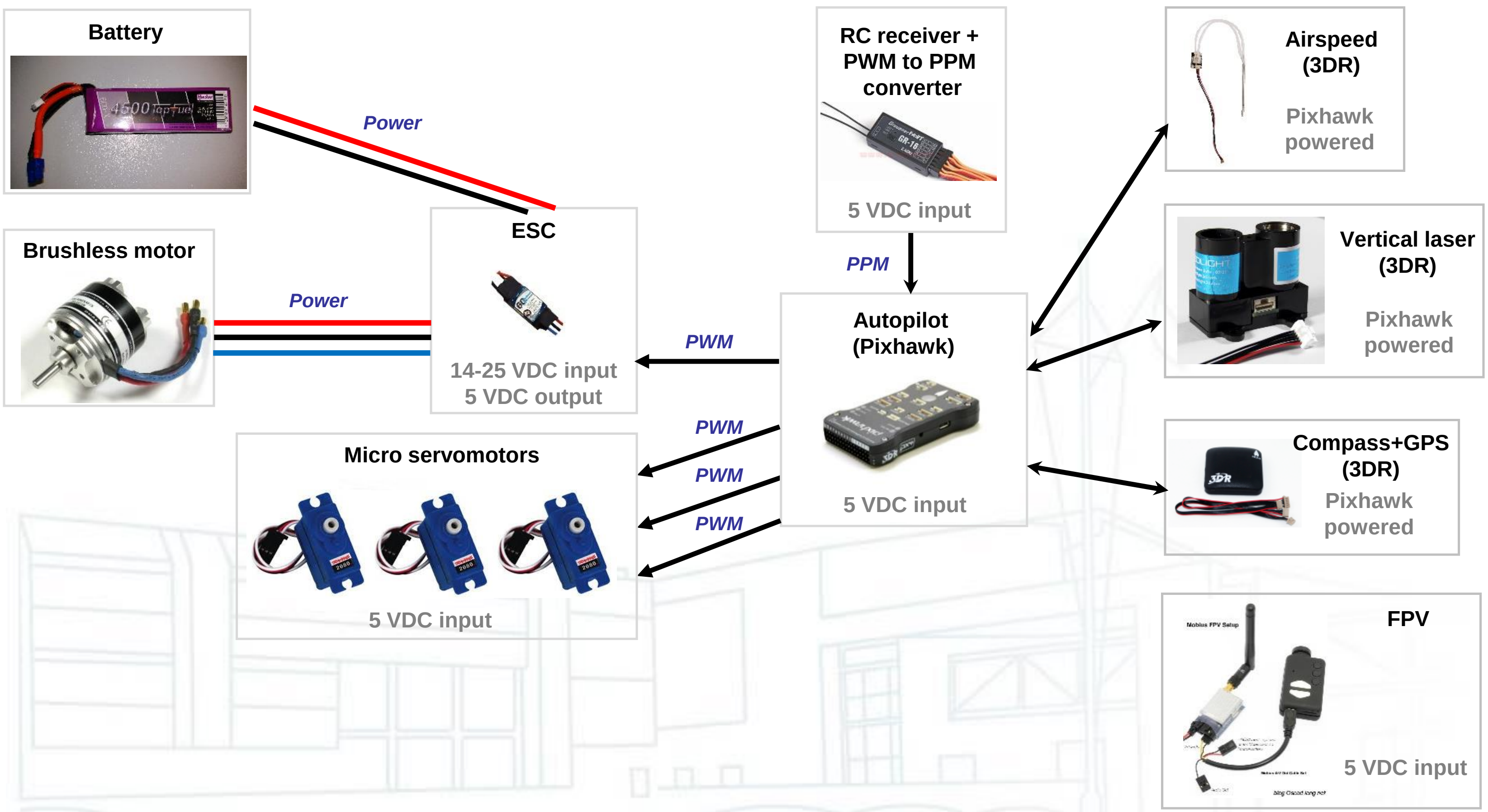
Indoor Quadrotor euRathlon (future)



Plane (future)



Plane (future)



Communication and remote control of autonomous computers

Communication and remote control of autonomous computers

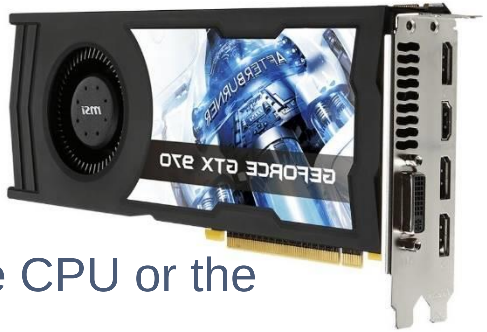
- Goal :
  - Learn how to setup a computer and the necessary network devices to be able to control it remotely when it will be installed inside a robot

- Building a PC :
  - A PC usually has :
    - A motherboard : board where all the other components are connected

    - A processor (CPU)

    - Physical memory (RAM)

    - SSD, flash memory, hard disk



- Building a PC :
  - A PC usually has :

    A display adapter (GPU) : often integrated inside the CPU or the motherboard

    An audio card : often integrated inside the motherboard

    Ethernet, Wi-Fi network adapters

    Passive or active cooling devices : on CPU, GPU

    Power supply unit (PSU): e.g. ATX format

    Box : contains everything and has buttons (Power, Reset), LEDs (power status, disk activity) and additional ports (USB, audio)

Communication and remote control of autonomous computers

- Building a PC :
  - Typical internal buses and connectors :
    - PCI Express : very fast data bus designed for optional extension boards (video, audio, Wi-Fi, LAN/Ethernet), replace PCI and AGP)
    - SATA : hard disks, SSD, DVD drives, replace ATA/PATA/IDE
    - ExpressCard : very fast data bus designed for laptops optional extension boards, replace PCMCIA
    - SMBus : for voltage, current, temperature sensors
  - To check the compatibility of various devices, see https://pcpartpicker.com/

Communication and remote control of autonomous computers

- Building a PC :
  - Efficient search for PC, robot, etc. parts :

    Example : powerful display adapter for simulations or sensors data processing (e.g. stereo camera ZED)

    Google search “powerful display adapter" gives for example in the 1st results "ASUS GeForce GTX 1070 - TURBO-GTX1070-8G display adapter" on amazon.com with only price indication and without technical specifications

    => Search on manufacturer website (NVIDIA) official documentation and compare with product line, search again on Google for other distributors and use their selection functions

Communication and remote control of autonomous computers

- Building a PC :
  - Efficient search for PC, robot, etc. parts :

    Distinction between **main component manufacturer** (NVIDIA), its **model** (GTX 1070), **full board manufacturer** (ASUS), its **model** (if several versions available) and the **distributor/reseller** (amazon.fr)

- Building a PC :
  - Efficient search for PC, robot, etc. parts :

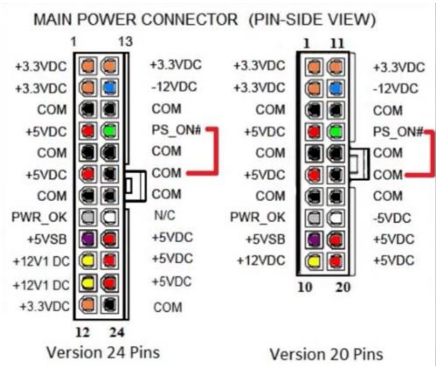
    Search also on Google for tests and reviews on specific websites, customers comments, some distributors make also their own independent tests (warning, sometimes comments can be biaised or fake!)

    Whenever your choice is made, it can be good to search on Google for "ASUS GeForce GTX 1070 - TURBO-GTX1070-8G problem" to check typical problems reported by users and evaluate the risk to be impacted

Communication and remote control of autonomous computers

- Building a PC :
  - Hardware unit tests :

    An ATX PSU can be tested using this procedure :



Motherboard : need PSU+Power button from the box (you can replace the Power button by making any temporary contact between its corresponding pins). If we add keyboard, display adapter (if not already integrated) and screen we can often start to get a limited interaction

Communication and remote control of autonomous computers

- Building a PC :
  - Hardware unit tests :

    Adequate cooling devices are mandatory as soon as we add the processor : overheat risk!

