Tutorial do Zero — BlackPill F411 + L298N + OLED SSD1306 + IMU GY-88

*Objetivo: reproduzir do zero e entender o projeto (motores, OLED e IMU) usando PlatformIO e STM32CubeMX.*

# 0) Pré-requisitos

Hardware: BlackPill F411CE, L298N, OLED SSD1306 I²C (0x3C), IMU GY-88, ST-Link.

Software: VS Code + PlatformIO, STM32CubeMX.

# 1) Projeto no PlatformIO

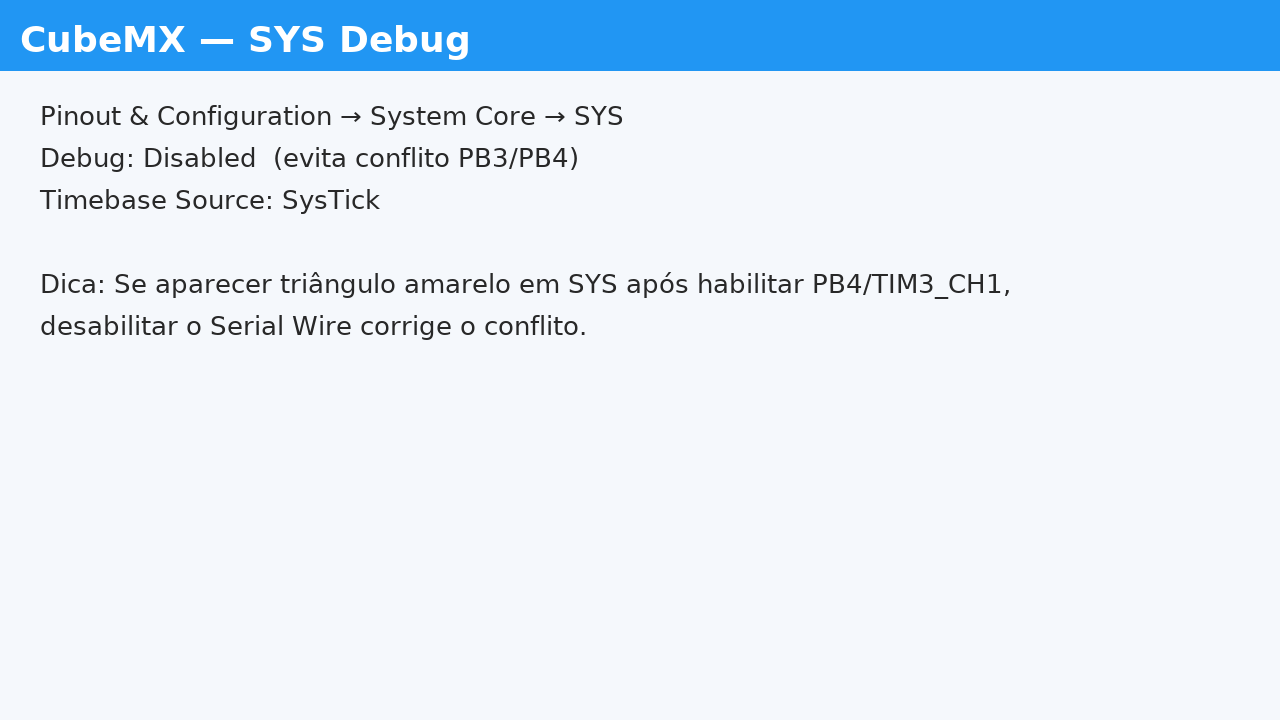
PlatformIO → New Project → Board: WeAct BlackPill F411CE; Framework: stm32cube; Nome: Carro\_Black\_Pill.

