**Mobile Robot Platform**

[1. Components 1](#_Toc677957638)

[2. Documentation 2](#_Toc1601491998)

[3. STM configuration 4](#_Toc1243899973)

[3.1. Clock Configuration 4](#_Toc1216035534)

[3.2. Timer Configuration 5](#_Toc1977041761)

[2.2. GPIO pins configuration 6](#_Toc710801813)

[4. Architecture 6](#_Toc772781467)

[4.1. Pin Configuration 7](#_Toc17716148)

[4.2. Module Interfaces 8](#_Toc1874013538)

[4.2.1. Sensor Data Capture 8](#_Toc332750188)

[4.2.2. Obstacle Detection 9](#_Toc1241182198)

[4.2.3. Obstacle Detection 9](#_Toc56256803)

[5. Code 9](#_Toc191584465)

# Components

ECU (Electronic Control Unit) board; STMF303RE

4 motors

2 H bridges

4 Ultrasonic Sensors

# Documentation

**Motor**

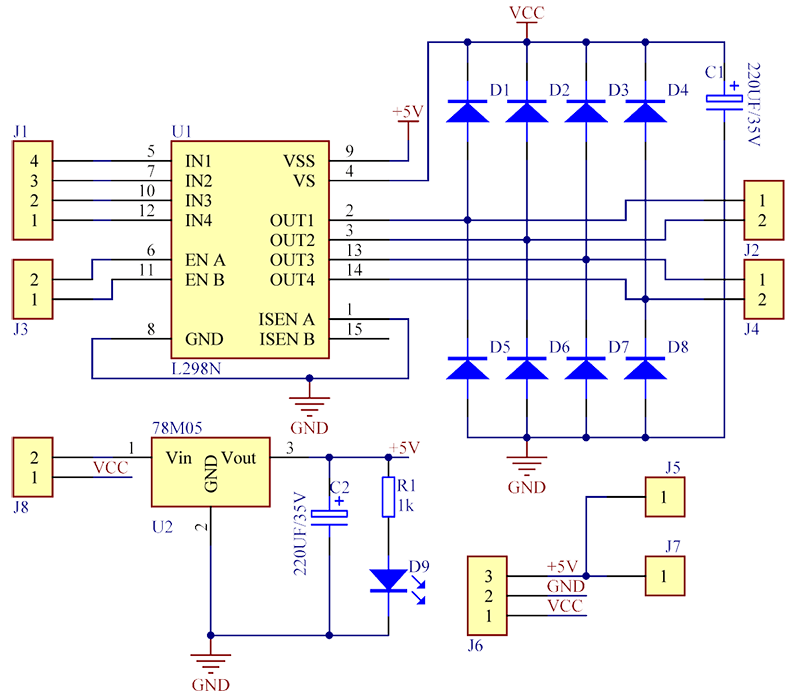
Descriere:

Diametru roată: 65mm;  
 Reducție motor: 1:48;  
 Tensiune de alimentare motor: 3V - 6V DC;  
 Cuplu: 0.8 kg \* cm;  
 RPM: 3V:125rpm, 5V:200rpm, 6V:230rpm;  
 Curent: 3V:60mA, 5V:100mA, 6V:120mA.

**H Bridge**

To control the speed of the motor, ENA pin is connecyed to a PWM signal.

|  |  |  |  |
| --- | --- | --- | --- |
| **ENA** | **IN1** | **IN2** | **Motor state (DC)** |
| 0 | X | X | Stop |
| 1 | 0 | 0 | Break |
| 1 | 0 | 1 | Clockwise rotation |
| 1 | 1 | 0 | Counter-clockwise rotation |
| 1 | 1 | 1 | Break |

Fig 1. Electronic Schematic of H bridge

**Ultrasonic sensor:**

Voltage: 5V  
HIGHV: 5V  
LOWV: 0V  
Angle: 15   
DIstanta detectata: 2cm - 450cm  
Precizie: 0.3cm



Fig 1. Electronic Schematic of Ultrasonic sensor

VCC supplies power to the HC-SR04 ultrasonic sensor. You can connect it to the 5V output from your Arduino.

Trig (Trigger) pin is used to trigger ultrasonic sound pulses. By setting this pin to HIGH for 10µs, the sensor initiates an ultrasonic burst.

Echo pin goes high when the ultrasonic burst is transmitted and remains high until the sensor receives an echo, after which it goes low. By measuring the time the Echo pin stays high, the distance can be calculated.