    As soon as we add the RAM, we should be able to configure the majority of the BIOS/UEFI

Communication and remote control of autonomous computers

- PC setup :
  - BIOS/UEFI :
    - Motherboard configuration tool, integrated inside
    - If the BIOS/UEFI do not work any more (e.g. due to wrong settings or a failed update) :
      - Push the Power button during 10 s than unplug 5-10 min
      - Restart 5 times (to make the motherboard understand that 5 startups failed in a row, this can sometimes trigger a reset to default parameters)
      - Invert during 10 s the CCMOS jumper on the motherboard (with power cord unplugged)
      - Remove/replace the motherboard Lithium battery if any (with power cord unplugged)
      - Check the motherboard documentation if none of these solutions work (e.g. special keys combinations, etc.)
      - RMA

=> Be careful for any BIOS/UEFI modification!

Communication and remote control of autonomous computers

- PC setup :
  - BIOS/UEFI :

    Typical access keys :

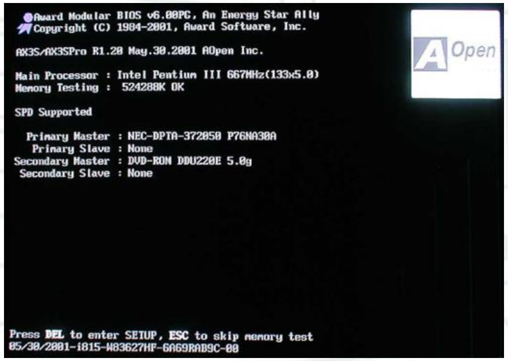
    DEL, F2 Enter Setup : main settings

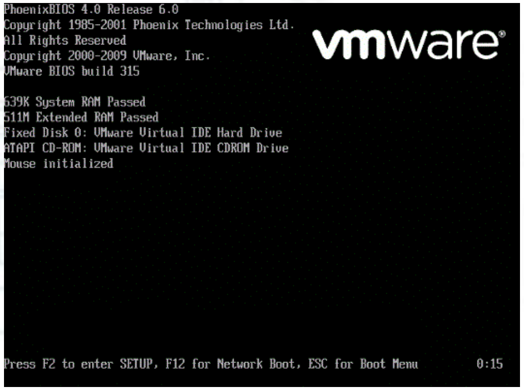
    F7 Update BIOS : update from a file on a USB key

    ESC, F8, F10, F11, F12 Boot Menu Override : start from a temporary device

    F12 Network Boot

    CTRL+I : configure a RAID (multiples hard disks in parallel)





Communication and remote control of autonomous computers

- PC setup :
  - BIOS/UEFI :

    Parameters to check :

    Load optimized settings : often a good starting point

    Date, time : might perturbate the OS if incorrect (put UTC), need a motherboard with battery or supercapacity

    Halt on no error : e.g. the robot PC should not consider that the absence of the keyboard is an error

    APM\AC loss auto restart, Power on if power failure : on a robot, we do not have always access to the PC Power button, it should start whenever it gets power supply

Communication and remote control of autonomous computers

- PC setup :
  - BIOS/UEFI :

    Parameters to check :

    Boot order : put in first the device containing the OS to ensure that the computer do not try to start on a USB key temporary connected when the computer starts, additionally a specific key enables in general to start from a temporary device if needed

    WOL (Wake On Lan) : might be used to remotely start a computer

    Watchdog : to disable since it might try to restore default parameters in case of unexpected power loss, which in practice often happens when a robot is working on batteries, the drawback will be that this failsafe function will not help if one day we set wrong parameters in the BIOS…

Communication and remote control of autonomous computers

- PC setup :
  - BIOS/UEFI :

    Parameters to check :

    AHCI/RAID/IDE, Compatible : if SSD or **SATA** hard disk and a modern OS, choose AHCI, if the OS cannot access to the hard disk, choose IDE, Compatible (might cause a little bit of performance loss) or check the motherboard documentation to find how to load the correct drivers during OS installation

    OS choice : if this option exists, a bad choice might cause an OS installation failure or reduced performance (presets often related to power supply, storage settings, etc.)

Some changes can be **interdependants** and require **several restarts** to appear or be taken into account!

Communication and remote control of autonomous computers

- PC setup :
  - Operating System (OS) installation :

    Be sure to **enable all devices before starting the installation** (in the BIOS, using hardware switches, etc.)

    A CD, DVD or USB key must be specifically prepared so that the PC can start from it and launch the installation

    If the **PC is 32 bit**, only a **32 bit OS** can be used, if the **PC is 64 bit**, we usually have the **choice** (BIOS settings might need to be modified accordingly)

    It is better to use the BIOS **Boot Menu Override** function (temporary startup menu) to start temporarily from a device, so that for next restarts, it should not try to start again the beginning of the installation

Communication and remote control of autonomous computers

- PC setup :
  - Operating system (OS) installation :

    Boot Menu Override :

    Devices can be sometimes **listed by categories** and a **dedicated menu** might exist to choose the order inside a category

    e.g. some USB can be listed in the Hard disk category along with the internal hard disk, therefore it might be necessary to check also the order inside this category…

    Category names can be sometimes misleading, check if you cannot find the desired device…

Communication and remote control of autonomous computers

- PC setup :
  - Operating system (OS) installation :

    UEFI boot (GPT) vs **Legacy boot (MBR)** :

    PCs with a **BIOS** (before 2010, 32 bit in general) start from hard disks or SSD with a **MBR** (contains a small piece of software as well as the partition table, with size limitations)

    PCs with a **UEFI** (after 2010) start differently and are especially compatible with **hard disks > 2 To** and can often also propose to work like a legacy BIOS with MBR disks

    => Choice of one or the other mode has often few impact for a robot (leave default), unless it has a hard disk or SSD > 2 To

Communication and remote control of autonomous computers

- PC setup :
  - Operating system (OS) installation :

    UEFI boot (GPT) vs **Legacy boot (MBR)** :

    Some PCs can have an **unusual mix between 32 bit or 64 bit UEFI and processors** (e.g. Intel Atom-based computers), in this cas esome OS might need a **specific installation procedure**…

    See http://liliputing.com/2014/10/run-ubuntu-zotac-zbox-pico-mini-pc-kinda.html

Communication and remote control of autonomous computers

- PC setup :
  - File systems :

    A hard disk usually contains several partitions, each one can contain a specific file system

    A **file system** defines a **folders, files, access rights structure** on a partition

    Disk image : file corresponding to a bit per bit copy of a disk (hard disk, SSD, DVD, etc.)

