DD-Boats

Luc Jaulin

February 8, 2026

Introduction

The aim of this lesson is to give an overview of the tools and methods for the control of a DD-Boat. Such a boat has two propellers. Instead of having a moving rudder, these type of boat turn by slowing down one propeller and speeding up the other. This is called "differential steering," and "DD" is used on the packaging to highlight this Dual Drive system.

In our DD-Boat, the sensors are magnetometers, accelerometers, gyrometers, and a GPS. The actuators are the two propellers (left and right).

DD-Boat (Dual Drive)

These robots are used at the Guerledan lake by the students in Robotics.

Chapter 1

Control

In order to illustrate the principle of model-free control, let us consider the case of a Dubins car described by:

$$\left\{\begin{array}{rcl} \dot{x} & = & v\cos\theta \\ \dot{y} & = & v\sin\theta \\ \dot{\theta} & = & u_2 \\ \dot{v} & = & u_1 - v \end{array}\right.$$

This model can be used for simulation, but not for obtaining the controller.

Proportional heading and speed controller

We will now propose a simple controller for this system by using our intuition about the system. Let us take $\widetilde{\theta} = \theta_d - \theta$ where θ_d is the desired heading and $\widetilde{v} = v_d - v$ where v_d is the desired speed.

- For speed control, we take:

  $$u_1 = a_1 \tanh \widetilde{v}$$

  where a_1 is a constant representing the maximum acceleration (in absolute value) that the motor is able to deliver. The hyperbolic tangent *tanh* (see Figure 1.1) is used as saturation function. Let us recall that :

  $$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{1.1}$$

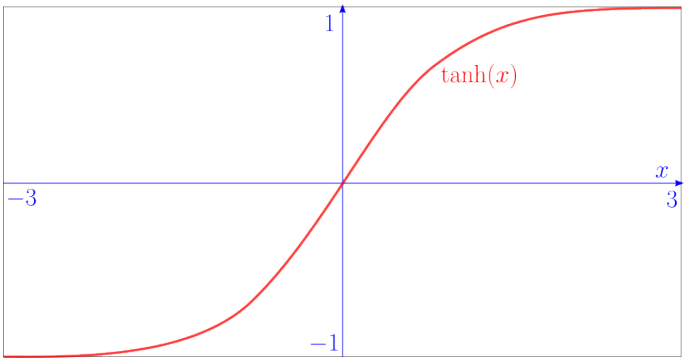

Figure 1.1: Hyperbolic tangent function used as saturation function

- For the heading control, we take:

$$u_2 = a_2 \cdot \text{sawtooth}(\widetilde{\theta})$$

In this last formula, *sawtooth* $\mathbb{R} \to \mathbb{R}$ corresponds to the sawtooth function which is the identity in $[-\pi, \pi]$ and 2π-periodic. It is defined by:

$$\text{sawtooth}(\widetilde{\theta}) = 2\text{atan}\left(\tan\frac{\widetilde{\theta}}{2}\right) = \text{mod}(\widetilde{\theta} + \pi, 2\pi) - \pi \tag{1.2}$$

Let us note that for numerical reasons, it is preferable to use the expression containing the *modulus* function. As illustrated in Figure 1.2, the function corresponds to an error in heading. The interest in taking an error $\widetilde{\theta}$ filtered by the *sawtooth* function is to avoid the problem of the $2k\pi$ modulus: we would like a $2k\pi$ to be considered non-zero.

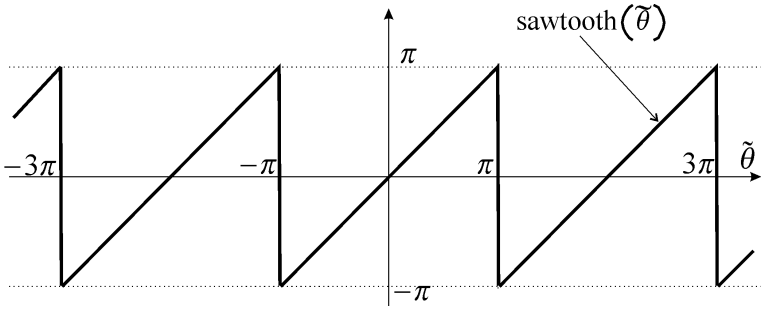

Figure 1.2: *Sawtooth* function used to avoid jumps in the heading control

The controller is thus defined by:

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} a_1 \cdot \tanh(v_d - v) \\ a_2 \cdot \text{sawtooth}(\theta_d - \theta) \end{pmatrix}$$

This model-free control does not need to use the state equations of the robot. It is based on the understanding that we have of the dynamics of the system and recalls our wireless operation method

of the Dubins car. It has two parameters a_1 and a_2 that are easy to set (a_1 represents the propelling power and a_2 the directional disturbances). Finally, this controller is easy to implement and to debug.

Navigation

Once a line tracking has been correctly implemented and validated, a number of lines should be chained together with the aim of performing complex missions (such as for instance connecting two points of the globe). In such a context, a Petri net strategy is well-adapted for representing the discrete state changes [1]. Before the robot is launched, it is in an initial state represented by the place p_0, as illustrated by Figure 1.3. The transition t_1 is crossed at the start of the mission. If everything goes well, the robot is in state p_1 and is tracking its first line $\mathbf{a}_1\mathbf{b}_1$.

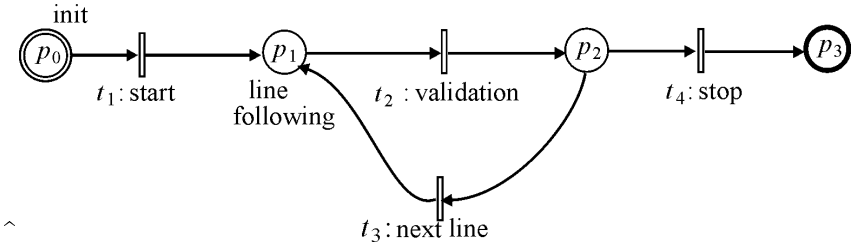

Figure 1.3: Petri net supervising the navigation of the drone

The line **ab** is validated as soon as the point **b** is surpassed, *i.e.*, if $\langle \mathbf{b}-\mathbf{a}, \mathbf{m}-\mathbf{b} \rangle > 0$ or equivalently,

$$(b_1 - a_1)(m_1 - b_1) + (b_2 - a_2)(m_2 - b_2) > 0$$

as illustrated by Figure 1.4.

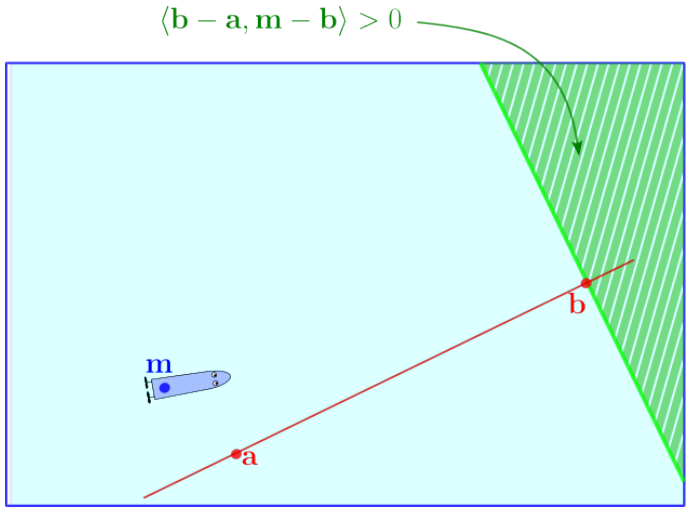

Figure 1.4: The DD-Boat validates the line **ab** as soon as it enters in the green half plane

Once a line is validated, we proceed to the next line. When the list of lines to track is empty, the mission ends (place p_3).