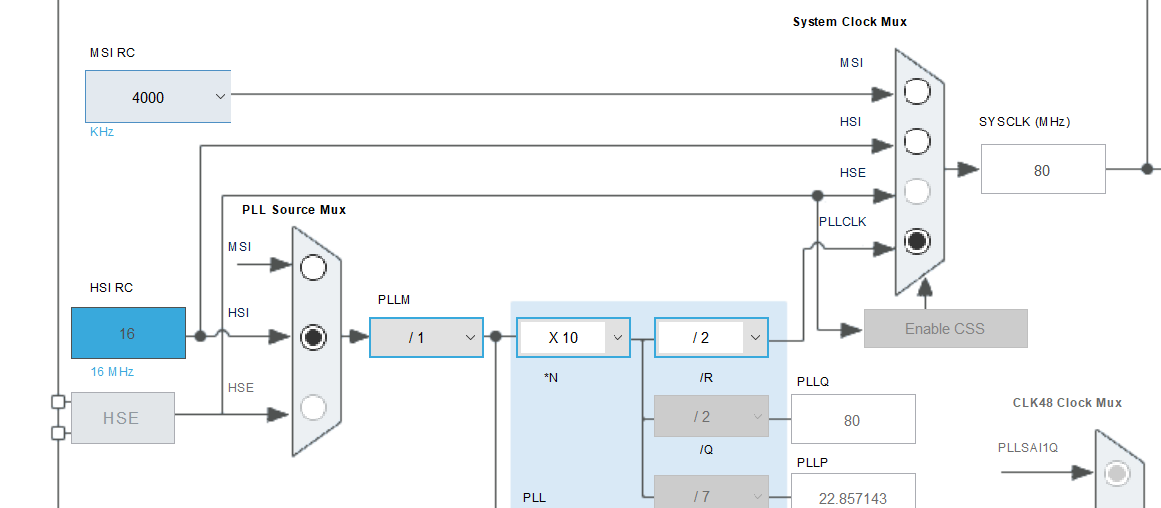
# STM configuration

### 3.1. Clock Configuration

For PLL source we use HSI with 16MHZ.

PPLM = 1; PPLN = 10; PPLR = 2;

SystemClock = HSIClock / PPLM \* PPLN / PPLR;

Fig. SystemClock Configuration

## 3.2. Timer Configuration

TIM2: Clock Source set to Internal Clock, Channel2 set to PWM Generation CH2

tim uptade frequency = TIM\_CLK/(TIM\_PSC+1)/(TIM\_ARR + 1)

EX:

PWM\_Freq = 80MHz

TIM\_CH2 prescaer =127

Counter Period 624

TIM\_CH2\_Freq = 80MHz/128/625=1KHz

**Timers:**

Frequency is determined by the value of the TIMx\_ARR register, and a duty cycle determined by the value of the TIMx\_CCRx register.

PWM mode can be selected independently on each channel (one PWM per OCx output) by writing 110 (PWM mode 1) or ‘111 (PWM mode 2) in the OCxM bits in the TIMx\_CCMRx register.

The user must enable the corresponding preload register by setting the OCxPE bit in the TIMx\_CCMRx register, and eventually the auto-reload preload register by setting the ARPE bit in the TIMx\_CR1 register.

OCx polarity is software programmable using the CCxP bit in the TIMx\_CCER register. It can be programmed as active high or active low. For applications where you need to generate complementary PWM signals, this option will be suitable for you.

To get TIM1 counter clock at 16 MHz, the prescaler is computed as follows:

Prescaler = (TIM1CLK / TIM1 counter clock) - 1

To get TIM1 output clock at 24 KHz, the period (ARR)) is computed as follows:

ARR = (TIM1 counter clock / TIM1 output clock) - 1

TIM1 Channel1 duty cycle = (TIM1\_CCR1/ TIM1\_ARR + 1)\* 100

Project Specific:

Inputs:

* TIM1 Channel 1 as input capture modefor Ultrasonic sensor.

Outputs:

* TIM2 Channel 1 as PWM Output mode for motor control.

## 2.2. GPIO pins configuration

Outputs:

* PA4 and PA5 for motor control.
* PA6 as Trigger for sensors.