    Mounting a partition : attribution of a path to access the files and folders from a partition

    Partition root : first level from the folder tree of a partition

- PC setup :
  - File systems :

Windows :

NTFS : best choice if we only use Windows, should work also in recent Linux

exFAT : not so much used, FAT32 update

FAT32 : best portability, but with a 4 GB file size limitation and no access rights support

FAT (FAT16) : good portability with some electronic devices (e.g. micro SD card on Arduino) because simple to manage if we must write our custom driver on a microcontroller

Linux :

ext4 : default choice on recent Linux, not handled by Windows by default

ext2-3 : can be a good choice if data need to be shared with Windows since third-party drivers are available (check limitations : http://www.fs-driver.org/)

vfat : FAT32, usually Linux cannot be but can be a good choice if data need to be shared with Windows

swap : specific virtual memory, can be avoided in general if we have enough RAM

Communication and remote control of autonomous computers

- PC setup :
  - Operating system (OS) startup :

    Steps :

    PC power ON

    BIOS/UEFI starts, makes various checks and load the beginning of hard disks, SSD, USB keys in the order configured and **launch the first valid MBR/GPT**

    MBR/GPT contains a very short piece of software (512-8192 bytes) that can begin to offer choices to the user, or execute the rest of the software on the beginning of a specified other partition (**VBR**, on the **active partition** of the disk in the case of a MBR)

Communication and remote control of autonomous computers

- PC setup :
  - Operating system (OS) startup :

    Steps :

    Some files will begin to be loaded on the partition and the **boot loader** (its code is spread on the beginning of the disk (MBR), the beginning of a partition (VBR), and files on this partition) will then finish to load and will start to look for the operating system, or provide choices to the user

    The files from the chosen **operating system** (might be on another partition or disk) will then begin to launch its **kernel**, load **drivers** and **services** and display the **desktop** and the **applications**

Communication and remote control of autonomous computers

- PC setup :
  - Operating system (OS) startup :

    Boot loaders :

    Linux : GRUB, LILO

    Windows XP : NTLDR

    Windows Vista, 7, 8, 10 : BOOTMGR

    The different parts of the boot loaders are not always directly visible file explorers since some parts are not really standard files (e.g. MBR, VBR), specific low-level tools (e.g. **dd**) might be needed to access to the binary code and save it as a file

Communication and remote control of autonomous computers

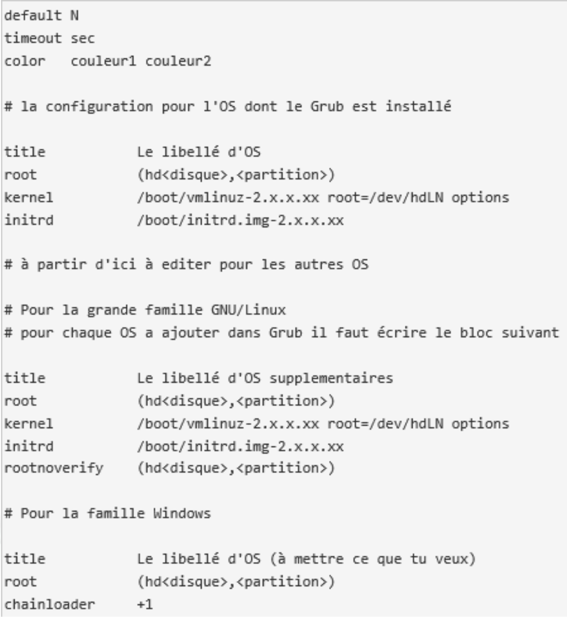
- PC setup :
  - Operating system (OS) startup :

    Specific commands or configuration files enable to set options

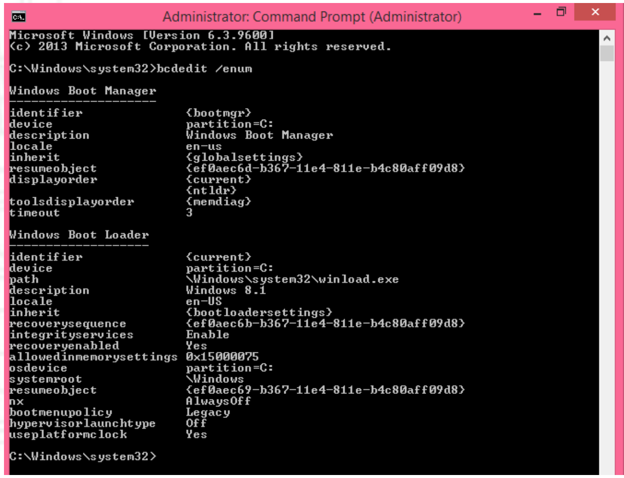
    /boot/grub/menu.lst or **/etc/default/grub** and **update-grub** for **GRUB**

    bcdedit, **bcdboot**, **bootsect** for **BOOTMGR**

    It is sometimes possible to chain boot loaders one after the other



```
default N
timeout sec
color    couleur1 couleur2

# la configuration pour l'OS dont le Grub est installé

title           Le libellé d'OS
root            (hd<disque>,<partition>)
kernel          /boot/vmlinuz-2.x.x.xx root=/dev/hdLN options
initrd          /boot/initrd.img-2.x.x.xx

# à partir d'ici à editer pour les autres OS

# Pour la grande famille GNU/Linux
# pour chaque OS a ajouter dans Grub il faut écrire le bloc suivant

title           Le libellé d'OS supplementaires
root            (hd<disque>,<partition>)
kernel          /boot/vmlinuz-2.x.x.xx root=/dev/hdLN options
initrd          /boot/initrd.img-2.x.x.xx
rootnoverify    (hd<disque>,<partition>)

# Pour la famille Windows

title           Le libellé d'OS (à mettre ce que tu veux)
root            (hd<disque>,<partition>)
chainloader     +1
```




```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Windows\system32>bcdedit /enum

Windows Boot Manager
--------------------
identifier              {bootmgr}
device                  partition=C:
description             Windows Boot Manager
locale                  en-us
inherit                 {globalsettings}
resumeobject            {ef0aec6d-b367-11e4-811e-b4c80aff09d8}
displayorder            {current}
                        {ntldr}
toolsdisplayorder       {memdiag}
timeout                 3

Windows Boot Loader
-------------------
identifier              {current}
device                  partition=C:
path                    \Windows\system32\winload.exe
description             Windows 8.1
locale                  en-US
inherit                 {bootloadersettings}
recoverysequence        {ef0aec6b-b367-11e4-811e-b4c80aff09d8}
integrityservices       Enable
recoveryenabled         Yes
allowedinmemorysettings 0x15000075
osdevice                partition=C:
systemroot              \Windows
resumeobject            {ef0aec69-b367-11e4-811e-b4c80aff09d8}
nx                      AlwaysOff
bootmenupolicy          Legacy
hypervisorlaunchtype    Off
useplatformclock        Yes

C:\Windows\system32>
```

Communication and remote control of autonomous computers

- PC setup :
  - Operating system (OS) startup :

    => When a computer does not startup correctly, we should try to identify at which specific step the problems begin to be able to find how to solve it!

Communication and remote control of autonomous computers

- PC setup :
  - Unattended OS installation, mass installations :

    For a group of robots, we might have several computers to set up

    Identical computers :

      Install everything on one, then make a low-level copy of its hard drive (e.g. with **dd**, **fsarchiver**, **dism**)

      Warning : BIOS settings are usually not stored on the hard drive, they must be modified for each PC

Communication and remote control of autonomous computers

- PC setup :
  - Unattended OS installation, mass installations :

    Heterogeneous computers :

    An automated installation of the OS and most of the applications can be prepared and launched in parallel on each PC

    Windows : **autounattend.xml** file, see **Windows PE**, Windows ADK, Windows AIK and usual application installers documentation : **Windows Installer**, **Inno Setup**, **InstallShield**, Nullsoft **NSIS**, etc. As a last resort, see **AutoIt** to **simulate mouse clicks**. **Keywords** : **unattended**, **silent**, **quiet**, **passive** installations

    Ubuntu : **preseed.cfg** file

    Modern versions of Windows and Ubuntu might under some conditions adapt during their first startup if we try to use the method of previous slide, but it is not really supported

Communication and remote control of autonomous computers

- PC setup :
  - Automatic launch of programs at OS startup

    Linux :

    cron

    Files in **/etc/init.d**

    Configure **auto login** and file **~/.bashrc** or similar

    Windows :

    Task Scheduler

    Creation of a service, e.g. with **srvany.exe** tool

    Configure **auto login** and **C:\ProgramData\Microsoft\Windows\Start Menu\Programs\StartUp**

Communication and remote control of autonomous computers

- PC setup :
  - Startup/stop/configuration of services

    Linux :

    Restarting network service :

    sudo /etc/init.d/networking restart

    or

    sudo service network-manager restart

    Windows :

    Stopping automatic update service :

    sc stop wuauserv

    Disabling automatic update service (warning : installation of some applications might fail) :

    sc config wuauserv start= disabled

Communication and remote control of autonomous computers

- PC setup :
  - Update management

    Be sure to disable automatic download and installation an autonomous system…

Communication and remote control of autonomous computers

- PC setup :
  - Introduction to networks, configuration for remote access, checking possible blockings, etc.

