**Mobile Robot Platform**

[1. Components 1](#_Toc2091645535)

[2. Documentation 1](#_Toc2132453456)

[3. STM configuration 4](#_Toc214676910)

[3.1. Clock Configuration 4](#_Toc600349919)

[3.2. Timer Configuration 5](#_Toc293837552)

[4. Architecture 6](#_Toc296757614)

[4.1. Hardware specification and Functionality 6](#_Toc2006429269)

[4.2. Pin Configuration 7](#_Toc108626136)

[4.3. Module Interfaces 8](#_Toc1092984815)

[4.2.1. Sensor Data Capture 8](#_Toc1188781008)

[4.2.2. Obstacle Detection 9](#_Toc892601844)

[4.2.3. PWM Stub 9](#_Toc850047461)

[4.2.4. Motors Control Service 9](#_Toc706862504)

[4.3. RTOS 10](#_Toc1986244319)

[5. Code 10](#_Toc406192678)

# Components

ECU (Electronic Control Unit) board; STMF303RE

4 motors

2 H bridges

4 Ultrasonic Sensors

# Documentation

**Motor**

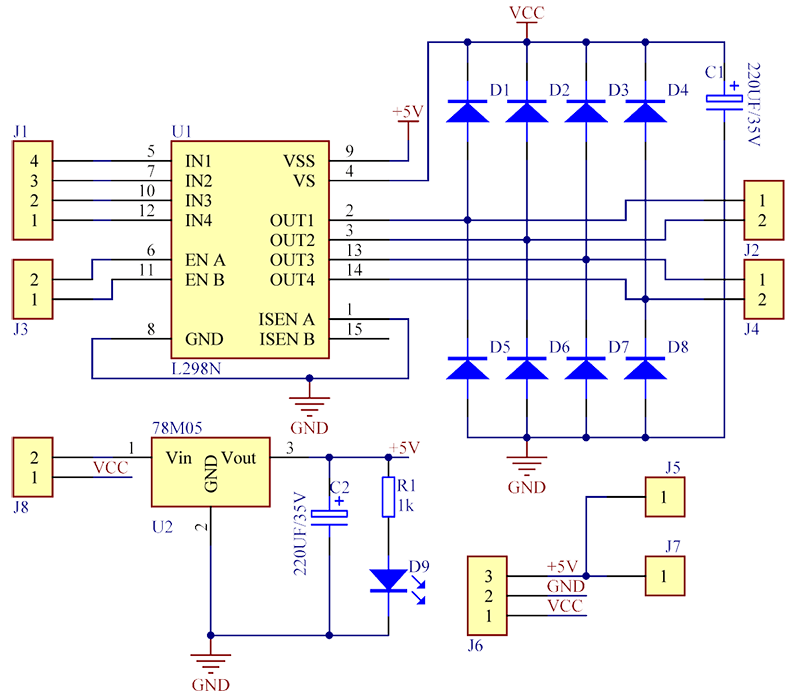
Descriere:

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:**

|  |  |
| --- | --- |
| Working Voltage | 3.3Vdc ~ 5Vdc |
| Working Current | 15mA |
| Working Frequency | 40Hz |
| Max Range | 4m ± 3mm |
| Min Range | 2cm ± 3mm |
| Measuring Angle | 15 degrees |
| Trigger Input Signal | 10uS TTL pulse |
| Echo Output Signal | Input TTL lever signal and the range in proportion |



Fig 1. Electronic Schematic of Ultrasonic sensor

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

Trig (Trigger) pin is used to trigger ultrasonic sound pulses. By setting this pin too 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, and the distance can be calculated.

