Tutorials

Table des matières

[How to use our API? 1](#_Toc472413748)

[Flash the STM32 with the Basic\_Car\_Control\_From\_Raspi project 1](#_Toc472413749)

[Check that all the wires are connected as it should be 1](#_Toc472413750)

[Create your application 1](#_Toc472413751)

[How to add a new functionality on the NUCLEO 1](#_Toc472413752)

[On which NUCLEO pin will the new sensor be plugged? 1](#_Toc472413753)

[How / where do I code the drivers? 1](#_Toc472413754)

[The syntax is important as you are expanding our existing API. 1](#_Toc472413755)

[How to add a new functionality on the PI3 1](#_Toc472413756)

[Getting started (NUCLEO board and NUCLEO Soft) 1](#_Toc472413757)

[Firsts steps: How to generate your own project using Curtis Libs ? 1](#_Toc472413758)

[Firsts steps: How to build your own project using Curtis Libs ? 1](#_Toc472413759)

[Firsts steps: How to connect Nucleo board to the car ? 1](#_Toc472413760)

[Firsts steps: How to execute the given examples  ? 1](#_Toc472413761)

[Extra tips – Commit a Keil project 1](#_Toc472413762)

[Common errors and solutions (NUCLEO board and NUCLEO soft) 1](#_Toc472413763)

[General soft problems 1](#_Toc472413764)

[pwm\_example.c 1](#_Toc472413765)

[motor\_example.c 1](#_Toc472413766)

[frontmotor\_example.c 1](#_Toc472413767)

[speedsensor\_example.c 1](#_Toc472413768)

[positionsensor\_example.c 1](#_Toc472413769)

# Getting started (NUCLEO board and NUCLEO Soft)

## Firsts steps: How to generate your own project using Curtis Libs ?

1. Retrieve the project environment generator by cloning <https://github.com/nsellam/curtisteamgit.git>
2. In the cloned folder, go to CurtisProject.
3. Launch CreateProjectFromTemplate.
4. A terminal window appears. Fill the project name.
5. Once generation is completed, you can find your new project in a folder named with the project name you have chosen.

## Firsts steps: How to build your own project using Curtis Libs ?

1. Launch using *Keil 4* your project.
2. Try to build your project by clicking “Build” (or pressing F7).
3. Building must terminate without any error or warning.

## Firsts steps: How to connect Nucleo board to the car ?

1. Plug the Nucleo board as described on PinAssignmen.xlsx file.
2. Plug through USB Nucleo board to your computer

## Firsts steps: How to execute the given examples  ?

By executing the given examples you will be able to check if the car is ready to be used. All the examples are provided with a short description of the expected behavior. If needed, please see “Debug” section to get help to fix the encountered problems.

1. Close *Keil 4*.
2. Copy the main.c and the example files you want to execute from Examples folder.
3. Paste these files in the Application directory of your project. You will need to replace main.c.
4. Launch your project using *Keil 4*
5. Add the files you need to your project. This can be done by using *Keil* interface:
   1. on the left panel (named “Project”), right click on “Application”
   2. then select “Add Existing Files to Group ‘Application’...”.
   3. an explorer window appears: you just need to go in your “Application” directory and select the files you want to add.
   4. once it is done, don’t forget to click “Add” then “Close”
6. Uncomment the line corresponding to the example you want to execute. It’s recommended to execute only one example at the time.
7. Once all the code is written, you can build and load your code on the STM32

## Extra tips – Commit a Keil project

1. If you are using Keil and Git, make sure to completely close Keil before performing a git pull. Similarly, each pull engenders a manual merge due to Keil files. For the conflict files, the easiest way to proceed is to systematically use the distant files (selecting “use their”).

# How to use our API?

## Flash the STM32 with the Basic\_Car\_Control\_From\_Raspi project

To do so, just open the folder “Basic\_Car\_Control\_From\_Raspi”.

You will find a subfolder named MDK\_ARM. In this subfolder, open the ‘.uvproj’ file.