    See https://www.ensta-bretagne.fr/lebars/initiation_reseaux.pdf

Communication and remote control of autonomous computers

- PC setup :
  - Access rights problems

    Use **sudo** if message like « permission denied » on Linux, Windows equivalent is right-click « **Run as administrator** », otherwise **chmod -R 777 foldername** or **chmod 777 file** to set specific files or folders access rights on Linux, Windows equivalent is right-click **Properties\Security**.

Communication and remote control of autonomous computers

- PC setup :
  - Access rights problems

    If problems accessing a serial port device on Linux, add current user to **dialout** group (vi /etc/group), reboot if needed :

    sudo addgroup $USER dialout

    See also https://www.ensta-bretagne.fr/lebars/Share/Ubuntu.txt to rename a serial port depending on the device (**udev** rules)

Communication and remote control of autonomous computers

- PC setup :
  - Using specific libraries in C/C++

    See : http://www.ensta-bretagne.fr/lebars/tutorials/Complements_C-C++.pdf

Algorithms for autonomy

Algorithms for autonomy

- Purpose :
  - Make a robot autonomous i.e. able to do detection, localization, cartography, measurements missions with as few as possible human help

Algorithms for autonomy

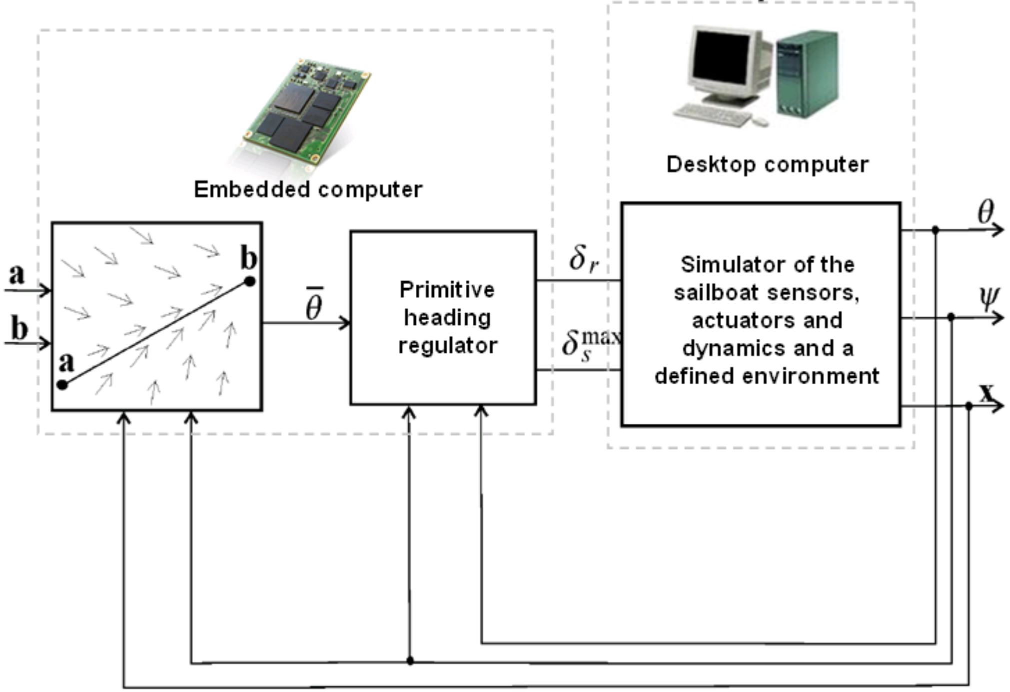
- Simulation :
  - Robot **in construction** :
    - Make a **software simulator** to start testing control algorithms, etc.
    - Make a **simplified or smaller version** as **hardware simulator** of the real robot…
  - **Existing teleoperated robot** or vehicle driven by a human :
    - **Add sensors and record the actuators inputs** to get an idea on its dynamic behavior
    - These data can be then used to design a simulator

Algorithms for autonomy

- Simulation :
  - Advantages to prepare simulations :
    - Less risks to break hardware
    - Easier to debug
    - etc.
  - Drawbacks :
    - Sometimes difficult to obtain models that describe correctly the robot => make experiments in teleoperated mode and record data if possible
    - Too complex model => maybe split in several more simple simulators to test specific aspects
    - Risk to develop algorithms that do not work in practice => make HIL (Hardware In the Loop) simulations

