Autonomous Sailboat Control System using STM32

Roberto Ibáñez Mingarro
Electronics & Automation Engineering (Double Degree: INSA Toulouse & Universitat Jaume I)

Abstract— This project consists in the complete embedded control development of an autonomous model sailboat platform using an STM32F103 microcontroller. The system integrates wind angle measurement, roll estimation, sail trimming via servo control, motorized yaw rotation, battery supervision, and wireless communication via XBee. The development was performed in C using Keil uVision5, relying exclusively on technical documentation and custom low-level driver implementation. The objective was to design a robust embedded architecture combining real-time acquisition, actuator control, interrupt management, and communication protocols.

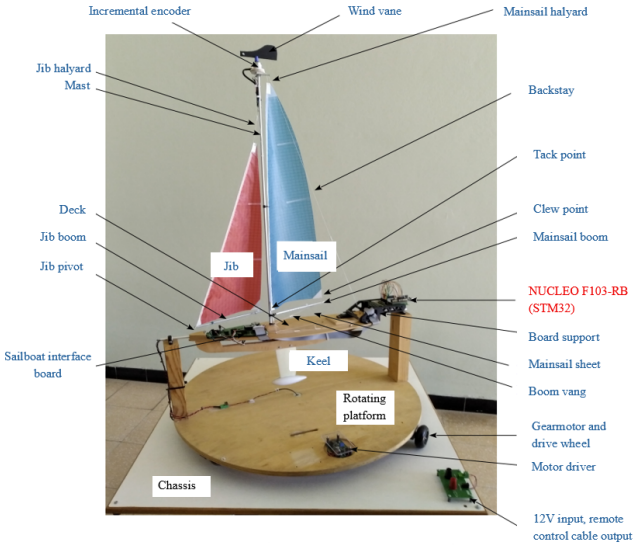


Figure 1: Complete autonomous sailboat platform with embedded electronics and sensors.

1 System Overview and Physical Platform

The educational sailboat platform used at INSA Toulouse is designed to emulate real-world automatic sail control. The system presents two degrees of freedom:

- 360° yaw rotation (motorized base),
- Free roll motion (longitudinal axis).

The objective of the embedded system is to automatically adjust sail trim according to wind direction while managing platform rotation and remote control communication.

The physical implementation of the platform, including mechanical structure, sensors, and embedded electronics, is illustrated in Figure 1.

2 Microcontroller Platform: STM32F103 (Nucleo F103-RB)

2.1 Why STM32F103?

The system is based on the STM32F103RB (ARM Cortex-M3), mounted on a NUCLEO-F103RB board.

This microcontroller was selected because:

- Cortex-M3 core with NVIC interrupt controller,

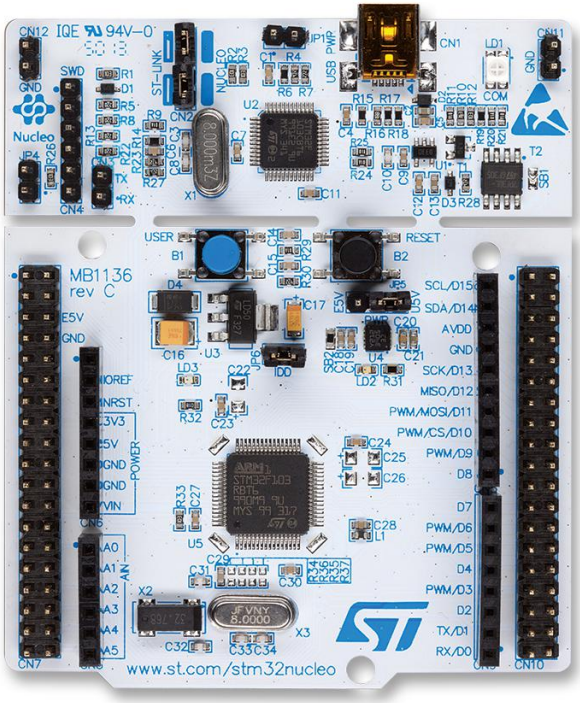


Figure 2: NUCLEO F103-RB board supporting the STM32F103 microcontroller.

- Multiple timers with PWM capability,
- SPI, I2C and USART peripherals,
- 12-bit ADC,
- Sufficient computing power for real-time control,
- Educational transparency (no abstraction layer required).

More powerful STM32 families (F4, H7) were intentionally not used. The objective was not raw computational performance but mastering low-level peripheral configuration, timing constraints, and deterministic embedded behavior on a constrained architecture.