Chapter 2

Magnetism

This chapter introduces some useful concepts about Earth's magnetism and how it can be used for navigation of the DD-Boat.

Magnetometers are compact sensors designed to measure the Earth's magnetic field, which allows the drone to determine its heading relative to magnetic north. Combined with gyroscopes and accelerometers (in an IMU), they help stabilize orientation, correct drift, and enable reliable navigation.

The Earth generates a weak magnetic field (about 25–65 µT depending on location). A magnetometer measures the vector components of this field along three axes. From these components, the drone's heading can be computed.

Most small drones use magnetometers, based on one of the following effects:

- Hall effect: When an electric current (gray) flows through a conductor (yellow) placed in a magnetic field, the field causes a transverse voltage. This measured voltage is proportional to the magnetic field strength.
- Magnetoresistive effect (dominant technology): The sensor contains tiny resistive elements whose resistance varies depending on the direction and strength of the magnetic field. By measuring these resistance changes, the sensor determines the magnetic field components.

When we place a magnetometer inside the DD-Boat, it does not measure only the Earth's magnetic field. It also "sees" magnetic effects created by the robot itself. To understand this, we can imagine that there are many tiny magnets inside the robot. When the boat moves, some of these tiny magnets are free to turn to be aligned with the Earth's magnetic field. They do not have a fixed direction, but they are easily influenced by the surrounding field. These corresponds to *soft iron* and the result corresponds to the magnetic distortion introduced in the previous section. Soft-iron effects distort the Earth's magnetic field: they stretch it or bend it, but they do not create a constant magnetic field on their own. Their effect depends on the robot's orientation.

Other tiny magnets are fixed to the robot. They keep the same direction in the robot's frame no matter how it moves. These correspond to *hard iron*. Hard-iron effects add a constant magnetic field to the measurement, shifting the magnetometer readings by a fixed amount.

During magnetometer calibration, we try to identify and remove both effects: hard iron causes a shift of the measurements, soft iron causes a deformation of the measurements (see Figure 2.1, bottom). By compensating both, the magnetometer can correctly measure the Earth's magnetic field and give an accurate heading.

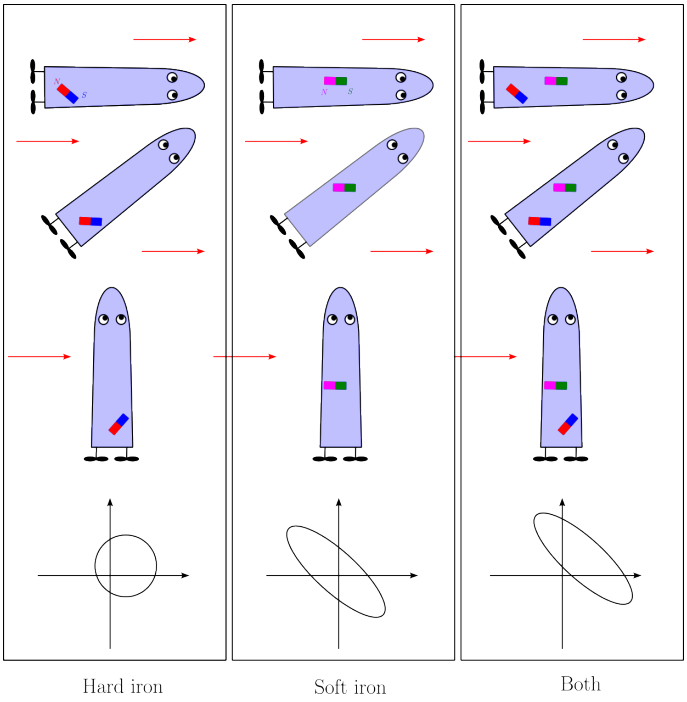

Figure 2.1: Left : hard iron perturbation, Center: solf iron; Right: Both

2.1 Calibration of the magnetometers

When the magnetic field is measured in the DD-Boat, two types of perturbation occur

- Soft iron effects are caused by nearby materials that distort the Earth's magnetic field. They don't create their own magnetic field, but they bend and reshape the field lines depending on orientation. This results in direction-dependent scaling and skewing of compass measurements (often seen as an ellipse instead of a circle when rotating the sensor).

- Hard iron effects come from permanently magnetized objects near the compass (like magnets, speakers, or magnetized screws). These objects add a constant magnetic offset to the Earth's field, shifting all compass readings in one direction regardless of orientation.

In calibration terms, hard iron is corrected by offset (bias) removal and soft iron is corrected by scaling and axis alignment (matrix correction).

Denote by $\mathbf{y} \in \mathbb{R}^3$ the magnetic field that exists at the robot's position if we remove the robot. It the correct magnetic field, *i.e.*, the one we would like to measure. Unfortunately, the raw magnetometer returns a vector of magnetic measurements $\mathbf{x} \in \mathbb{R}^3$ that is corrupted by the presence of the robot. We can assume that $\mathbf{x}$ and $\mathbf{y}$ are linked by the linear equation:

$$\mathbf{x} = \mathbf{A}\mathbf{y} + \bar{\mathbf{x}}.$$

The calibration step searches for an estimation $\hat{\mathbf{A}}$ and $\hat{\bar{\mathbf{x}}}$ for $\mathbf{A}$ and $\bar{\mathbf{x}}$. It is obtained by a sequence of measurement made at a given position of the robot, but for different orientations, as illustrated by Figure 2.2. These quantities depend on the boat only.

The *corrector* is given by

$$\hat{\mathbf{y}} = \hat{\mathbf{A}}^{-1}(\mathbf{x} - \hat{\bar{\mathbf{x}}})$$

The *calibrated magnetometer* includes both the raw magnetometer and the corrector.

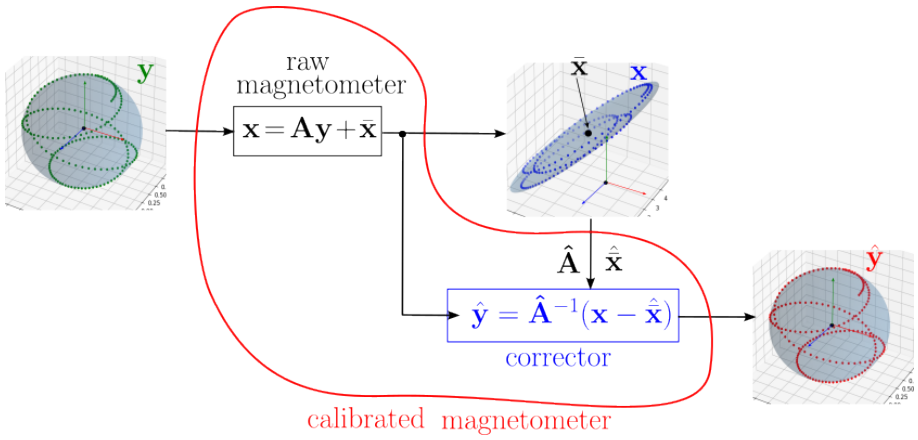

Figure 2.2: Calibrated magnetometer

2.1.1 Pointing method

The pointing method is probably the simplest to understand, but requires to orientate the robot toward some specific directions, which is not always possible, as for instance if the robot is huge.