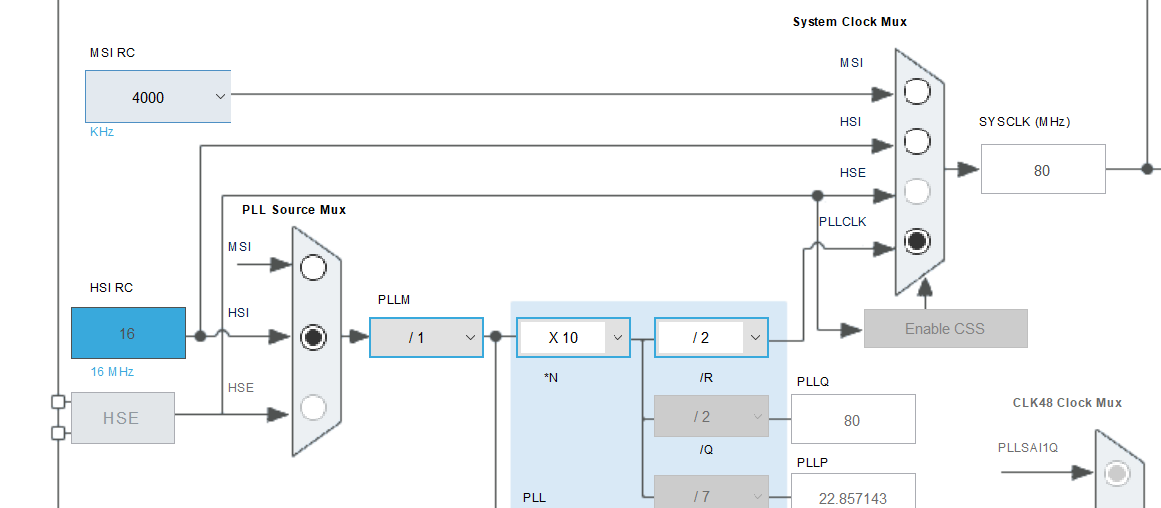
# STM configuration

### 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 to register by setting the OCxPE bit in the TIMx\_CCMRx register, and eventually the auto-reload preload registers 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

Datasheets:

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

<https://os.mbed.com/platforms/ST-Nucleo-L476RG/>

[Datasheet - STM32L476xx - Ultra-low-power Arm<Sup>® </Sup>Cortex<Sup>®</Sup>-M4 32-bit MCU+FPU, 100DMIPS, up to 1MB flash, 128 KB SRAM, USB OTG FS, LCD, ext. SMPS](https://www.st.com/resource/en/datasheet/stm32l476je.pdf)

# Architecture

## 4.1. Hardware specification and Functionality

**Hardware Specification:**

The platform is supplied by 2 accumulators of 3.6v each, which consist in a STM32L476RG development board, 2 H bridges, 4 DC motors and one ultrasonic sensor.

The H bridge is powered to the 7V supply and controls 2 motors at a time. To control the RPM for the motors a PWM signal is used. The duty cycle of this signal adjusts the voltage sent to the motor. The relation between the PWM signal and the supply input is defined by the following formula:

As for RPM here are some values corresponding to the input Voltage:

RPM: 3V:125rpm, 5V:200rpm, 6V:230rpm;

As for the ultrasonic sensor the VCC is connected to 5V Vin from the board. To enable the transmission, we send 10µs pulse on the Trigger pin. After the sensor is triggered it sends an 8-cycle sonic burst and listens for the reflected waves. We read this information from the ECHO pin and calculate the period in which the signal is high.

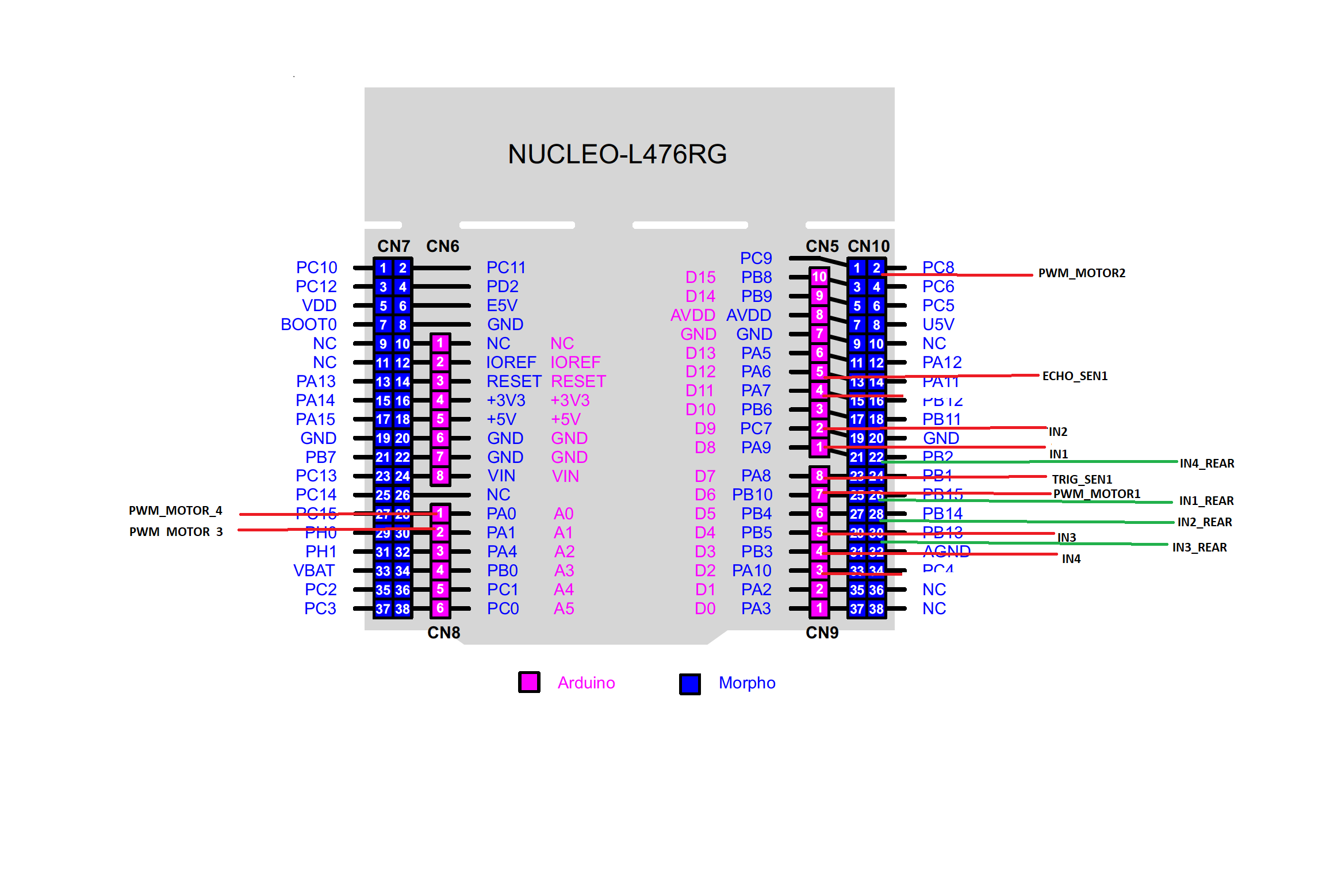
**Motor Control and Obstacle Detection:**