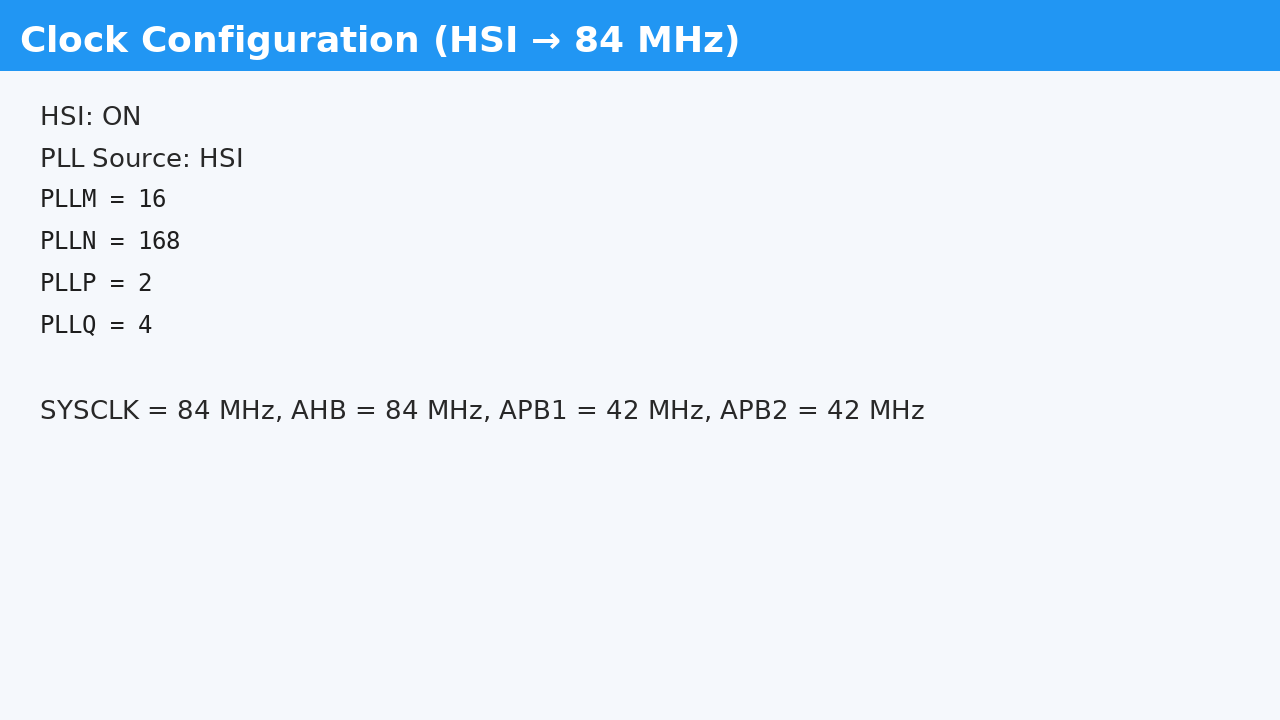
**platformio.ini:**

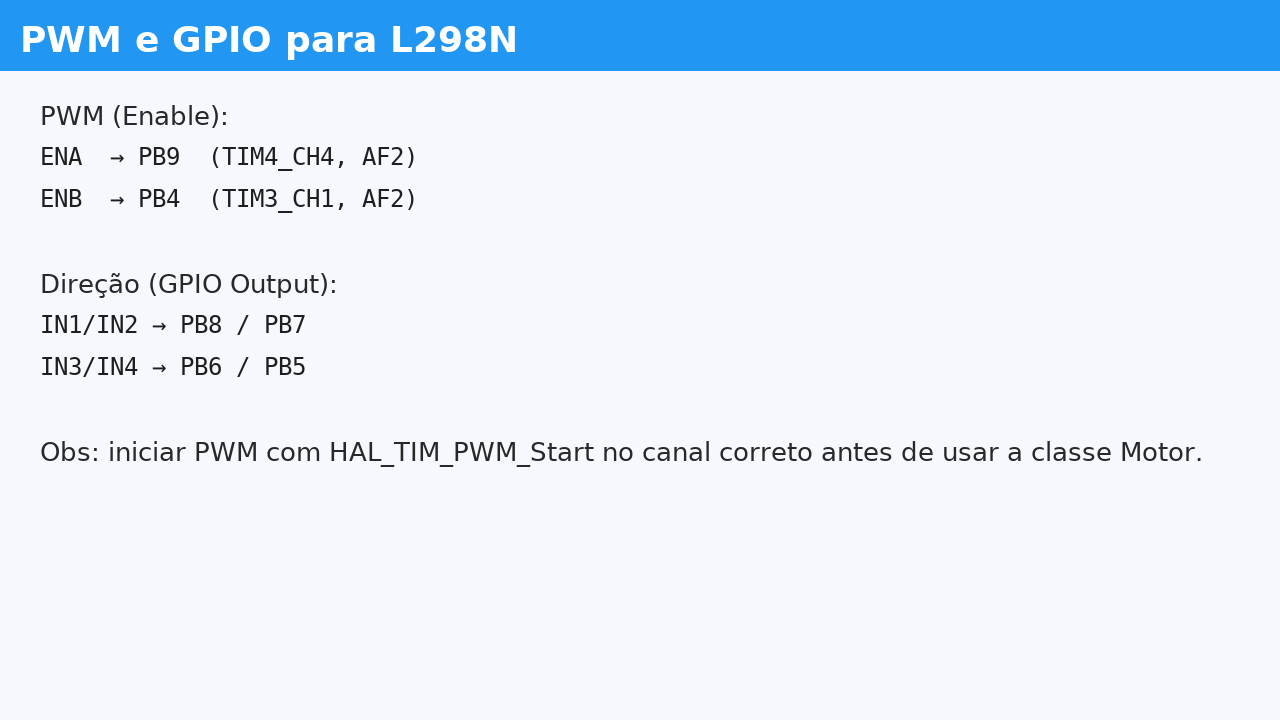
[env:blackpill\_f411ce]  
platform = ststm32  
board = blackpill\_f411ce  
framework = stm32cube  
upload\_protocol = stlink  
debug\_tool = stlink  
build\_flags =  
 -DUSE\_HAL\_DRIVER  
 -DSTM32F411xE  
 -DUSE\_OLED=1  
 -DUSE\_IMU=1  
 -DUSE\_MOTORS=1