DataSheets:

<https://ardushop.ro/img/cms/L298_H_Bridge_1.pdf>

[DS10362.pdf](file:///C:/Users/Ovidiu.suciu/STM32Cube/Repository/DS10362.pdf)

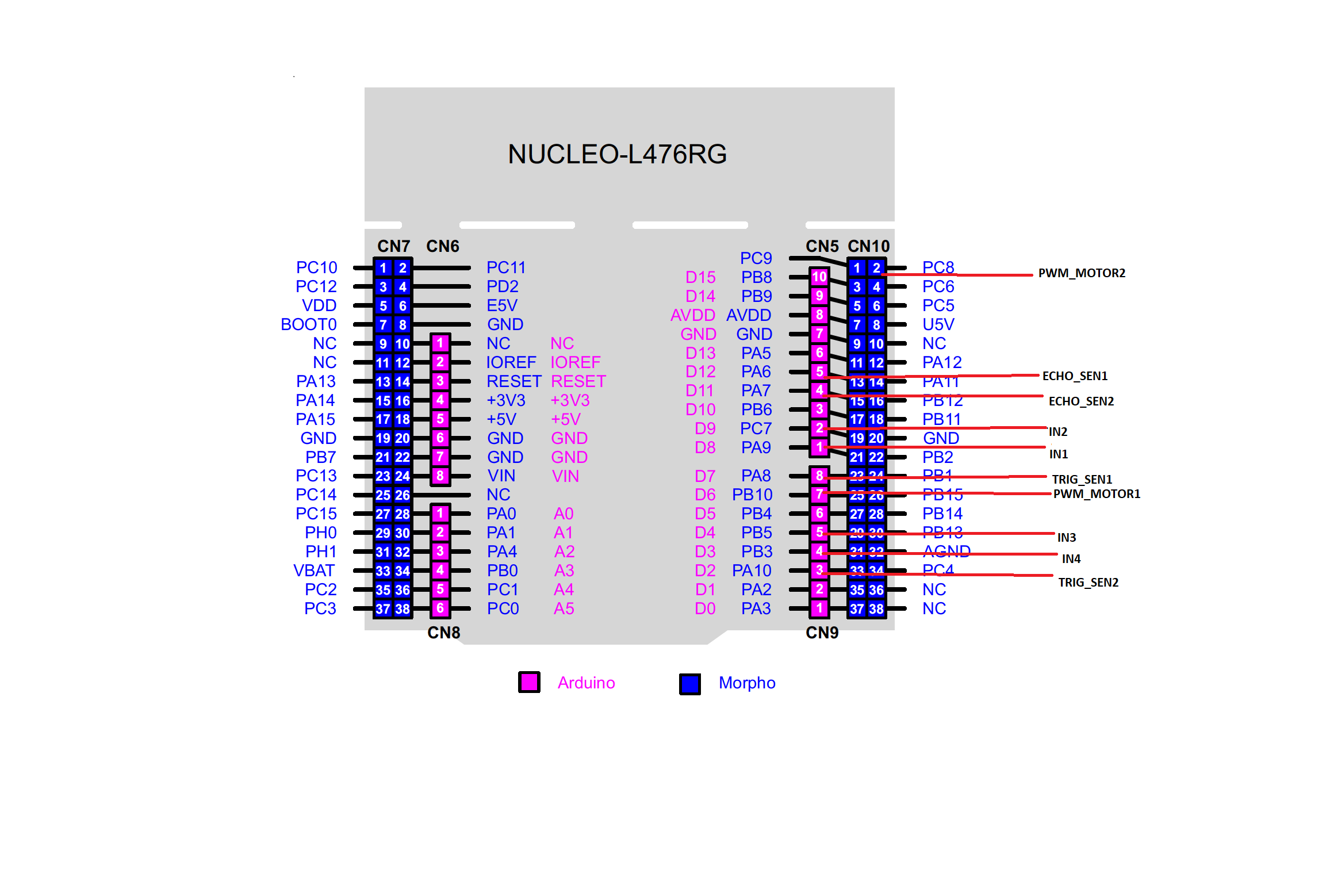
[STM32CubeL4GettingStarted.pdf](file:///C:/Users/Ovidiu.suciu/STM32Cube/Repository/STM32Cube_FW_L4_V1.18.0/Documentation/STM32CubeL4GettingStarted.pdf)