The platform is set by default to move forward until an obstacle is detected. For obstacle detection we set a threshold of 25cm. The mobile robot is programmed to turn right once he detects an obstacle. Whenever we must make a turn, the factor fill for the PWM\_MOTOR is set to 60% and the direction of the motors from that side is set backwards while the others are kept moving forward. Otherwise, on strait line we use a 50% fill factor.

## 4.2. Pin Configuration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pin configured | Pin Used | Description | Data Range/Signal | Direction |
| PWM\_MOTOR1 | PB10 | PWM for front right motor control (TIM2\_CH3) | [0;12] V | Output |
| IN1 | PA9 | Control direction front right motor | GPIO PIN [0;1] | Output |
| IN2 | PC7 | Control direction front right motor | GPIO PIN [0;1] | Output |
| PWM\_MOTOR2 | PC8 | PWN for front left motor control  (TIM8\_Ch3) | [0;12] V | Output |
| IN3 | PB5 | Control direction front left motor | GPIO PIN [0;1] | Output |
| IN4 | PB3 | Control direction front left motor | GPIO PIN [0;1] | Output |
| PWM\_MOTOR\_3 | PA1 | PWM for rear right motor control (TIM2\_CH2) | [0;12] V | Output |
| IN1\_REAR | PB15 | Control direction rear right motor | GPIO PIN [0;1] | Output |
| IN2\_REAR | PB14 | Control direction rear right motor | GPIO PIN [0;1] | Output |
| PWM\_MOTOR\_4 | PA0 | PWM for rear left motor control (TIM2\_CH1) | [0;12] V | Output |
| IN3\_REAR | PB13 | Control direction rear left motor | GPIO PIN [0;1] | Output |
| IN4\_REAR | PB2 | Control direction rear left motor | GPIO PIN [0;1] | Output |
| ECHO\_SEN | PA6 | Input Capture Timer  (TIM3\_CH1) | GPIO PIN [0;1] | Input |
| TRIG\_SEN | PA8 | Enables sensor | GPIO PIN [0;1] | Output |
| USART2\_Tx | PA2 | Communication with PC via USB |  | Output |
| USART2\_RX | PA3 | Communication with PC via USB |  | Output |

Table. Pin Configuration

Figure. Pin Schematic

## 4.3. Module Interfaces

### 4.3.1. 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;

### 4.3.2. Obstacle Detection

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

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

### 4.3.3. 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”

### 4.3.4. 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;

### **Interfaces in use:**

**void** **pwm\_start** (TIM\_HandleTypeDef \*htim, uint32\_t Channel)

* This function starts the PWM configured on the timers’ “htim” channel “Channel”.

**void** **pwm\_stop** (TIM\_HandleTypeDef \*htim, uint32\_t Channel)

* This function stops the PWM.

**void** **set\_direction** (GPIO\_TypeDef\* GPIO\_1,

uint16\_t GPIO\_Pin\_1,

GPIO\_TypeDef\* GPIO\_2,

uint16\_t GPIO\_Pin\_2,

uint8\_t direction)

* This function writes the GPIO pins for motor control based on the “direction” received as parameter.