We take four measurements $\mathbf{x}_N$, $\mathbf{x}_S$, $\mathbf{x}_W$, $\mathbf{x}_U$, with the DD-Boat, as illustrated by Figure 2.3:

- Direction N : The robot is laid flat toward the North
- Direction S : The robot is laid toward the South, upside down
- Direction W : The robot is laid flat toward the West

- Direction U : The robot is set vertically with its z axis toward the North

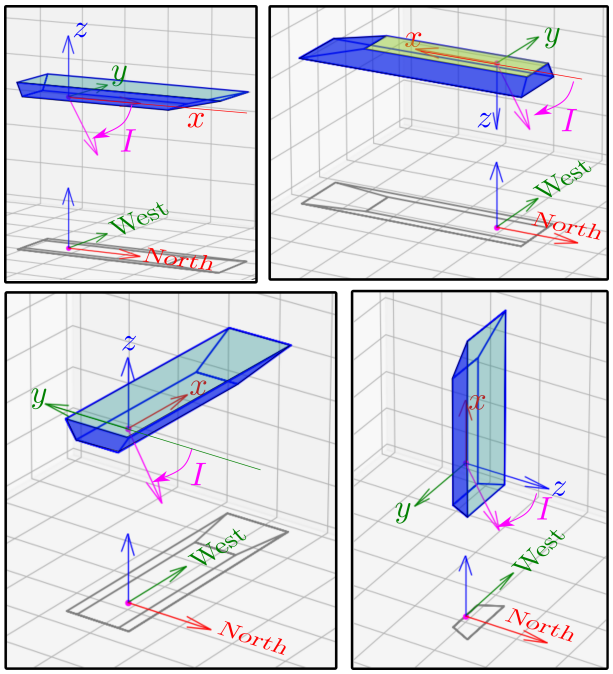

Figure 2.3: Procedure for the calibration of the magnetometers

The four corresponding transformation matrices are

$$\mathbf{R}_N = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{R}_S = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

$$\mathbf{R}_W = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{R}_U = \begin{pmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

The corrected magnetic field should be

$$\mathbf{y}_N = \begin{pmatrix} \beta \cdot \cos I \\ 0 \\ -\beta \cdot \sin I \end{pmatrix}, \qquad \mathbf{y}_S = \mathbf{R}_S^{\mathrm{T}} \cdot \mathbf{y}_N = \begin{pmatrix} -\beta \cdot \cos I \\ 0 \\ \beta \cdot \sin I \end{pmatrix}$$

$$\mathbf{y}_W = \mathbf{R}_W^{\mathrm{T}} \cdot \mathbf{y}_N = \begin{pmatrix} 0 \\ -\beta \cdot \cos I \\ -\beta \cdot \sin I \end{pmatrix}, \quad \mathbf{y}_U = \mathbf{R}_U^{\mathrm{T}} \cdot \mathbf{y}_N = \begin{pmatrix} -\beta \cdot \sin I \\ 0 \\ \beta \cdot \cos I \end{pmatrix}$$

where $\beta = 46\mu T$.

Since

$$\begin{array}{rcl} \mathbf{0} & = & \mathbf{y}_N + \mathbf{y}_S \\ & = & \mathbf{A}^{-1}(\mathbf{x}_N - \bar{\mathbf{x}}) + \mathbf{A}^{-1}(\mathbf{x}_S - \bar{\mathbf{x}}) \end{array}$$

we get

$$(\mathbf{x}_N - \bar{\mathbf{x}}) + (\mathbf{x}_S - \bar{\mathbf{x}}) = \mathbf{0}$$

i.e.,

$$\bar{\mathbf{x}} = \frac{1}{2}(\mathbf{x}_N + \mathbf{x}_S).$$

Moreover

$$\begin{array}{rcl} \mathbf{A}\mathbf{y}_N & = & \mathbf{x}_N - \bar{\mathbf{x}} \\ \mathbf{A}\mathbf{y}_W & = & \mathbf{x}_W - \bar{\mathbf{x}} \\ \mathbf{A}\mathbf{y}_U & = & \mathbf{x}_U - \bar{\mathbf{x}} \end{array}$$

yields

$$\mathbf{A} \cdot \underbrace{\begin{pmatrix} \mathbf{y}_N & \mathbf{y}_W & \mathbf{y}_U \end{pmatrix}}_{\mathbf{Y}} = \underbrace{\begin{pmatrix} \mathbf{x}_N - \bar{\mathbf{x}} & \mathbf{x}_W - \bar{\mathbf{x}} & \mathbf{x}_U - \bar{\mathbf{x}} \end{pmatrix}}_{\mathbf{X}}$$

i.e., $\mathbf{A} = \mathbf{X} \cdot \mathbf{Y}^{-1}$.

We thus get the following calibration procedure.

	Algorithm MagCalib(In: $\mathbf{x}_N$, $\mathbf{x}_S$, $\mathbf{x}_W$, $\mathbf{x}_U$; Out :$\hat{\mathbf{A}}, \hat{\bar{\mathbf{x}}}$)
1	$I = 64 \cdot \frac{\pi}{180}; \beta = 46.0 \cdot 10^{-6} T.$
2	$\hat{\bar{\mathbf{x}}} = \frac{1}{2}(\mathbf{x}_N + \mathbf{x}_S)$
3	$\mathbf{Y} = \beta \cdot \begin{pmatrix} \cos I & 0 & -\sin I \\ 0 & -\cos I & 0 \\ -\sin I & -\sin I & \cos I \end{pmatrix}$
4	$\mathbf{X} = \begin{pmatrix} \mathbf{x}_N - \bar{\mathbf{x}} & \mathbf{x}_W - \bar{\mathbf{x}} & \mathbf{x}_U - \bar{\mathbf{x}} \end{pmatrix}$
5	$\hat{\mathbf{A}} = \mathbf{X} \cdot \mathbf{Y}^{-1}$
6	Return $\hat{\mathbf{A}}$, $\hat{\bar{\mathbf{x}}}$

2.2 Magnetometer autocalibration

Magnetometer autocalibration is the process of determining hard- and soft-iron calibration parameters from arbitrary 3D orientation data, without requiring known attitude or external references. Equivalently, we want to find a transformation that converts the distorted ellipsoid into a sphere centered at the origin, to correct for hard- and soft-iron effects, without taking care of the rotation part which can be known in advance.

2.2.1 Normalization of the ellipse

We have many points $(x_1(i), x_2(i), x_3(i))$, $i \geq 1$, collected by the magnetometer.