Just compile the project and load it on a NUCLEOF103 RB board.

Once you have done this, the NUCLEO board is ready to periodically acquire data from:

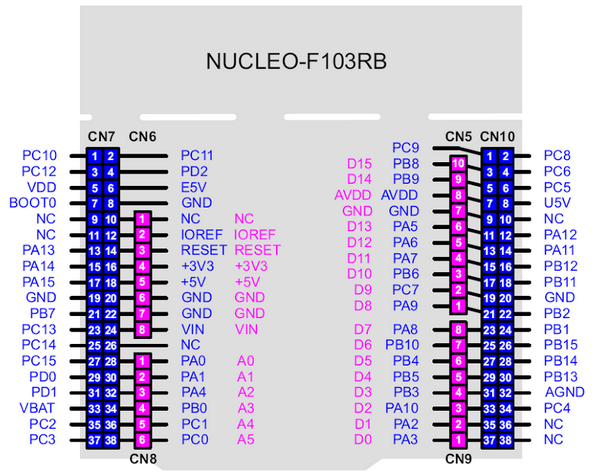
* Car speed
* Motors current
* Rear wheels Hall Sensors
* Front wheels Steering stop sensors

It is also ready to set a speed on the rear wheels (with a PI controller for the speed) and to turn right and left.

The NUCLEO board is ready to send its sensors acquisitions every millisecond via SPI and to receive actuators commands via SPI too.

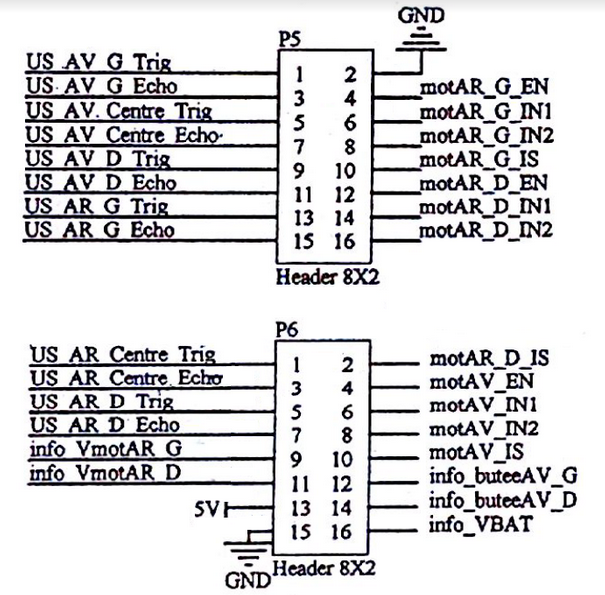
## Check that all the wires are connected as it should be

On the document “Hardware connectors – PinAssignement”, you will find the NUCLEO pins that are used by our API. Here is the NUCLEOF103RB pinout:



Now you have to connect each pin of the NUCLEO to its corresponding function on the car.

Here is the car pinout:



Car front

Car back

Let’s see with an example. On the document: “Hardware connectors – PinAssignement” you will find that MOT\_AR\_G\_EN (to enable the hardware PWM drivers on the car) is linked to PC 13.

It means you have to put a wire between MOT\_AR\_G\_EN pin on the car and PC13 on the NUCLEO.

**+ photo**

For the functions SCLK / MISO / MOSI (SPI communication with the PI) you have to wire as explained:

* Port PB13 on the NUCLEO wired with port BCM11 on the PI (SCLK)
* Port PB14 on the NUCLEO wired with port BCM9 on the PI (MISO)
* Port PB15 on the NUCLEO wired with port BCM10 on the PI (MOSI)

## Create your application

Now that everything is wired as it should and that the NUCLEO board is ready to use you can code your own application on the raspberry PI.

**+ comment faire? On a un thread mirroring a nous au moins. Expliquer comment / ou rajouter un thread a notre code.**

# How to add a new functionality on the NUCLEO

As an example, let’s assume you want to add an accelerometer to your battery of sensors.