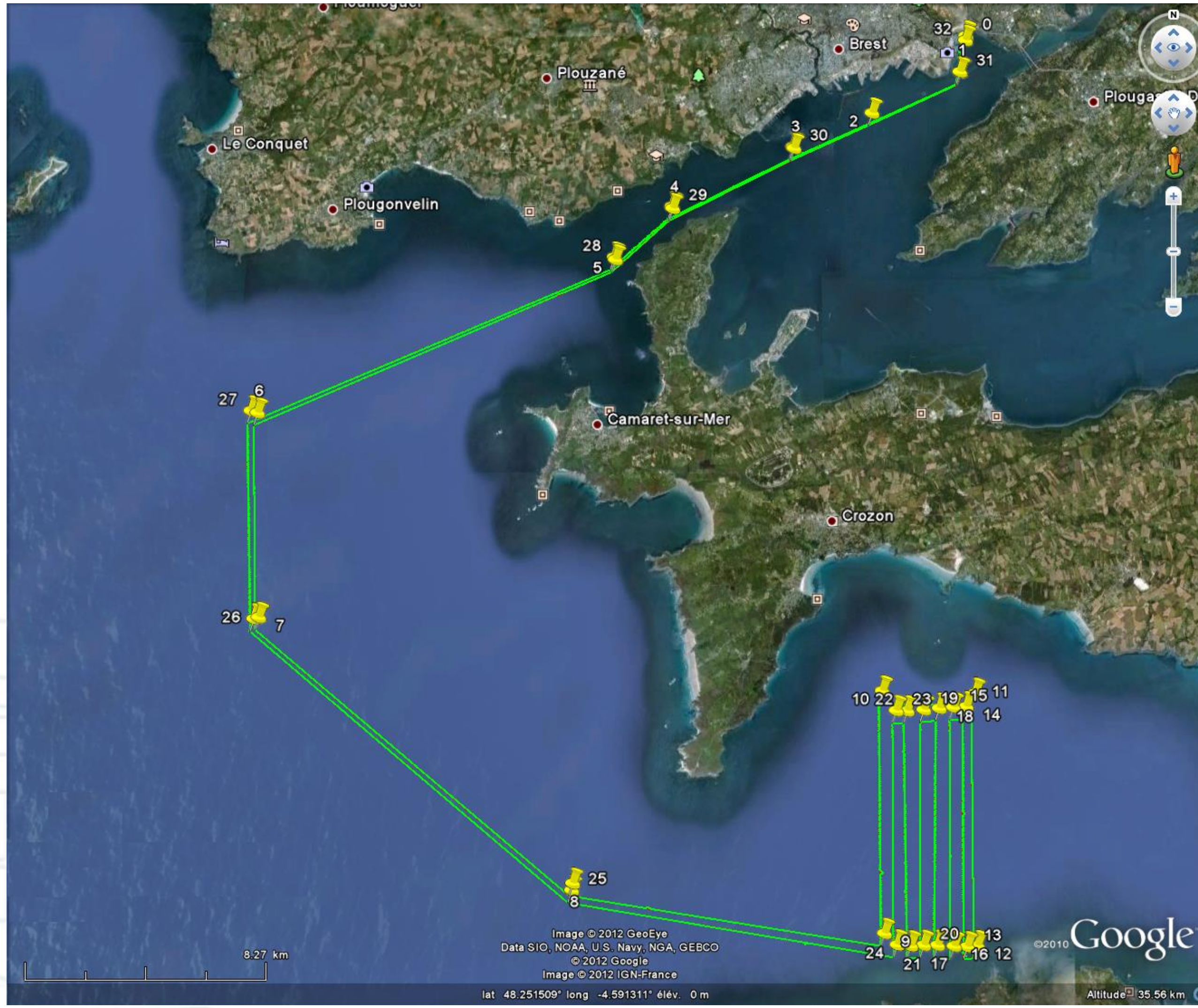
Algorithms for autonomy

- Example of HIL (Hardware In the Loop)simulation : with VAIMOS



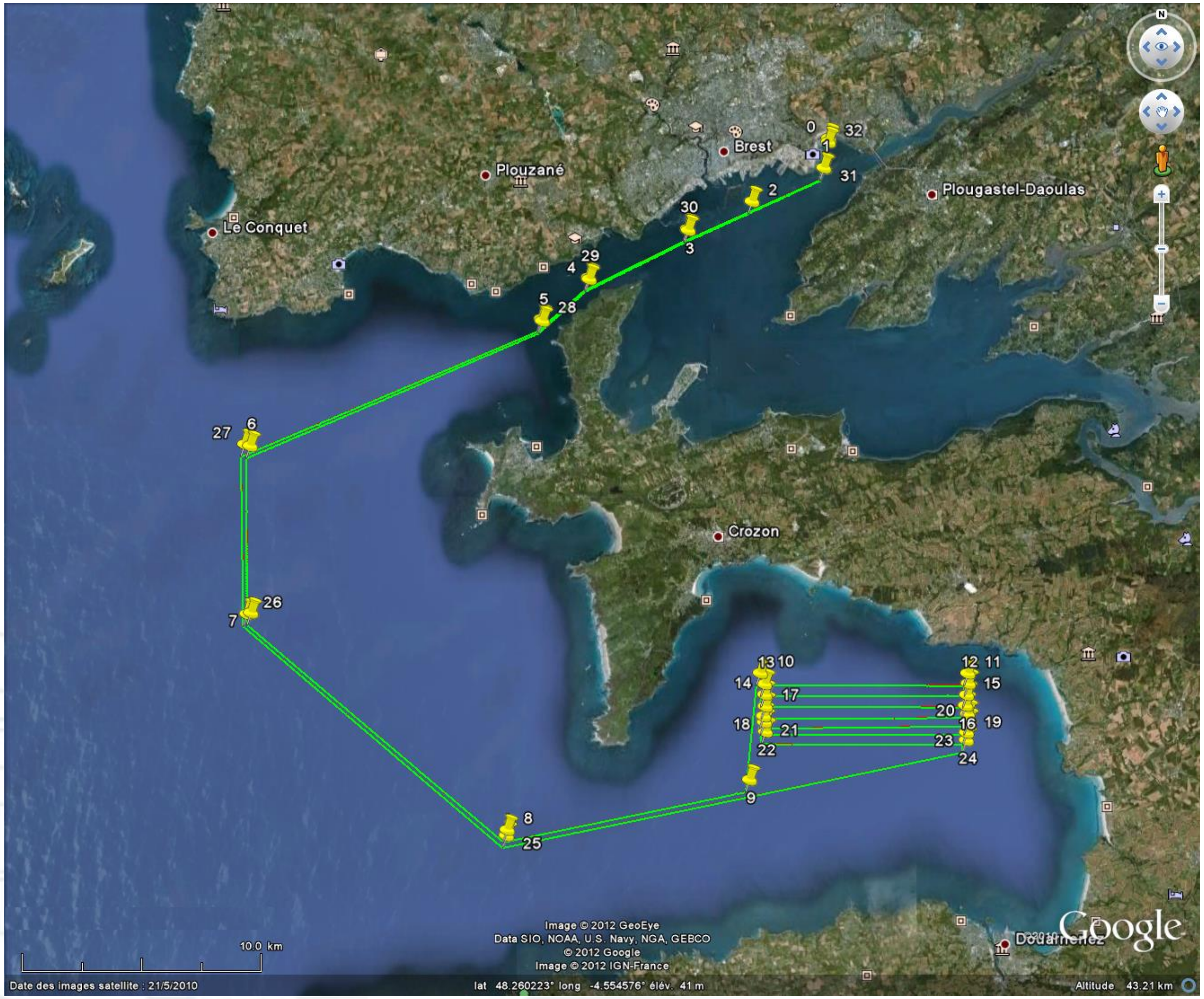
Algorithms for autonomy

- Example of HIL (Hardware In the Loop) simulation : with VAIMOS



Algorithms for autonomy

- Example of HIL (Hardware In the Loop) simulation : with VAIMOS



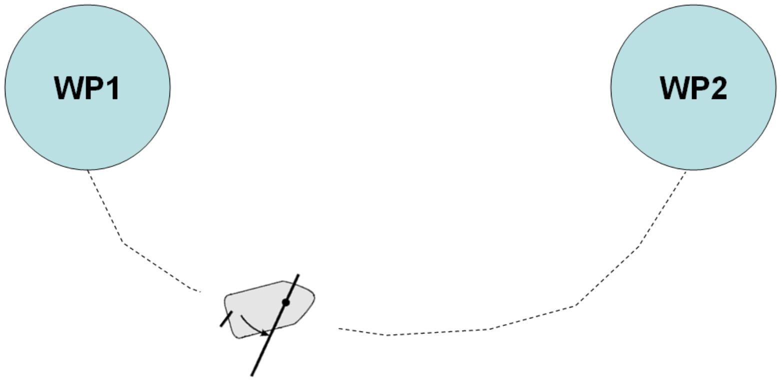
Algorithms for autonomy

- Robots simulation in MATLAB :
  - See http://www.ensta-bretagne.fr/lebars/simu_robots_matlab.ppt

- Simple control of a robot to follow waypoints :
  - See http://www.ensta-bretagne.fr/lebars/buggy_control_simple.ppt

Algorithms for autonomy

- From waypoints following to line following :
  - Basic waypoint following :
    - The robot follows a heading in direction of its waypoint
    - Waypoint reached when in a predefined radius
    - Problem : nothing prevent the drift between waypoints (because of currents...)