All belong to an ellipsoid:

$$\underbrace{p_1x_1^2 + p_2x_2^2 + p_3x_3^2 + p_4x_1x_2 + p_5x_1x_3 + p_6x_2x_3 + p_7x_1 + p_8x_2 + p_9x_3}_{\mathbf{f_p(x)}} = 1 \tag{2.1}$$

i.e.,

$$\left(x_1^2|x_2^2|x_3^2|x_1x_2|x_1x_3|x_2x_3|x_1|x_2|x_3\right) \cdot \mathbf{p} = 1$$

We should have

$$\underbrace{\begin{pmatrix} x_1^2(1) & x_2^2(1) & x_3^2(1) & x_1(1)x_2(1) & x_1(1)x_3(1) & x_2(1)x_3(1) & x_1(1) & x_2(1) & x_3(1) \\ x_1^2(2) & x_2^2(2) & x_3^2(2) & x_1(2)x_2(2) & x_1(2)x_3(2) & x_2(2)x_3(2) & x_1(2) & x_2(2) & x_3(2) \\ x_1^2(3) & x_2^2(3) & x_3^2(3) & x_1(3)x_2(3) & x_1(3)x_3(3) & x_2(3)x_3(3) & x_1(3) & x_2(3) & x_3(3) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}}_{\mathbf{X}} \cdot \mathbf{p} = \underbrace{\begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \end{pmatrix}}_{\mathbf{1}} \tag{2.2}$$

A least-square solution of (2.2) is

$$\mathbf{p} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1} \cdot \mathbf{X}^{\mathrm{T}} \cdot \mathbf{1}$$

Proposition 1. *Consider the general quadratic surface in* $\mathbb{R}^3$*:*

$$p_1x_1^2 + p_2x_2^2 + p_3x_3^2 + p_4x_1x_2 + p_5x_1x_3 + p_6x_2x_3 + p_7x_1 + p_8x_2 + p_9x_3 = 1$$

or equivalently

$$\mathbf{x}^{\mathrm{T}} \underbrace{\begin{pmatrix} p_1 & \frac{p_4}{2} & \frac{p_5}{2} \\ \frac{p_4}{2} & p_2 & \frac{p_6}{2} \\ \frac{p_5}{2} & \frac{p_6}{2} & p_3 \end{pmatrix}}_{\mathbf{P}} \mathbf{x} + \underbrace{\mathbf{c}^{\mathrm{T}}}_{(p_7,p_8,p_9)} \cdot \mathbf{x} = 1$$

If the surface corresponds to an ellipsoid, we can write the equation in the canonical quadratic (matrix) form :

$$(\mathbf{x} - \bar{\mathbf{x}})^{\mathrm{T}}\mathbf{Q}(\mathbf{x} - \bar{\mathbf{x}}) = 1$$

where

$$\begin{array}{rcl} \bar{\mathbf{x}} & = & -\frac{1}{2}\mathbf{P}^{-1}\mathbf{c} \\ \mathbf{Q} & = & \frac{\mathbf{P}}{1+\bar{\mathbf{x}}^T\mathbf{P}\bar{\mathbf{x}}} \end{array}$$

The vector $\bar{\mathbf{x}} \in \mathbb{R}^3$ is the center of the ellipsoid and $\mathbf{Q}$ is a symmetric positive definite matrix.

Proof. The center $\bar{\mathbf{x}}$ satisfies the stationary condition

$$\frac{d}{d\mathbf{x}}\left(\mathbf{x}^{\mathrm{T}}\mathbf{P}\mathbf{x}+\mathbf{c}^{\mathrm{T}}\mathbf{x}\right)=\mathbf{0}$$

i.e.,

$$2\mathbf{P}\bar{\mathbf{x}}+\mathbf{c}=\mathbf{0}$$

Thus,

$$\bar{\mathbf{x}}=-\tfrac{1}{2}\mathbf{P}^{-1}\mathbf{c}.$$

Set $\mathbf{z}=\mathbf{x}-\bar{\mathbf{x}}$ to shift the coordinate system. We have

$$(\mathbf{z}+\bar{\mathbf{x}})^T\mathbf{P}(\mathbf{z}+\bar{\mathbf{x}})+\mathbf{c}^T(\mathbf{z}+\bar{\mathbf{x}})=1$$

i.e.,

$$\mathbf{z}^T\mathbf{P}\mathbf{z}+\underbrace{\bar{\mathbf{x}}^{\mathrm{T}}\mathbf{P}\mathbf{z}+\mathbf{z}^{\mathrm{T}}\mathbf{P}\bar{\mathbf{x}}+\mathbf{c}^{\mathrm{T}}\mathbf{z}}_{=\mathbf{0}}+\bar{\mathbf{x}}^{\mathrm{T}}\mathbf{P}\bar{\mathbf{x}}+\mathbf{c}^{\mathrm{T}}\bar{\mathbf{x}}=1$$

The fact that the linear terms cancel out is due to the fact that the minimum is reached for $\mathbf{z}=\mathbf{0}$. Since $\mathbf{c}=-2\mathbf{P}\bar{\mathbf{x}}$, we get

$$\mathbf{z}^T\mathbf{P}\mathbf{z}-\bar{\mathbf{x}}^{\mathrm{T}}\mathbf{P}\bar{\mathbf{x}}=1$$

i.e.,

$$\mathbf{z}^T\mathbf{P}\mathbf{z}=1+\bar{\mathbf{x}}^T\mathbf{P}\bar{\mathbf{x}}$$

i.e.,

$$\mathbf{z}^T\left(\frac{\mathbf{P}}{1+\bar{\mathbf{x}}^T\mathbf{P}\bar{\mathbf{x}}}\right)\mathbf{z}=1$$

Therefore,

$$\mathbf{Q}=\frac{\mathbf{P}}{1+\bar{\mathbf{x}}^T\mathbf{P}\bar{\mathbf{x}}}$$

which concludes the proof. □

Proposition 2. *To transform the ellipsoid to a unit sphere, we take the transformation*

$$\mathbf{y}=\sqrt{\mathbf{Q}}\cdot(\mathbf{x}-\bar{\mathbf{x}})$$

Proof. We have

$$\mathbf{x}^{\mathrm{T}}\mathbf{P}\mathbf{x}+\mathbf{c}^{\mathrm{T}}\cdot\mathbf{x}=1$$

Since

$$\begin{array}{rcl}\bar{\mathbf{x}} & = & -\frac{1}{2}\mathbf{P}^{-1}\mathbf{c}\\ \mathbf{Q} & = & \frac{\mathbf{P}}{1+\bar{\mathbf{x}}^T\mathbf{P}\bar{\mathbf{x}}}=\frac{\mathbf{P}}{1-\frac{1}{2}\bar{\mathbf{x}}^T\mathbf{c}}=\frac{\mathbf{P}}{1+\frac{1}{4}\mathbf{c}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{c}}\\ \mathbf{x} & = & \mathbf{Q}^{-\frac{1}{2}}\mathbf{y}+\bar{\mathbf{x}}\end{array}$$