You have to follow these steps:

## On which NUCLEO pin will the new sensor be plugged?

For the example of an accelerometer communicating via an i2c bus it means that you will use port PB8 for SDA and port PB9 for SCL (in addition to GND an VCC). The first thing you have to do is to make sure these ports are not used by another function. The excel document named “Hardware connectors – PinAssignement” lists all the ports that are already used by our basic API.

For PB8 and PB9, you are lucky, they are not used. But if the pin you want to use is already used by our API, follow these steps:

* Make sure you cannot use any other pin. If there is another pin you can use for the same function and that the API does not use just use this other pin and save time.
* If there is no other pin, please consult the STM32 documentation to see if you can remap the function used by the API on anther pin (that is still unused). After the remap has been made you will be able to use the pin you wanted to use.
* If no remap is available or if it is only available on a pin that is already used by the API, it means you cannot add your function to the API (at least not on the NUCLEO board). If this case happens try to add your function directly on the raspberry PI which has SPI, I2C peripherals, GPIOs too. For some specials functions (as PWM generation) you may need to buy external drivers (such as PWM drivers that you can control via I2C bus).

## How / where do I code the drivers?

You will have to develop your own drivers for the new functionality you want to add to the API.

These drivers will use some of the “STDPeriph” library functions or some of our API functions. Make sure to include the right headers to your source files.

You have to create a source file (.c) and a header file (.h) for your functionality and you should put these two files into the folder “API\_Extension\_Nucleo”. Usually a driver code contains:

**+ illustration de l’arborescence des fichiers**

* Initialization functions :

… nameOfYourSensor\_QuickInit(…) 🡪 ex : Accelero\_QuickInit(param1, param2, ..)

* Start and stop functions

nameOfYourSensor\_Startt(…) / nameOfYourSensor\_Stop(…)

* Calibration functions

nameOfYourSensor\_Calib(…)

* Callbacks (if you want to use systick interrupts

nameOfYourSensor\_Callback(…)

## The syntax is important as you are expanding our existing API.

* **How to return the data to the PI?**

One thing you should know at this point is that we use mirroring to communicate between the PI and the NUCLEO. It means we store information on a data structure that is the same on our NUCLEO code an on our PI3 code. Then we periodically send these data structures via SPI to update their content on both boards.

The NUCLEO board periodically updates information about the car sensors’ state and this information is sent to the PI that can update its “sensors’ info” field in local.

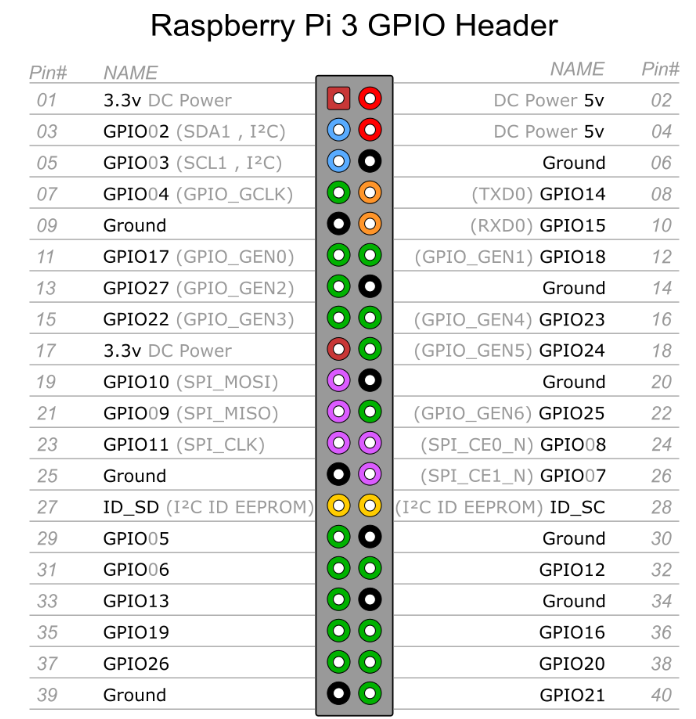
When a program wants to update a command, it writes on the corresponding field of the data structure the new value for this command. As commands are periodically sent to the NUCLEO, it has a local copy of the commands.