The NUCLEO F103-RB board used as the central processing unit is shown in Figure 2.

2.2 Development Environment and Philosophy

Development was performed using **Keil uVision5**. This IDE provides:

- Detailed peripheral register visualization,
- Integrated debugger with breakpoints,
- Deterministic compilation toolchain,

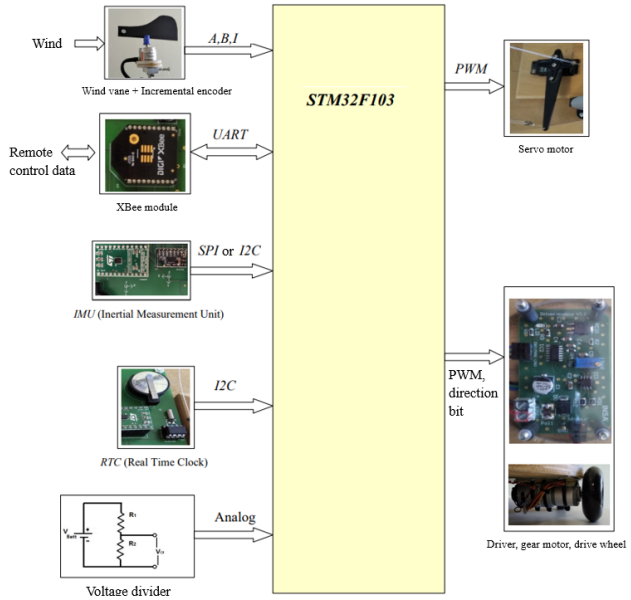


Figure 3: Global embedded architecture of the sailboat control system.

- Fine memory and linker control,
- Integrated CPU simulation environment,
- Real-time logic analyzer view,
- Instruction-level disassembly window,
- Call stack and execution trace,
- Live memory and variable tracking,

The project was written entirely in **C**, without HAL abstraction layers. All peripherals were configured manually via reference manual consultation.

Custom drivers were developed for:

- System clock configuration (RCC),
- NVIC priority management,
- GPIO configuration,
- Timer PWM generation,
- Timer input capture/compare & encoder interface,
- SPI communication,
- I2C communication,
- USART communication,
- ADC conversion,
- DMA configuration.

This approach ensured full hardware control, minimized memory footprint, reduced latency, and eliminated abstraction overhead, while fostering a deep understanding of clock architecture, register mapping, and real-time hardware constraints.

3 Global Electronic Architecture

The overall architecture integrates multiple sensing and actuation subsystems connected to the STM32 core.

The microcontroller interfaces with:

- Incremental encoder (wind angle),
- IMU (roll measurement),
- RTC via I2C,
- XBee module via USART,

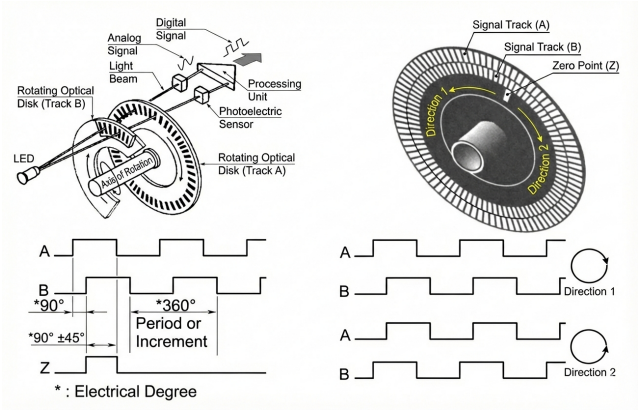


Figure 4: Wind vane and incremental encoder measurement chain.

- Servo motor (PWM),
- Motor driver (PWM + direction),
- Battery voltage divider (ADC).

The global interaction between sensing, actuation, power management, and communication subsystems is summarized in Figure 3.

4 Wind Angle Measurement (Incremental Encoder)

The wind vane is mechanically coupled to a quadrature incremental encoder providing signals A, B, and index I.

Resolution: 360 electrical periods per revolution.

Using quadrature decoding:

- Direction detection,
- 4x resolution (¼ degree precision),
- Signed angular position tracking.

Timer input capture combined with interrupt service routines allows deterministic pulse counting.

The quadrature measurement principle and signal relationships are illustrated in Figure 4.

5 Roll Measurement via IMU

Two accelerometer-based solutions were studied:

- LSM9DS1 (I2C),
- ADXL345 (SPI).

The two evaluated digital accelerometer modules are shown in Figure 5.