Thus

$$\begin{aligned}
\mathbf{y}^{\mathrm{T}}\mathbf{y} &= (\mathbf{x}-\bar{\mathbf{x}})^{\mathrm{T}}\mathbf{Q}(\mathbf{x}-\bar{\mathbf{x}}) \\
&= (\mathbf{x}-\bar{\mathbf{x}})^{\mathrm{T}}\frac{\mathbf{P}}{1+\frac{1}{4}\mathbf{c}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{c}}(\mathbf{x}-\bar{\mathbf{x}}) \\
&= (\mathbf{x}+\tfrac{1}{2}\mathbf{P}^{-1}\mathbf{c})^{\mathrm{T}}\frac{\mathbf{P}}{1+\frac{1}{4}\mathbf{c}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{c}}(\mathbf{x}+\tfrac{1}{2}\mathbf{P}^{-1}\mathbf{c}) \\
&= \frac{1}{1+\frac{1}{4}\mathbf{c}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{c}}\left(\mathbf{x}^{\mathrm{T}}\mathbf{P}\mathbf{x}+(\mathbf{P}^{-1}\mathbf{c})^{\mathrm{T}}\mathbf{P}\mathbf{x}+(\tfrac{1}{2}\mathbf{P}^{-1}\mathbf{c})^{\mathrm{T}}\mathbf{P}(\tfrac{1}{2}\mathbf{P}^{-1}\mathbf{c})\right) \\
&= \frac{1}{1+\frac{1}{4}\mathbf{c}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{c}}\left(\mathbf{x}^{\mathrm{T}}\mathbf{P}\mathbf{x}+\mathbf{c}^{\mathrm{T}}\mathbf{x}+\tfrac{1}{4}\mathbf{c}^{\mathrm{T}}\mathbf{P}^{-1}\mathbf{c}\right) \\
&= 1
\end{aligned}$$

□

2.2.2 Pointing toward cardinal direction

As said previously, magnetometer autocalibration requires no known attitude or external references. Now, in practice, we may not know exactly how the magnetometer has been positioned inside the boat and a precise pointing procedure should be performed. This means that a rotation should be added to the transformation to rotate the sphere properly. The complete correction should have the form $\beta \cdot \mathbf{R}.\sqrt{\mathbf{Q}} \cdot (\mathbf{x}-\bar{\mathbf{x}})$, where β is the magnitude of the magnetic field. For simplicity, we compute the normalized version of this correction:

$$\mathbf{y} = \mathbf{R}.\sqrt{\mathbf{Q}} \cdot (\mathbf{x}-\bar{\mathbf{x}})$$

We take two measurements $\mathbf{x}_N, \mathbf{x}_W$, with the DD-Boat, as illustrated by Figure 2.4.

- Direction N : The robot is laid flat toward the North
- Direction W : The robot is laid flat toward the West

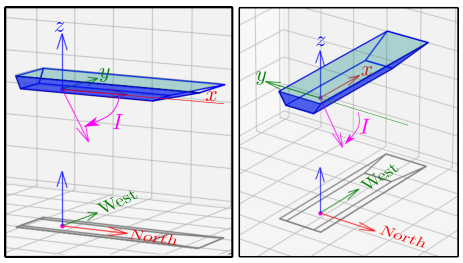

Figure 2.4: Magnetic field at the surface of the Earth

The corrected magnetic field should be

$$\mathbf{y}_N = \begin{pmatrix} \cos I \\ 0 \\ -\sin I \end{pmatrix}, \quad \mathbf{y}_W = \begin{pmatrix} 0 \\ -\cos I \\ -\sin I \end{pmatrix}$$

We have

$$\begin{aligned} \mathbf{y}_N &= \mathbf{R}.\underbrace{\sqrt{\mathbf{Q}}\cdot(\mathbf{x}_N-\bar{\mathbf{x}})}_{\mathbf{v}_N} \\ \mathbf{y}_W &= \mathbf{R}.\underbrace{\sqrt{\mathbf{Q}}\cdot(\mathbf{x}_W-\bar{\mathbf{x}})}_{\mathbf{v}_W} \end{aligned}$$

We add the equation

$$\mathbf{y}_N \wedge \mathbf{y}_W = \mathbf{R}.(\mathbf{v}_N \wedge \mathbf{v}_W)$$

to get

$$\underbrace{\left(\begin{array}{c|c|c} \mathbf{y}_N & \mathbf{y}_W & \mathbf{y}_N \wedge \mathbf{y}_W \end{array}\right)}_{\mathbf{Y}} = \mathbf{R}.\underbrace{\left(\begin{array}{c|c|c} \mathbf{v}_N & \mathbf{v}_W & \mathbf{v}_N \wedge \mathbf{v}_W \end{array}\right)}_{\mathbf{V}}$$

Thus

$$\mathbf{R} = \mathbf{Y}\cdot\mathbf{V}^{-1}$$

If $\mathbf{R}$ is not exactly a rotation matrix, we can find the best rotation matrix close to $\mathbf{R}$ using a QR-factorization. The code is given below

```
def projSO3(M): # return a rotation matrix close to M
   Q,R = np.linalg.qr(M)
   return Q@diag(diag(sign(R)))
```

Algorithm MagCalib(In: $\mathbf{x}(i)$,$\mathbf{x}_N, \mathbf{x}_W$; Out :$\mathbf{R}, \mathbf{Q}$,$\bar{\mathbf{x}}$)

$I = 64 \cdot \frac{\pi}{180}$;

$$\mathbf{X} = \begin{pmatrix} x_1^2(1) & x_2^2(1) & x_3^2(1) & x_1(1)x_2(1) & x_1(1)x_3(1) & x_2(1)x_3(1) & x_1(1) & x_2(1) & x_3(1) \\ x_1^2(2) & x_2^2(2) & x_3^2(2) & x_1(2)x_2(2) & x_1(2)x_3(2) & x_2(2)x_3(2) & x_1(2) & x_2(2) & x_3(2) \\ x_1^2(3) & x_2^2(3) & x_3^2(3) & x_1(3)x_2(3) & x_1(3)x_3(3) & x_2(3)x_3(3) & x_1(3) & x_2(3) & x_3(3) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

$$\mathbf{p} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1} \cdot \mathbf{X}^{\mathrm{T}} \cdot \mathbf{1}$$

$$\mathbf{P} = \begin{pmatrix} p_1 & \frac{p_4}{2} & \frac{p_5}{2} \\ \frac{p_4}{2} & p_2 & \frac{p_6}{2} \\ \frac{p_5}{2} & \frac{p_6}{2} & p_3 \end{pmatrix}; \mathbf{c} = \begin{pmatrix} p_7 \\ p_8 \\ p_9 \end{pmatrix}$$

$\bar{\mathbf{x}} = -\frac{1}{2}\mathbf{P}^{-1}\mathbf{c}$; $\mathbf{Q} = \frac{\mathbf{P}}{1+\bar{\mathbf{x}}^T\mathbf{P}\bar{\mathbf{x}}}$

$\mathbf{v}_N = \sqrt{\mathbf{Q}} \cdot (\mathbf{x}_N - \bar{\mathbf{x}})$

$\mathbf{v}_W = \sqrt{\mathbf{Q}} \cdot (\mathbf{x}_W - \bar{\mathbf{x}})$

$$\mathbf{y}_N = \begin{pmatrix} \cos I \\ 0 \\ -\sin I \end{pmatrix}; \mathbf{y}_W = \begin{pmatrix} 0 \\ -\cos I \\ -\sin I \end{pmatrix}$$

$\mathbf{V} = \begin{pmatrix} \mathbf{v}_N & | & \mathbf{v}_W & | & \mathbf{v}_N \wedge \mathbf{v}_W \end{pmatrix}$

$\mathbf{Y} = \begin{pmatrix} \mathbf{y}_N & | & \mathbf{y}_W & | & \mathbf{y}_N \wedge \mathbf{y}_W \end{pmatrix}$

$\mathbf{R} = \mathbf{Y} \cdot \mathbf{V}^{-1}$

$\hat{\mathbf{A}} = (\mathbf{R}.\sqrt{\mathbf{Q}})^{-1}$, $\hat{\bar{\mathbf{x}}} = \bar{\mathbf{x}}$

Return $\hat{\mathbf{A}}, \hat{\bar{\mathbf{x}}}$

The calibration gives us the corrector $\hat{\mathbf{y}} = \hat{\mathbf{A}}^{-1} \cdot (\mathbf{x} - \hat{\bar{\mathbf{x}}})$.