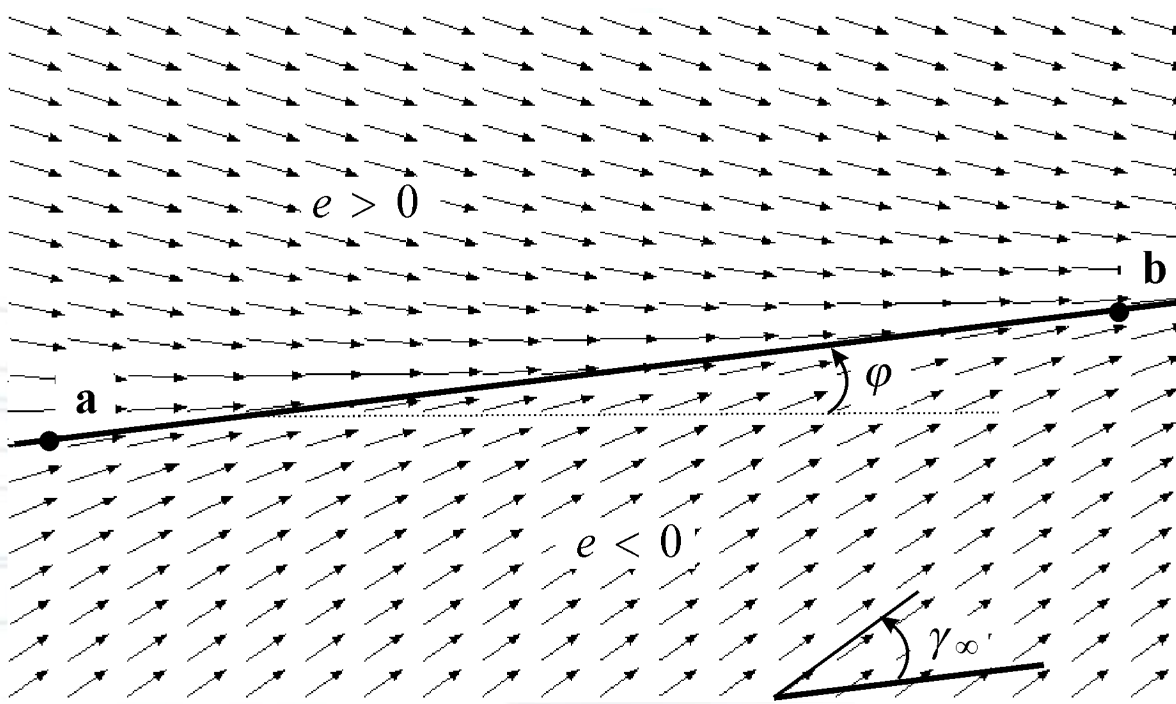


  - => Line following (use the distance to the line)

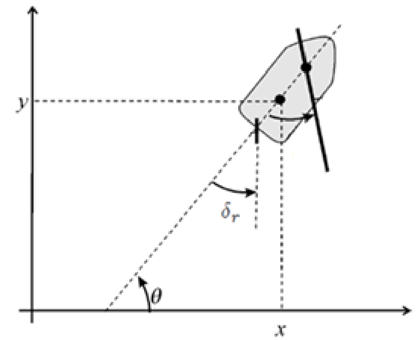
Algorithms for autonomy

- Line following :
  - The heading to follow will be the line between the 2 current waypoints, with an attractive angle to the line depending on the distance to the line

$$e = \det\left(\frac{\mathbf{b}-\mathbf{a}}{\|\mathbf{b}-\mathbf{a}\|}, \mathbf{m}-\mathbf{a}\right)$$
$$\varphi = \text{atan2}(\mathbf{b}-\mathbf{a})$$
$$\bar{\theta} = \varphi - \frac{2.\gamma_\infty}{\pi}.\text{atan}\left(\frac{e}{r}\right)$$



Algorithms for autonomy



- Line following :
  - Primitive controller stage for heading control
    - Rudder control

$$\delta_r = \begin{cases} \delta_r^{\max}.\sin\left(\theta - \bar{\theta}\right) & \text{if } \cos\left(\theta - \bar{\theta}\right) \geq 0 \\ \delta_r^{\max}.\mathrm{sign}\left(\sin\left(\theta - \bar{\theta}\right)\right) & \text{otherwise} \end{cases}$$

  - A supervisor sends heading to follow to the primitive heading controller depending on the current line

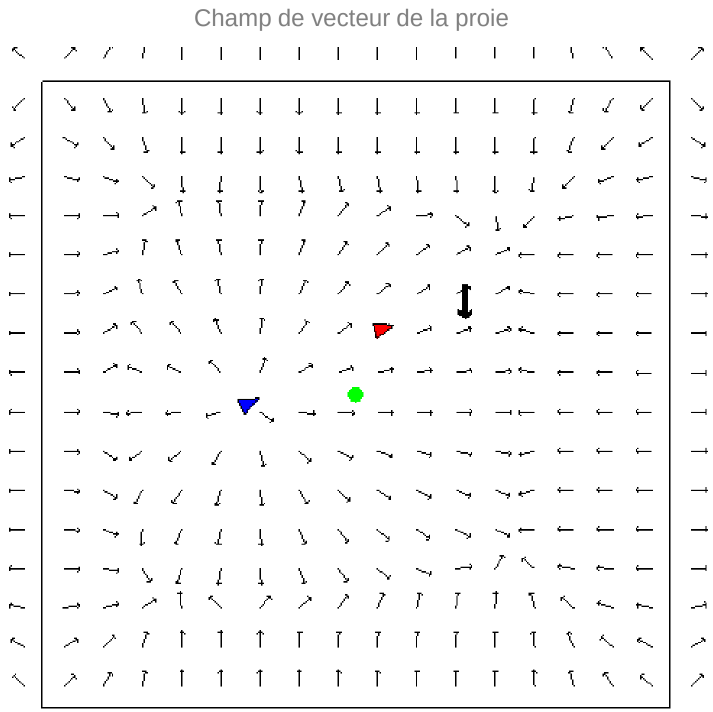
$$\begin{aligned} e &= \det\left(\frac{\mathbf{b}-\mathbf{a}}{\|\mathbf{b}-\mathbf{a}\|}, \mathbf{m}-\mathbf{a}\right) \\ \varphi &= \mathrm{atan2}(\mathbf{b}-\mathbf{a}) \\ \bar{\theta} &= \varphi - \frac{2.\gamma_\infty}{\pi}.\mathrm{atan}\left(\frac{e}{r}\right) \end{aligned}$$

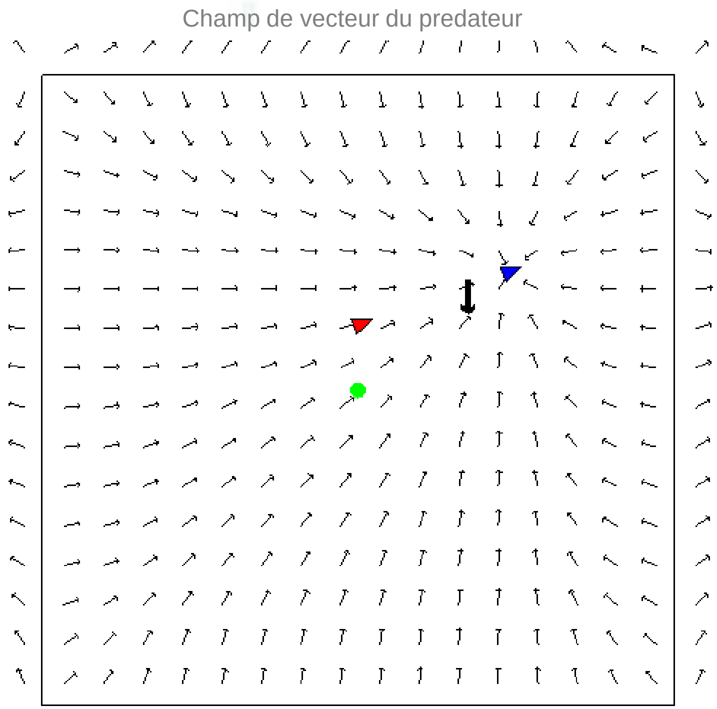
  - Navigation manager sends lines to supervisor and validates lines
    - Validation condition

$$\langle \mathbf{b}_j - \mathbf{a}_j, \mathbf{m} - \mathbf{b}_j \rangle \geq 0$$

Algorithms for autonomy

- Potential fields :
  - Enable the combination of attractive, repulsive points or lines
  - Example : prey and predator in a square where they are not allowed to escape





Algorithms for autonomy

- Other control algorithms :
  - See http://robmooc.ensta-bretagne.fr/

Algorithms for autonomy

- Localization :
  - **Knowing its position** at all times is mandatory in most of the cases
  - However, **sensors are not always available** to return directly an accurate position whatever the conditions…

    Example : GPS works well when we are on a road with a clear view of the sky, however it does not work at all in a long tunnel
  - Therefore, theoretical and numerical methods need to be developed to get estimations : these methods are called **state observer**