# 2) Configuração no STM32CubeMX

Abra o CubeMX → New Project → STM32F411CEUx (UFQFPN48).









*Gere o código (Project Manager → Toolchain: Makefile).*

# 3) Trazendo os arquivos do Cube para o PlatformIO

Copie do projeto CubeMX para o projeto PlatformIO:

• Core/Inc/\* → include/ (stm32f4xx\_hal\_conf.h, stm32f4xx\_it.h)

• Core/Src/\* → src/ (stm32f4xx\_hal\_msp.c, stm32f4xx\_it.c, system\_stm32f4xx.c, syscalls.c, sysmem.c)

*O main.c do Cube será substituído por um main.cpp enxuto.*

# 4) Motores (L298N): pinos e classe

Classe `Motor` em lib/motor/: begin(), start(dir,duty), brake(), coast(). PWM em TIM4\_CH4 (PB9) e TIM3\_CH1 (PB4).

// Ex.: criação e uso  
Motor motorA(&htim4, TIM\_CHANNEL\_4, GPIOB, GPIO\_PIN\_8, GPIOB, GPIO\_PIN\_7, 999, false);  
Motor motorB(&htim3, TIM\_CHANNEL\_1, GPIOB, GPIO\_PIN\_6, GPIOB, GPIO\_PIN\_5, 999, false);  
motorA.begin(); motorB.begin();  
motorA.start(Motor::Direction::Forward, 60);  
motorB.start(Motor::Direction::Backward, 60);

# 5) OLED SSD1306

Lib vendor em C em lib/SSD306; wrapper opcional em C++ em lib/Display.

// ssd1306\_conf.h  
#define SSD1306\_USE\_I2C  
#define SSD1306\_I2C\_PORT hi2c2  
#define SSD1306\_I2C\_ADDR (0x3C << 1)

# 6) IMU GY-88

Drivers vendor em lib/GY88 e lib/GY88Utils; wrapper C++ e telas em lib/IMU.

// gy88\_conf.h  
#define GY88\_I2C (&hi2c2)

Formatação leve sem %f (include/text\_fmt.hpp):

void fmt\_signed\_fp(char\* buf, size\_t n, float v, int decs);  
// Ex.: fmt\_signed\_fp(tmp, sizeof(tmp), valor, 2);

# 7) Integração no main.cpp

int main(void){  
 HAL\_Init(); SystemClock\_Config();  
 MX\_GPIO\_Init(); MX\_TIM3\_Init(); MX\_TIM4\_Init(); MX\_I2C2\_Init();  
  
 // OLED  
 ssd1306\_Init(); ssd1306\_Fill(Black);  
 ssd1306\_SetCursor(0,0); ssd1306\_WriteString((char\*)"Init", Font\_7x10, White);  
 ssd1306\_UpdateScreen();  
  
 // IMU  
 gy88\_t imu; bool imu\_ok = (gy88\_init(&imu) == 0);  
  
 // Motores  
 Motor A(&htim4, TIM\_CHANNEL\_4, GPIOB,GPIO\_PIN\_8, GPIOB,GPIO\_PIN\_7, 999, false);  
 Motor B(&htim3, TIM\_CHANNEL\_1, GPIOB,GPIO\_PIN\_6, GPIOB,GPIO\_PIN\_5, 999, false);  
 A.begin(); B.begin();  
  
 while(1){  
 // ui::showAccel(...); ui::showGyro(...);  
 // A.start(...); B.start(...);  
 HAL\_Delay(50);  
 }  
}

# 8) Troubleshooting

• Motor não gira: PWM startado? AF2 nos pinos? INs como saída? GND comum.

• OLED só com labels: use fmt\_signed\_fp (sem %f); endereço 0x3C; I2C2 PB10/PB3 AFs.

• IMU FAIL: gy88\_conf.h aponta &hi2c2; endereços I2C válidos; cabos.

• Conflito PB4/SYS: desabilite Debug (Serial Wire) no Cube.