**void** **set\_turn\_values** (uint8\_t turn\_direction,

uint16\_t\* right\_motor\_value,

uint16\_t\* left\_motor\_value,

uint8\_t\* direction\_left\_front,

uint8\_t\* direction\_right\_front,

uint8\_t\* direction\_left\_rear,

uint8\_t\* direction\_right\_rear)

* This function implements the state machine of motors control based on the input “turn\_direction” and provides the corresponding values.

## 4.4. RTOS

|  |  |  |
| --- | --- | --- |
| Tasks | Priority | Functionality |
| StartDefaultTask | 1 | Default Task. |
| UartTask | 1 | Sends the data from sensors to the PC |
| MotorControlTask | 1 | Function implementing the Motor Control thread. Decides the direction of the car based on the data from the sensor. |
| FrontUltrasonicSensorTask | 1 | Function implementing the Front Ultrasonic thread  Triggers the sensor transmission and calculates the distance. |

Table. Task configuration

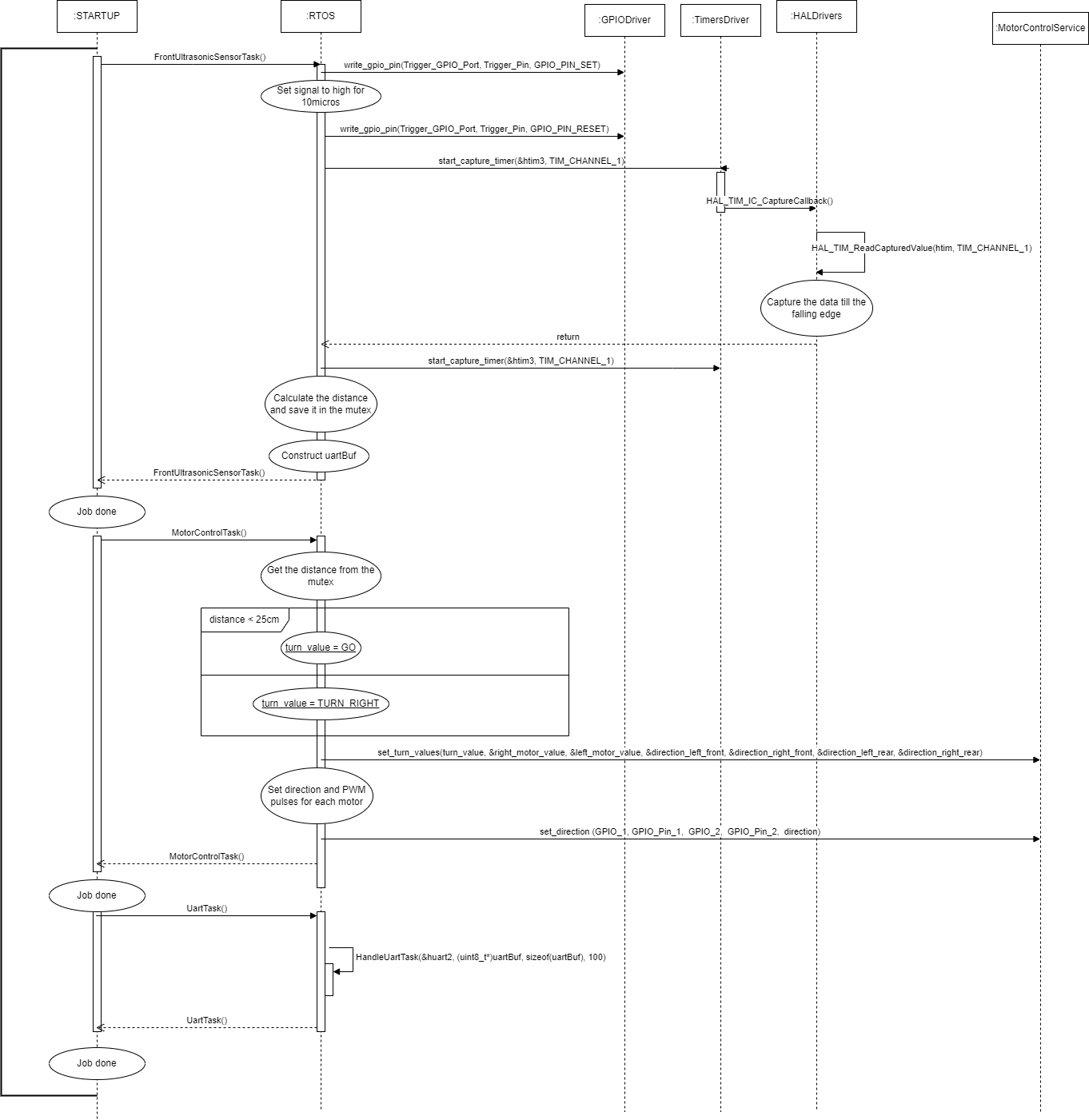


Figure. RTOS Sequence diagram