Algorithms for autonomy

- Localization :
  - In 1A (Automatics), you might have learned **Luenberger observers for linear systems**, others can be made depending on the specificities of the system :
    - **Dead-reckoning** : uses state equations to predict the next position at t+dt from the current one at t
    - Computations using geometrical relations : distance and/or angles to marks (objects whose position is more or less known) relations
    - **Kalman Filter** : see http://www.ensta-bretagne.fr/lebars/Obs_et_filt_de_Kalman.ppt
    - **Set-membership methods** : see https://www.ensta-bretagne.fr/jaulin/iamooc.html

Algorithms for autonomy

- Localization :
  - Note : in robotics, **state estimation**, **state observer** will be often a synonym of **localization**, **position estimation** since it is often the position that we really want to estimate, among the state variables

Algorithms for autonomy

- Localization :
  - Usual coordinate systems conversions :
    - Usual coordinate systems :
      - NED (North-East-Down)
      - NWU (North-West-Up)
      - ENU (East-North-Up)
      - WGS84 (axes orientation like ENU)
    - Distinctions between différents types of altitude (e.g. distance w.r.t. bottom often for a submarine), depth (distance w.r.t. surface), that can be different depending on the sensor and the physical effect used

Algorithms for autonomy

- Localization :
  - Usual coordinate systems conversions :

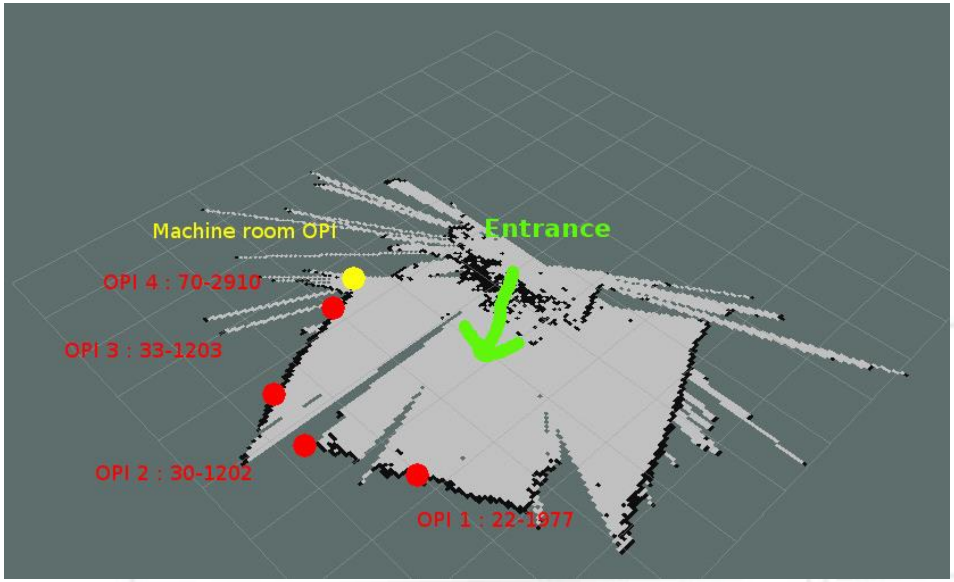
    NED->ENU : 90-yaw, -pitch, roll

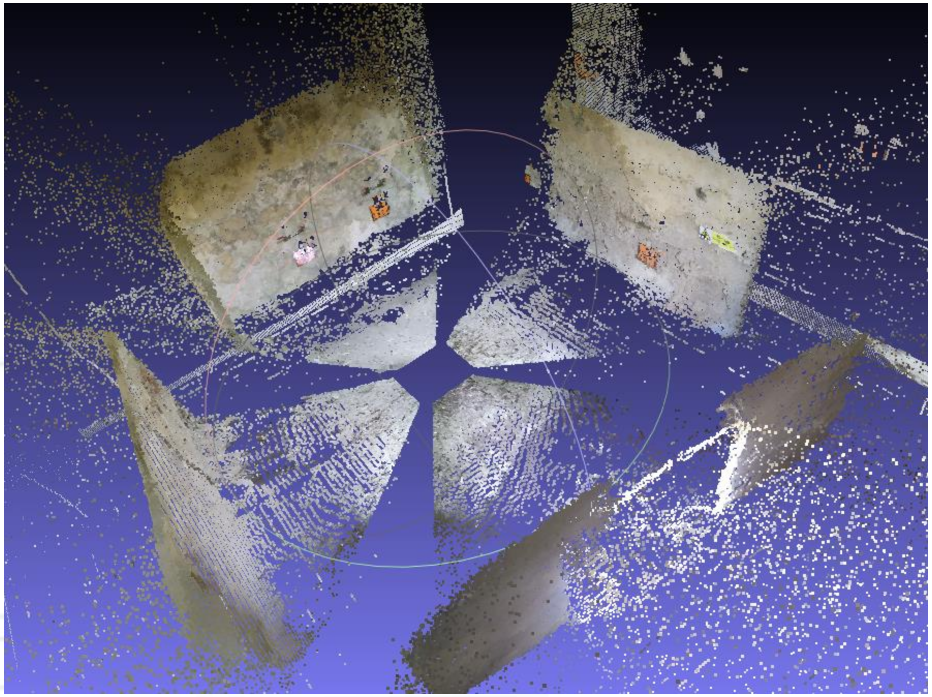
    WGS84 to local ENU coordinate system : see http://www.ensta-bretagne.fr/lebars/these/thesis_2011-11-23.pdf (p28) or « A simple controller for line following of sailboats » on http://www.ensta-bretagne.fr/jaulin/ to take into account the fact that the Earth cannot be assumed as flat for long distances

Algorithms for autonomy

- Cartography :
  - **Drawing of the robot environment**

    e.g. combination of lidar, sonar, camera, Kinect, etc. data.





Algorithms for autonomy

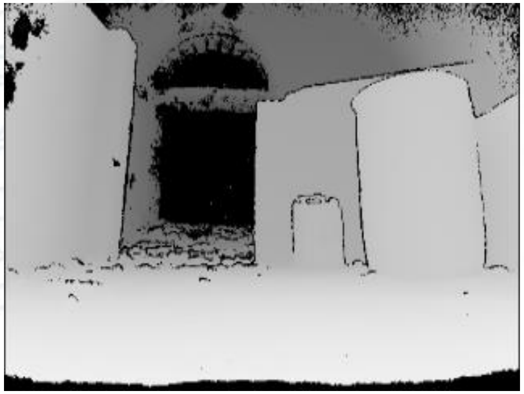
- Cartography :
  - Example of interesting sensor for cartography : Kinect 2 Xbox One

    Gather several sensors :
    - A Full HD color camera
    - An (active) infrared camera
    - Several microphones

Algorithms for autonomy

- Cartography :
  - Example of interesting sensor for cartography : Kinect 2 Xbox One

    It returns :
    - **Color** images
    - **Infrared** images
    - Gray-level images, with darkness depending on object distance to the sensor (**depth**)



Algorithms for autonomy

- Cartography :
  - Example of interesting sensor for cartography : Kinect 2 Xbox One

    Since these images come from separated sensors, several problems have to be considered :

    • Different view point -> areas visible on one sensor but not on another

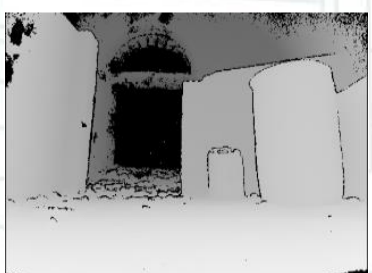
    • Different resolution and angular limits (FOV)