2.3 Estimation of the North

If now, assume that the magnetometer is perfectly calibrated and returns $\hat{\mathbf{y}} = \mathbf{y}_1$, the corrected magnetic field expressed in the frame of the robot. Do not confuse with $\mathbf{y}_0$, the corrected magnetic field in the word's frame.

Assume that the calibrated magnetometer is far from the wall or other man-made structure. It returns the local magnetic field vector, *i.e.*,

$$\mathbf{y}_1 = \mathbf{B}_{\text{Earth}} + \underbrace{\mathbf{B}_{\text{induced}} + \mathbf{B}_{\text{remanent}}}_{=\mathbf{0}}$$

in the robot's body frame. We now have to find where is the North.

Recall that

- The magnetic field does not lie entirely in the horizontal plane.

- The vector points toward *magnetic North*, not geographic North.

The notion of North direction is defined in the horizontal plane. We must remove the component of the magnetic field that lies along the gravity direction.

Assume that a calibrated accelerometer provides the gravity direction $\mathbf{g}_1$ in the robot frame. To get the North, the magnetic field should be projected onto the horizontal plane orthogonal to gravity:

$$\mathbf{m}_h = \mathbf{y}_1 - (\mathbf{y}_1 \cdot \mathbf{g}_1)\mathbf{g}_1$$

As illustrated by Figure 2.5, when the boat is going to the North, it may see the magnetic field behind it. But the projected magnetic field $\mathbf{m}_h$ is ahead.

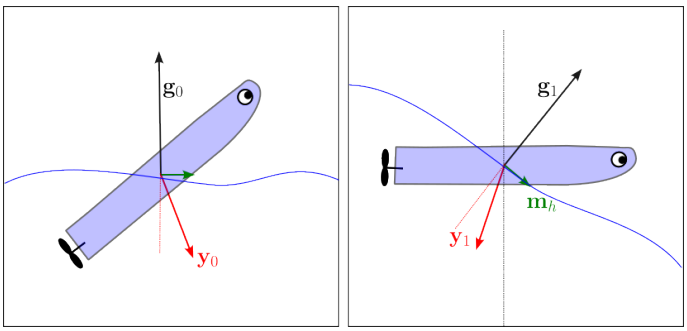

Figure 2.5: The boat sees the magnetic field behind it but the horizontal component $\mathbf{m}_h$ is in front of it

The horizontal magnetic field vector $\mathbf{m}_h$ points toward magnetic North. If the robot operates on a flat surface, the z component can be neglected.

$$\psi = -\text{atan2}(m_{h2}, m_{h1})$$

which corresponds to the magnetic North. To obtain true North, the magnetic declination δ must be added, *i.e.*,

$$\psi_{\text{true}} = \psi + \delta.$$

Near Brest, the declination $\delta = 1.4°$.

2.4 IMU

An IMU estimates all Euler angles from a magnetometer and an accelerometer, not only the heading ψ.

These Euler angles φ, θ, ψ have to be estimated from

1. the magnetic field $\mathbf{y}_1$ and

2. its acceleration $\mathbf{a}_1$, both given in the boat frame.

2.4.1 Estimation of the vertical axis

Assumption. In average the accelerometers returns the gravity vector $\mathbf{g}_1$ is the robot frame, i.e.,

$$\mathbf{a}_1 = \mathbf{g}_1 + \boldsymbol{\beta}$$

where $\boldsymbol{\beta}$ is a centered noise. The assumption is false if the boat turns around a buoy for long. We also assume that the sensors return $\mathbf{a}_1$ and $\boldsymbol{\omega}_1$ in the robot frame. The gravity vector $\mathbf{g}_1$ returns the vertical.

We have

$$\mathbf{g}_0 = \mathbf{R}\mathbf{g}_1$$

Differentiate with respect to t:

$$\mathbf{0} = \dot{\mathbf{R}}\mathbf{g}_1 + \mathbf{R}\dot{\mathbf{g}}_1$$

Thus

$$\begin{array}{rcl} \dot{\mathbf{g}}_1 & = & -\mathbf{R}^{\mathrm{T}}\dot{\mathbf{R}}\mathbf{g}_1 \\ & = & -\boldsymbol{\omega}_1 \wedge \mathbf{g}_1 \end{array}$$

We have

$$\begin{array}{rcll} \mathbf{g}_1(k+1) & = & \underbrace{(\mathbf{I} - dt\boldsymbol{\omega}_1(k)\wedge)}_{=\mathbf{A}(k)} \cdot \mathbf{g}_1(k) & \text{(prediction)} \\ \mathbf{a}_1(k) & = & \mathbf{g}_1(k) + \boldsymbol{\beta}(k) & \text{(observation)} \end{array}$$

with a large variance for $\boldsymbol{\beta}(k)$, which means that we measure $\mathbf{g}_1$ (through the acceleration $\mathbf{a}_1$), only in average.

A Luenberger observer could be

$$\hat{\mathbf{g}}_1(k+1) = \lambda \cdot (\mathbf{I} - dt\boldsymbol{\omega}_1(k)\wedge) \cdot \hat{\mathbf{g}}_1(k) \; + \; (1-\lambda) \cdot \mathbf{a}_1(k+1)$$

with $\lambda = 0.99$.

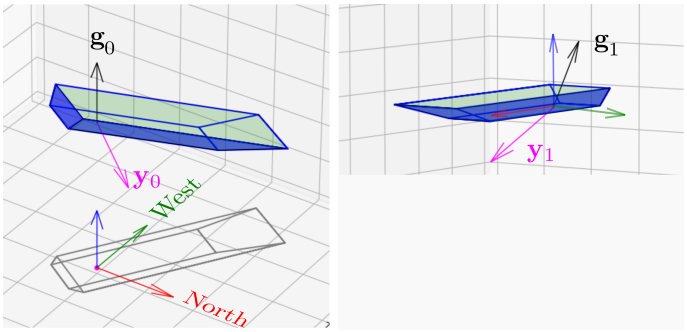

Figure 2.6: We now assume that we are able to have a good estimation of $\mathbf{g}_1$

2.4.2 Estimation of the Euler angles

For simplicity, we assume that the measured vectors $\mathbf{y}_1$ and $\mathbf{g}_1$ have been normalized, *i.e.*, their norm are both equal to 1. Moreover, we assume that sensors are well calibrated and are given in the body frame.