The roll angle is computed from gravity vector projection. SPI and I2C drivers were implemented from scratch, including:

- Register configuration,
- Multi-byte read sequences,
- Bus state management.

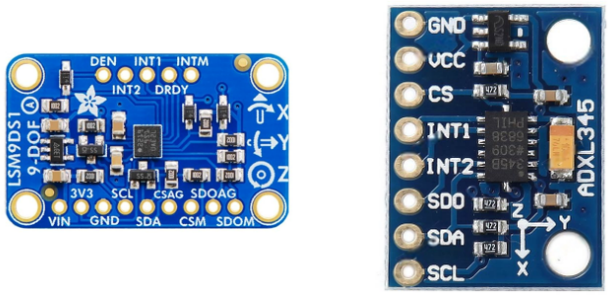


Figure 5: Accelerometer modules evaluated for roll angle estimation: LSM9DS1 (left, I2C interface) and ADXL345 (right, SPI interface).

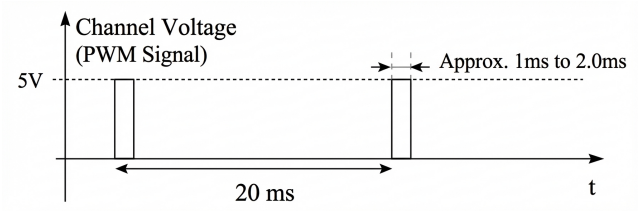


Figure 6: Standard RC servo control signal (20 ms period, 1–2 ms pulse width).

6 Sail Trim Control (Servo PWM)

Servo motors require a 20 ms period PWM signal:

- 1 ms pulse → minimum angle,
- 2 ms pulse → maximum angle.

TIM peripherals were configured in PWM mode with precise duty-cycle control.

The standard 20 ms period RC servo control signal used for sail trimming is shown in Figure 6.

7 Yaw Rotation Control (Motor Driver)

The motorized base requires:

- One digital direction signal,
- One high-frequency PWM (>20 kHz).

A dedicated power driver board interfaces between STM32 logic-level outputs and motor current stage.

The dedicated motor driver board and its associated power-stage schematic are presented in Figure 7.

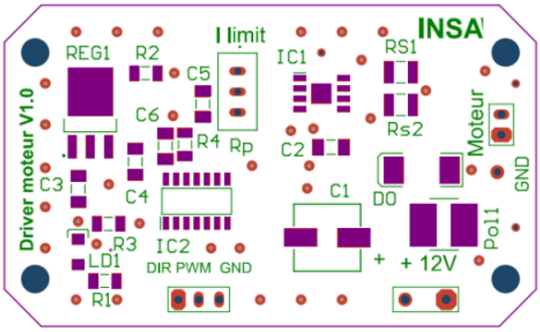
8 Wireless Communication (USART + XBee)

Communication with the remote controller is achieved via an XBee module operating at 2.4 GHz.

From the STM32 perspective, XBee behaves as a full-duplex UART interface.

- 100 ms periodic command reception,
- Bidirectional data exchange,
- Status reporting (battery, roll angle, timestamp).

USART interrupts were configured to handle reception without polling.



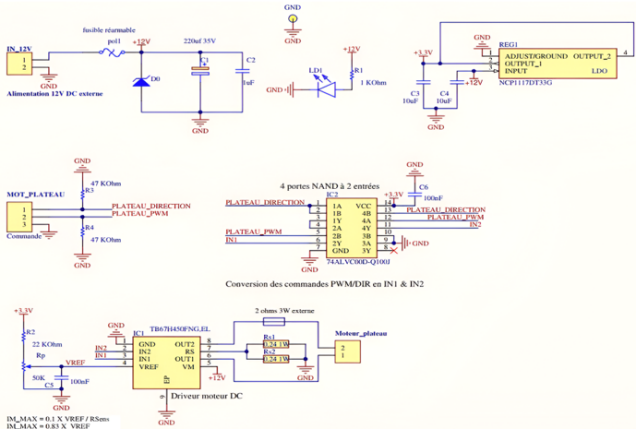


Figure 7: Motor driver PCB layout (top) and corresponding power stage schematic (bottom) used for yaw rotation control.

Figure 8: Digi XBee 2.4 GHz RF module used as UART-based wireless communication interface.

The Digi XBee module interfaced via USART for bidirectional wireless communication is shown in Figure 8.