<https://github.com/Ltran0325/STM32-UART-Communication/blob/main/README.md>

# Architecture

## 4.1. Pin Configuration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pin configured | Pin Used | Description | Data Range/Signal | Direction |
| PWM\_MOTOR1 | PB10 | PWM for right motor control (TIM2\_CH3) | [0;12] V | Input |
| IN1 | PA9 | IN1 for right motor | GPIO PIN [0;1] | Input |
| IN2 | PC7 | IN2 for right motor  (TIM9\_Ch3) | GPIO PIN [0;1] | Input |
| PWM\_MOTOR2 | PC8 | PWN for left motor control | [0;12] V | Input |
| IN3 | PB5 | IN1 for left motor | GPIO PIN [0;1] | Input |
| IN4 | PB3 | IN2 for left motor | GPIO PIN [0;1] | Input |
| ECHO\_SEN1 | PA6 | Input Capture Timer  (TIM3\_CH1 |  | Output |
| TRIG\_SEN1 | PA8 | Enables right sensor | GPIO PIN [0;1] | Input |
| ECHO\_SEN2 | PA7 | Input Capture Timer  (TIM3\_CH2) |  | Output |
| TRIG\_SEN2 | PA10 | Enables left sensor | GPIO PIN [0;1] | Input |
|  |  |  |  |  |



## 4.2. Module Interfaces

### 4.2.1. Sensor Data Capture

1. **void** **HCSR04\_Read** (**void**)

* Triggers the sensor to start and enables the interupt

1. **void** **HAL\_TIM\_IC\_CaptureCallback**(TIM\_HandleTypeDef \*htim)

* Callback for timer interupt, where the data is saved in local buffers

1. HAL\_StatusTypeDef **Get\_SensorData\_Mapping**(SenData\_TypeDef \*mapping)

* Provides sensor mapping.

**DataTypes:**

**typedef** **enum**{

*POS\_FR*, /\*Front Right\*/

*POS\_FL*, /\*Front Left\*/

*POS\_BR*, /\*Back Right\*/

*POS\_BL*, /\*Back Left\*/

}SenPos\_TypeDef;

**typedef** **struct**

{

SenPos\_TypeDef orientation; /\* the position of the sensor\*/

uint16\_t Value; /\*Distance Value captured \*/

} SenData\_TypeDef;

### 4.2.2. Obstacle Detection

1. **void** **OD\_Main**(**void**)

* process the data from the sensors and decides the direction the car must take.

### 4.2.3. PWM Stub

1. **void** **Set\_PWM\_Pulse**(TIM\_HandleTypeDef \*htim, uint32\_t channel, uint32\_t pulse)

* This function provides the configuration of the PWM pulse specified by the timer instance “htim” and “channel”

### 4.2.4. Motors Control Service

1. HAL\_StatusTypeDef **MCS\_Init**(**void**)
2. **void** **MCS\_MainFunction**(**void**)

* Controls the motors based on the state and direction provided by the Obstacle Detection module

1. **void** **MCS\_SetDirection**(MCS\_Direction\_TypeDef Direction)

* This function provides the direction that the car needs to take.

1. **void** **Mcs\_SetState**(MCS\_State\_TypeDef state)

* This function provides the state of the car.

**DataTypes:**

**typedef** **enum**{

*MCS\_LEFT*,

*MCS\_RiGHT*,

*MCS\_NOTURN*

}MCS\_Direction\_TypeDef;

**typedef** **enum**{

*MCS\_FORWARD*,

*MCS\_BACKWARD*,

*MCS\_BREAK*,

*MCS\_LOCKED*,

*MCS\_ERROR*

}MCS\_State\_TypeDef;

# Code

1. Test code for motors:

int speed;

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_4, 0);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_5, 0);

//Rotatie in sensul acelor de ceasornic

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_4, 0);//IN1

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_5, 1);//in2

speeed = 500;

\_\_HAL\_TIM\_SET\_COMPARE(&htim2,TIM\_CHANNEL\_2, speed);

//frana

HAL\_Delay(200);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_4, 0);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_5, 0);

speeed = 500;

\_\_HAL\_TIM\_SET\_COMPARE(&htim2,TIM\_CHANNEL\_2, speed);

//Rotatie in sensul invers acelor de ceasornic

HAL\_Delay(200);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_4, 1);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_5, 0);

speeed = 500;

\_\_HAL\_TIM\_SET\_COMPARE(&htim2,TIM\_CHANNEL\_2, speed);

HAL\_Delay(200);

//frana

HAL\_Delay(200);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_4, 1);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_5, 1);

speeed = 500;

\_\_HAL\_TIM\_SET\_COMPARE(&htim2,TIM\_CHANNEL\_2, speed);

HAL\_Delay(200);