In the world frame $\mathcal{R}_0$, we have

$$\mathbf{y}_0 = \begin{pmatrix} \cos I \\ 0 \\ -\sin I \end{pmatrix}, \quad \mathbf{g}_0 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Step 1. From $\mathbf{g}_1$, we get an estimation of the bank φ (roll angle) and the elevation θ (pitch angle).

We have

$$\hat{\varphi} = \arcsin\left((0,1,0)\cdot \mathbf{g}_1\right)$$

and

$$\hat{\theta} = -\arcsin\left((1,0,0)\cdot \mathbf{g}_1\right)$$

Step 2. For the heading, we first compute the rotation which leads our boat to the nearest horizontal orientation.

If $\mathbf{u}$ and $\mathbf{v}$ are two unit vectors, the rotation $\mathbf{R}$ which minimizes $\|\mathbf{R}-\mathbf{I}\|$ and transforms $\mathbf{u}$ to $\mathbf{v}$ is given by

$$\mathbf{R}_{uv}(\mathbf{u},\mathbf{v}) = \mathrm{Exp}\left(\arccos(\mathbf{u}^{\mathrm{T}}\cdot\mathbf{v})\cdot\frac{\mathbf{u}\wedge\mathbf{v}}{\|\mathbf{u}\wedge\mathbf{v}\|}\right)$$

The rotation which transforms $\mathbf{g}_1$ to $\mathbf{g}_0$ is given by

$$\mathbf{R}_h = \mathbf{R}_{uv}(\mathbf{g}_1,\mathbf{g}_0).$$

Step 3. Provide an estimation $\hat{\psi}$ of the heading angle ψ.

We form the vector

$$\mathbf{y}_h = \mathbf{R}_h \mathbf{y}_1$$

which corresponds to the magnetic field measured by the horizontal avatar.

The heading is then obtained by

$$\hat{\psi} = -\text{atan2}\left(y_{h2}, y_{h1}\right).$$

	Algorithm EulerAngles(In: $\mathbf{a}_1, \mathbf{y}_1$; out:$\hat{\varphi}, \hat{\theta}, \hat{\psi}$)
1	$\mathbf{g}_0 = (0,0,1)^{\mathrm{T}}$,$\mathbf{g}_1^n = \frac{\mathbf{a}_1}{\|\mathbf{a}_1\|}$,$\mathbf{y}_1^n = \frac{\mathbf{y}_1}{\|\mathbf{y}_1\|}$
2	$\hat{\varphi} = \arcsin\left((0,1,0) \cdot \mathbf{g}_1^n\right)$
3	$\hat{\theta} = -\arcsin\left((1,0,0) \cdot \mathbf{g}_1^n\right)$
4	$\mathbf{R}_h = \mathbf{R}_{uv}(\mathbf{g}_1^n, \mathbf{g}_0)$
5	$\mathbf{y}_h = \mathbf{R}_h \mathbf{y}_1^n$
6	$\hat{\psi} = -\text{atan2}(y_{h2}, y_{h1})$

With magnetometers, accelerometers and without gyro

	Algorithm EulerAngles(In: $\mathbf{a}_1, \mathbf{y}_1, \boldsymbol{\omega}_1, \hat{\mathbf{g}}_1$; out:$\hat{\varphi}, \hat{\theta}, \hat{\psi}, \hat{\mathbf{g}}_1$)
1	$\mathbf{g}_0 = (0,0,1)^{\mathrm{T}}$, $\mathbf{y}_1^n = \frac{\mathbf{y}_1}{\|\mathbf{y}_1\|}$
2	$\hat{\mathbf{g}}_1 = \lambda \cdot (\mathbf{I} - dt\boldsymbol{\omega}_1\wedge) \cdot \hat{\mathbf{g}}_1 + (1-\lambda) \cdot \mathbf{a}_1$, with $\lambda \simeq 1$
3	$\mathbf{g}_1^n = \frac{\hat{\mathbf{g}}_1}{\|\hat{\mathbf{g}}_1\|}$
4	$\hat{\varphi} = \arcsin\left((0,1,0) \cdot \mathbf{g}_1^n\right)$
5	$\hat{\theta} = -\arcsin\left((1,0,0) \cdot \mathbf{g}_1^n\right)$
6	$\mathbf{R}_h = \mathbf{R}_{uv}(\mathbf{g}_1^n, \mathbf{g}_0)$
7	$\mathbf{y}_h = \mathbf{R}_h \mathbf{y}_1^n$
8	$\hat{\psi} = -\text{atan2}(y_{h2}, y_{h1})$

With magnetometers, accelerometers and gyro

Chapter 3

Practice with the DD-Boat

This chapter is partly taken from the lesson of B. Zerr [2].

3.1 Introduction

The DD-Boat is a small autonomous surface vehicle designed as an experimental platform for robotics education. It integrates sensing, computation, communication, and actuation in a real outdoor environment, making it particularly well suited for introducing students to embedded robotics and mobile navigation. You need to learn how to

- interact with an embedded robotic system through a network interface,
- deploy and execute Python programs on a remote onboard computer,
- acquire and log GPS data,
- understand practical issues related to communication latency, blocking I/O, and network failures.

The DD-Boat architecture follows a classical mobile robotics design:

- an onboard computer (Raspberry Pi) running Linux,
- sensors (GPS, IMU),
- motor controllers for propulsion and steering,
- a wireless communication interface (WiFi).

All user programs are executed directly on the onboard computer. The student laptop acts only as a development and supervision terminal, not as a real-time controller. This separation reflects standard practice in field robotics.

3.2 Building the Teams

Students work in teams to reflect real engineering workflows.

- Each team consists of 4 students.
- At least one team member must have a laptop with Ubuntu Linux, to ensure compatibility with SSH, SCP, and development tools.
- Each team could have access to one DD-Boat.
- Reach team has a name which corresponds to the family name of one person of the team.

It is recommended to assign roles such as:

- system operator (communication and deployment),
- software developer,
- hardware and mechanics,
- test and validation engineer.

3.3 Talking to the DD-Boat

Communication with the DD-Boat is achieved through a dedicated WiFi network. This network will probably be DARTAP. It provides a simple IP-based communication channel between the laptop and the onboard computer.

- The WiFi password for DARTAP is `ABACA`.
- Each DD-Boat has a static IP address: `172.20.25.2xx`, where xx is the two-digit boat number.

Using static IP addresses simplifies network configuration and avoids the need for service discovery mechanisms, which are often unreliable in outdoor environments.

Before executing any program, network connectivity must be verified.

```
ping 172.20.25.2xx
```

A successful ping confirms:

- the WiFi link is established,
- IP routing is correct,
- the onboard computer is powered and responsive.

SSH

SSH is used to remotely access and test the hardware components of a DD-Boat. We need to understand both the general principles of remote access and the practical commands used to verify that the boat's sensors and actuators are functioning correctly. SSH, which stands for *Secure Shell*, is a network protocol that allows a user to securely connect to another computer over a network. Once connected, the user can control the remote computer through a command-line interface as if they were physically sitting in front of it.