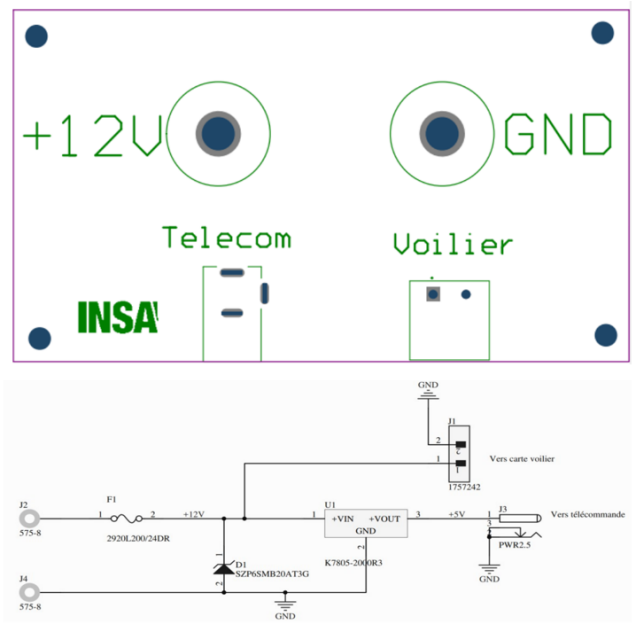


Figure 9: 12 V power input protection and 5 V regulation stage for embedded system supply.

9 Interrupt Architecture and NVIC Management

The embedded system relies heavily on interrupts:

- Encoder edge detection,
- USART reception,
- Timer overflow events,
- ADC conversion complete.

The NVIC was configured manually:

- Priority grouping,
- Preemption management,
- Latency optimization.

This ensured deterministic timing behavior and avoided blocking loops.

10 Power Architecture and Battery Monitoring

The system is powered by 12V supply.

Battery voltage is monitored via a resistor divider (ratio 1/13) connected to ADC input.

The dedicated 12 V power supply board and its associated regulation schematic are presented in Figure 9.

11 Interface Board

The microcontroller is mounted on a removable board allowing:

- Independent debugging,
- Oscilloscope probing,
- Flexible IO assignment.

The removable student interface board enabling flexible STM32 integration is shown in Figure 10.

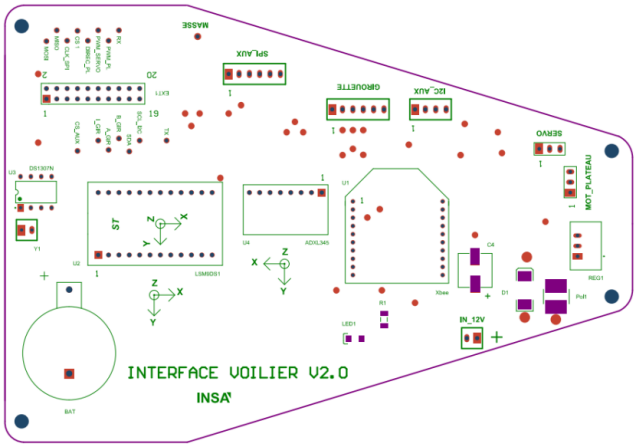


Figure 10: Custom interface PCB integrating STM32 Nucleo module, IMU, communication interfaces, and power distribution.

12 Engineering Outcomes

This project required:

- Complete peripheral-level firmware design,
- Real-time embedded architecture structuring,
- Power electronics integration,
- Communication protocol handling,
- Hardware–software co-design.

All firmware was developed entirely in **C** using **Keil uVision5**, without HAL abstraction. Custom low-level drivers were implemented for all utilized peripherals, including GPIO, RCC, timers (PWM and encoder mode), SPI, I2C, USART, ADC, DMA, and NVIC. Each peripheral was configured directly at register level using the STM32 reference manual, ensuring full transparency of hardware behavior.

Particular attention was given to clock tree configuration, interrupt prioritization, peripheral interactions, and memory-mapped register consistency to guarantee deterministic execution. Timing constraints were analyzed to ensure predictable behavior under concurrent interrupt sources, avoiding blocking loops and uncontrolled latency.

This approach minimized Flash and RAM footprint, reduced abstraction overhead, and provided precise control over hardware resources, while reinforcing a deep understanding of ARM Cortex-M3 architecture, bus arbitration mechanisms, and real-time embedded system design principles.

Although several circuit schematics, PCB layouts, and subsystem diagrams are presented, not all design artifacts are included. This portfolio article aims to convey the architectural complexity, integration challenges, and engineering depth of the embedded system rather than serve as exhaustive technical documentation.

Overall, this work represents a complete full-stack embedded implementation combining sensing, actuation, control logic, power management, communication protocols, and low-level firmware architecture on a constrained STM32F103 platform.