This way the PI that generates the commands also has a local copy of the sensors data, and the NUCLEO that acquires the sensors data also has a local copy of the commands. So the NUCLEO can update the motors commands.

On our data structures and SPI communication, we let some fields empty in case a user wants to exchange some more information. These slots are the fields **GIVE NAMES + CODE THAT.** So you only need to update one of these fields with the new data you want to send to the PI and it will automatically be sent via SPI.

# How to add a new functionality on the PI3

As you may know, you have a GPIO, SPI and I2C peripherals on the PI3. For now, our API only uses one of the two SPI channels available. This means that you can add some devices to the PI and add some functionalities to our existing API. Make sure you do not interface your new device on pins 19/21/23 (which are dedicated to SPI communication with STM32).



Your new drivers on raspberry PI3 can be added to the folder “API\_Extension\_PI3”

**+ illustration de l’arborescence des fichiers**

Once again, please make sure you use the same syntax as the existing API as you are making and API extension.

# 

# Common errors and solutions (NUCLEO board and NUCLEO soft)

## General soft problems

|  |  |  |
| --- | --- | --- |
| **Encountered problem** | **Cause** | **Solution** |
| Program compilation failed | Coding error(s) occur(s) | > Check if all the needed files are correctly included to your project. Read the section explaining how to add new files to your project.  *The easiest way to solve this kind of problem is to read the debug log.* |
| Program linking failed | Coding error(s) occur(s | > You probably used the mnemonics already used.  *The easiest way to solve this kind of problem is to read the debug log.* |
| Program loading failed | STM32 is not connected to your computer    STM32 is not recognized by your computer | > Connect STM32 through USB to your computer  > Your computer needs a few time to recognize STM32. Wait a while and try again.  > Needed drivers are not installed on your machine. Please read the section explaining how to install and configure *Keil 4* |

## pwm\_example.c

>> Expected behavior: Compilation and loading must be performed without any warning or error. PWM output is written on PA8 and its complementary on PA7. It is recommended to use the simulator view to observe the results easily.

>> Most common problems and potential solutions:

|  |  |  |
| --- | --- | --- |
| **Encountered problem** | **Cause** | **Solution** |
| Program compilation failed | *Please see section dedicated to program problems* | |
| Program linking failed |
| Program loading failed |
| Nothing happen | Please note that the only result is a PWM output on two pins. It is not observable directly on the car. | > Use *Keil* simulator and activate the “Analysis Windows”. By observing PA8 and PA7 you should observe something |
| Outputs stay stick to ‘0’ or are not displayed | *Keil* is not correctly configured | > Verify you are using Simulator mode. This can be done by opening “Options for Target”, and, in “Debug” tab, selecting the Simulator mode.  > Start Debug Session (Ctrl + F5)  > Display Analysis Windows  > In Logic Analyser Setup, make sure you are observing PA8 and PA7 and display type is ‘bit’. Pay attention to the zoom of Logic Analyser window. |

## motor\_example.c

>> Expected behavior: Compilation and loading must be performed without any warning or error. The motor you chose will spin forward (3”), stop (3”), spin forward (3”) and stops (to relaunch it you need to reset the board).

>> Most common problems and potential solutions:

|  |  |  |
| --- | --- | --- |
| **Encountered problem** | **Cause** | **Solution** |
| Program compilation failed | *Please see section dedicated to program problems* | |
| Program linking failed |
| Program loading failed |
| Motor stops spinning | This is a normal behavior | > You can reset the board to restart the program |
| Motor does not spin | Power is off  Motor interfacing is not correct    Keil is still in simulator mode | > Switch on the car by pushing the red button.  > Try to charge the battery (DO NOT USE THE CAR WHILE BATTERY IS CHARGING)  > Check pins assignment  > Set ST-Link as debugger. This can be done by opening “Options for Target”, and, in “Debug” tab, selecting the adequate Debugger. |
| Motor only spins forward | Motor interfacing is not correct | > Check pins assignment |
| Motor starts spinning backward | Motor interfacing is not correct | > Check pins assignment (motor pins must be exchanged) |

## frontmotor\_example.c

>> Expected behavior: Compilation and loading must be performed without any warning or error. Front wheels are successively oriented to right and left. Front motor is not supposed to force at any time.