SSH is widely used because it encrypts all communications between the two machines. This means that passwords, commands, and data exchanged during the session cannot be easily intercepted or read by third parties. For this reason, SSH is the standard tool for remote administration of servers, embedded systems, and robots. SSH allows us to connect from our personal computer to the onboard computer (here a Raspberry Pi inside the DD-Boat) and execute programs that interact with the boat's hardware.

To connect to a DD-Boat, we run the following command from our own computer:

```
ssh pi@172.20.25.2xx
```

The username `pi` refers to the user account on the Raspberry Pi inside the DD-Boat. The system may ask for a password. Once authenticated, all subsequent commands are executed directly on the DD-Boat. At this point, the terminal on the student's computer acts as a remote terminal for the boat.

After logging in, we must navigate to the directory containing the hardware drivers:

```
$ cd git/drivers-ddboat-v2
```

The `cd` (change directory) command is used to move within the file system. The directory `drivers-ddboat-v2` contains the software programs that allow the Raspberry Pi to communicate with the boat's sensors and actuators, such as the GPS, IMU, and motors.

The following command is sometimes used to update the driver code:

```
$ git pull
```

This command downloads and integrates the latest changes from the remote Git repository into the local directory. In practice, the code does not change often, which is why this step is generally useless.

3.4 First program

After connecting with `ssh`, you are in the Linux of the DD-Boat. Write:

```
$ ls -l
```

to see the files. Each team creates its own directory:

```
$ mkdir DroneDupond
$ cd DroneDupond
```

where Dupond is the family name of the Team's leader. This directory-based organization avoids conflicts between teams and mirrors common multi-user development practices. The first Python program serves as a *deployment and execution test*. It is saved in `test.py` and contains

```
print("Hi!")
```

This trivial script verifies:

- Python is correctly installed on the DD-Boat,
- file transfer and execution permissions are correct,
- the `ssh` terminal correctly displays program output.

The file is transferred from your computer to the boat using:

```
$ scp test.py pi@172.20.25.2xx:DroneDupond/
```

3.5 Drivers

Each driver is implemented as an independent Python script. This modular approach makes the system easier to understand, test, and maintain. Several drivers are provided with the DD-Boat software. The inertial measurement unit (IMU) driver handles the magnetometer, accelerometer, and gyroscope. A second driver is dedicated to the GPS receiver and processes standard NMEA messages. Motor commands are sent through an Arduino board using a serial communication channel. Finally, a simple radio driver is available for monitoring purposes only, using a LoRa radio link. All drivers follow the same general philosophy: read data from a sensor or send commands through a serial interface, then expose this functionality to higher-level programs.

Installing and Testing the Drivers. To begin working with the DD-Boat drivers, you must first copy the software repository onto your laptop. The recommended approach is to clone the repository into a folder named `drivers`:

```
$ git clone https://gitlab.ensta-bretagne.fr/zerrbe/drivers-ddboat-v2.git
```

Equivalently, you can download these drivers here

```
https://webperso.ensta.fr/jaulin/drivers-ddboat-v2.zip
```

Once the repository is available locally, you can inspect the Python files and run test scripts to verify that each driver behaves correctly on your boat.

GPS. To test the GPS module, we run the following Python program:

```
$ python3 gps_driver_v2.py
```

This program communicates with the GPS sensor and displays the position data, such as latitude and longitude. If the GPS is working correctly, the measurements should appear consistently after a short initialization time. Once the first measurements appear correct, the program can be stopped by pressing `Ctrl+C`. This keyboard shortcut sends an interrupt signal that safely terminates the running program.

Note that the GPS measurements are:

- noisy (meter-level accuracy),
- low-frequency (typically 1–10 Hz),
- subject to dropouts in obstructed environments, like the ENSTA robotics pool.

Some additional Python packages are required to handle GPS data and coordinate transformations. They can be installed using:

```
$ sudo apt install python3-gpxpy python3-pyproj
```

The GPS receiver provides NMEA messages, in particular the `GPGLL` message, which contains latitude and longitude information. These coordinates are stored in a GPX file so that they can be visualized using mapping software. In NMEA format, coordinates are expressed as degrees and decimal minutes. To convert them into decimal degrees, the following formula is used:

$$\text{degrees} = DD + \frac{mm.mmmm}{60}$$

Once expressed in decimal degrees, coordinates can be projected into a metric system such as UTM. This makes it possible to compute distances in meters relative to a reference point.

IMU. The IMU (Inertial Measurement Unit) is tested using the following command:

```
$ python3 imu9_driver_v2.py
```

The IMU typically provides information about acceleration, angular velocity, and orientation. When this program is running, the displayed values should change if the boat is moved or rotated. This confirms that the motion sensors are operating properly.

Motor. To test the motors, the following command is used:

```
$ python3 arduino_driver_v2.py speed_left speed_right
```

In this command, `speed_left` and `speed_right` represent the speeds of the left and right motors, respectively. These values must be integers in the range from -255 to $+255$. Positive values correspond to forward motion, negative values to backward motion, and zero stops the motor.

By choosing different values for the two motors, it is possible to test forward motion, reverse motion, and turning behavior. As with the other tests, the motors can be stopped safely by pressing `Ctrl+C`.

3.6 Screen

A critical aspect of field robotics is dealing with unreliable communication.

- WiFi coverage is limited and intermittent.
- SSH terminals rely on continuous TCP connections.
- Blocking output operations (e.g. `print`) can freeze a program if the connection is lost.

This behavior is particularly dangerous for robots, as it can stop control loops. To avoid this, programs must be executed inside a detached terminal using `screen` (or similar tools such as `tmux`). This ensures that the program continues running even if WiFi is lost. Control and data acquisition remain autonomous. A screen session is started with a chosen name, for example:

```
$ screen -S sesddboat
```

This opens a new terminal managed by `screen`. You can verify that the session is active by listing existing sessions:

```
$ screen -ls
```

At any time, the session can be detached from the terminal using the key sequence `Ctrl+A` followed by `d`. The programs continue to run in the background. Later, when the WiFi connection is restored, the session can be resumed with:

```
$ screen -r
```

When the work is finished, the screen session can be completely stopped with:

```
$ screen -X -S sesddboat quit
```

Example: Logging GPS Data

As an example, a GPS logging program can be started inside a screen session:

```
$ screen -S sesgps
$ python3 tst_gpx.py
```

After detaching the session, the boat can move outside the WiFi coverage area while recording GPS data. Once finished, the session is resumed, the program is stopped, and the generated file can be renamed for later use.

3.7 Mapping and Data Visualization

GPS data can be exported to standard formats:

- KML (Google Earth),
- GPX (many GIS tools).

These formats allow post-mission analysis of trajectories, speed, and mission execution quality, which is a key step in experimental robotics. GPX files can be displayed directly on the GeoPortail website. For use with Google Earth, it is often more convenient to convert the file to the KML format:

```
$ gpsbabel -i gpx -f file.gpx -o kml -F file.kml
```

This conversion does not alter the data but allows compatibility with a wider range of visualization tools.

Bibliography

[1] T. Murata. Petri nets : properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[2] B. Zerr. Getting started with dd-boat. Lesson in Robotics, 2025.