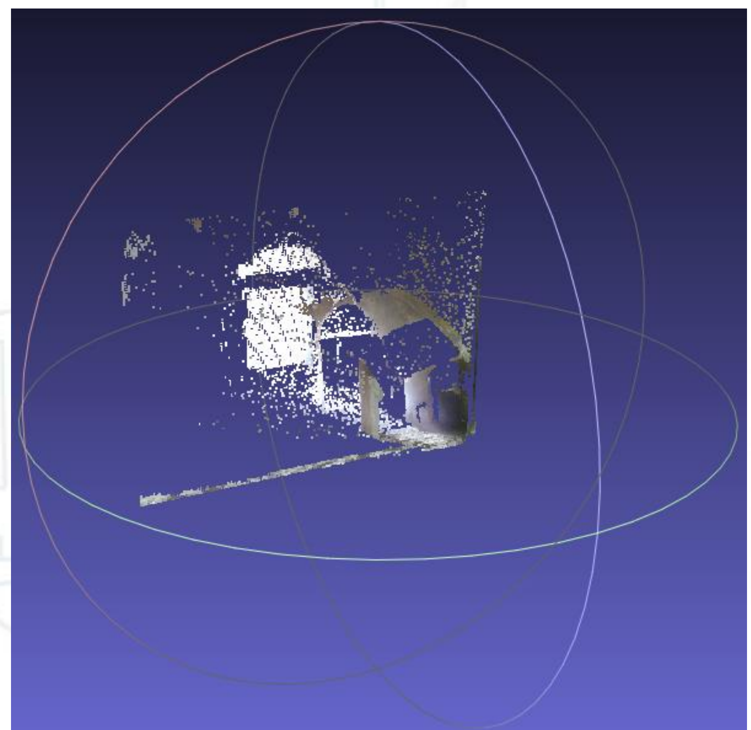
    • Symetries, etc.

Algorithms for autonomy

- Cartography :
  - Example de sensor interessant pour la cartography : Kinect 2 Xbox One

    Then, knowing the angular limits of the camera and the depth ranger, color image and depth image information can be combined to draw a point clouds in 3D...

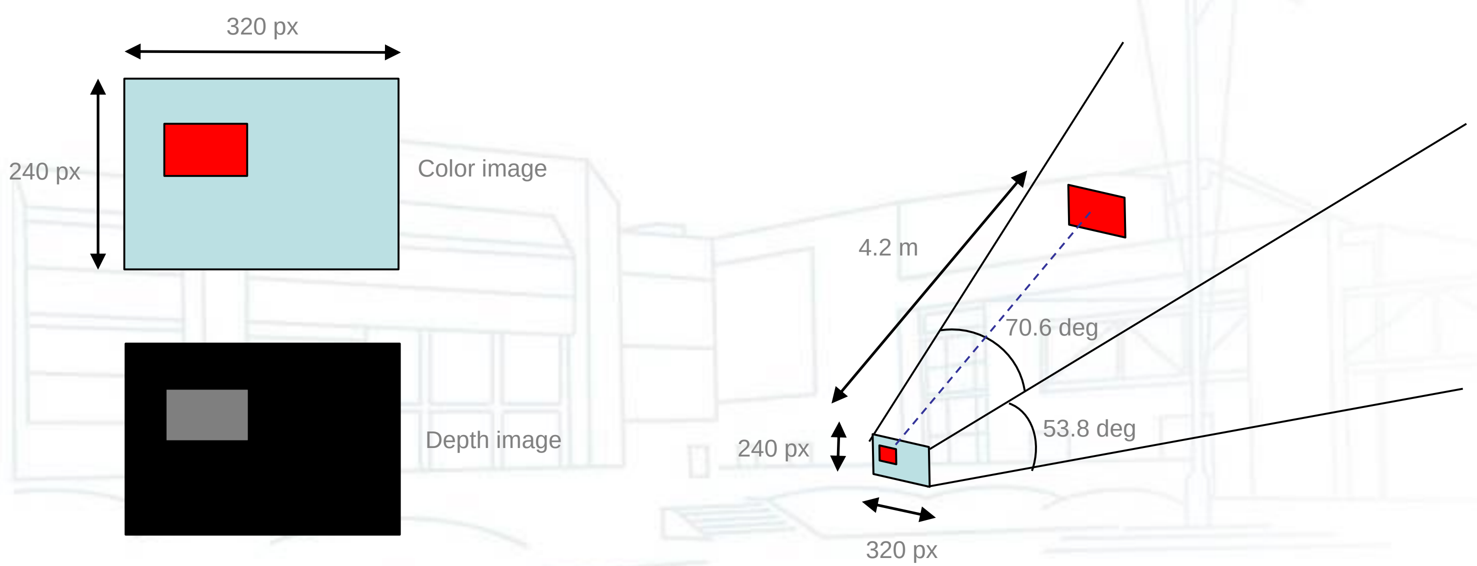




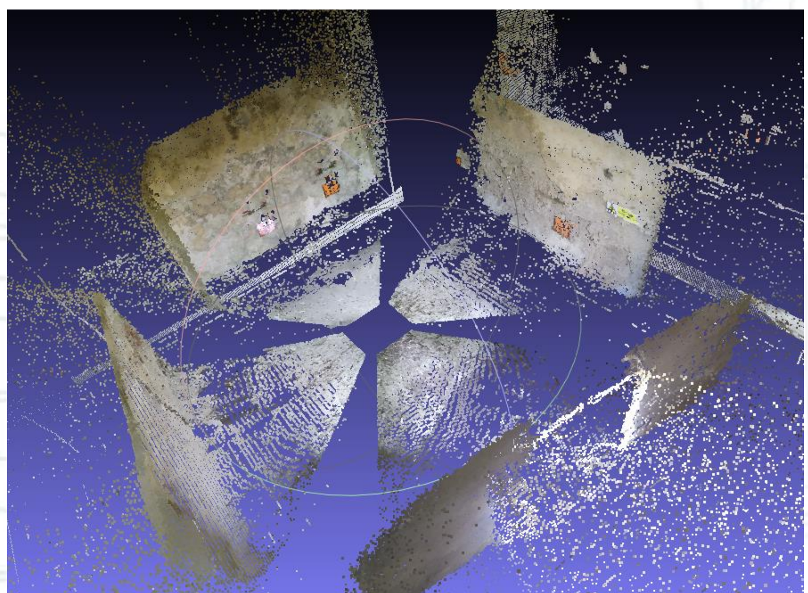
Algorithms for autonomy

- Cartography :
  - Example de sensor interessant pour la cartography : Kinect 2 Xbox One

    Example : 3D representation of a group of red pixels from the color image, whose corresponding **grey levels** in the depth picture represent a **distance of 4.2 m**, if images of e.g. **320 x 240 resolution** with a **FOV 70.6 x 53.8**



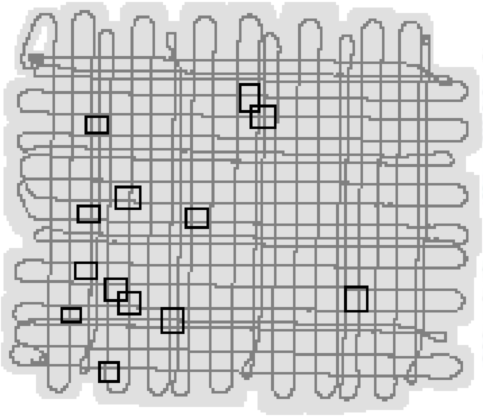
Algorithms for autonomy



Robot that took Kinect data
and turned 4 fimes of 90 deg

Algorithms for autonomy

- SLAM (Simultaneous Localization And Mapping) :
  - The robot often knows or measure :
    - Its initial and final position
    - Its dynamic model (state equations)
    - Its navigation data (orientation, speed, etc.)
    - Its teledetection data (camera, sonar, etc. images)
  - Idea of SLAM :
    - Estimate the position of marks from its own position estimation
    - Use the position of this marks to localize itself



Algorithms for autonomy

- SLAM (Simultaneous Localization And Mapping) :
  - Data association : knowing if a mark is a new one or an already known one can somtimes be a problem itself
  - Outliers (bad data) : determining if a sensor data is correct or not is also a real problem