>> Most common problems and potential solutions:

|  |  |  |
| --- | --- | --- |
| **Encountered problem** | **Cause** | **Solution** |
| Program compilation failed | *Please see section dedicated to program problems* | |
| Program linking failed |
| Program loading failed |
| Motor does not spin | Power is off  Motor interfacing is not correct    Keil is still in simulator mode | > Switch on the car by pushing the red button.  > Try to charge the battery (DO NOT USE THE CAR WHILE BATTERY IS CHARGING)  > Check front motor pins assignment  > Set ST-Link as debugger. This can be done by opening “Options for Target”, and, in “Debug” tab, selecting the adequate Debugger. |
| Motor forces when it arrives on its mechanical stops | Motor does not detect hard stops | > Check hard stops pins assignment  > Check if the hard stops (at the steering wheel base) are well positioned |

## speedsensor\_example.c

>> Expected behavior: Compilation and loading must be performed without any warning or error. The variable containing speed value is updating according to the speed of the associated wheel.

>> Most common problems and potential solutions:

|  |  |  |
| --- | --- | --- |
| **Encountered problem** | **Cause** | **Solution** |
| Program compilation failed | *Please see section dedicated to program problems* | |
| Program linking failed |
| Program loading failed |
| Nothing happen | Please note that the only result is the update of an internal variable. If you want to observe it, please use the Debugger. | > Use *Keil* debugger. If needed display the *Wach* *Window*. Type on the appropriate field the name of the variable you want to observe (here, WheelSpeed). |
| Even if wheels are rolling backward, speed is positive | This is a normal behavior | > This problem will no longer appear when you will use car in normal conditions (meaning when motors will roll). |
| Speed stays at zero | Wheel is not rotating    Power is off    Motor interfacing is not correct    Keil is still in simulator mode | > As motor is off, you need to manually turn the wheel.  > Switch on the car by pushing the red button.  > Try to charge the battery (DO NOT USE THE CAR WHILE BATTERY IS CHARGING)  > Check hall sensors pins assignment  > Set ST-Link as debugger. This can be done by opening “Options for Target”, and, in “Debug” tab, selecting the adequate Debugger. |

## positionsensor\_example.c

>> Expected behavior: Compilation and loading must be performed without any warning or error. The variable containing position value is updating according to the traveled distance of the associated wheel.

>> Most common problems and potential solutions:

|  |  |  |
| --- | --- | --- |
| **Encountered problem** | **Cause** | **Solution** |
| Program compilation failed | *Please see section dedicated to program problems* | |
| Program linking failed |
| Program loading failed |
| Nothing happen | Please note that the only result is the update of an internal variable. If you want to observe it, please use the Debugger. | > Use *Keil* debugger. If needed display the *Wach* *Window*. Type on the appropriate field the name of the variable you want to observe (here, WheelSpeed). |
| Even if wheels are rolling backward, position is increasing | This is a normal behavior | > This problem will no longer appear when you will use car in normal conditions (meaning when motors will roll). |
| Position stays at zero | Wheel is not rotating    Power is off    Motor interfacing is not correct    Keil is still in simulator mode | > As motor is off, you need to manually turn the wheel.  > Switch on the car by pushing the red button.  > Try to charge the battery (DO NOT USE THE CAR WHILE BATTERY IS CHARGING)  > Check hall sensors pins assignment  > Set ST-Link as debugger. This can be done by opening “Options for Target”, and, in “Debug” tab, selecting the adequate Debugger. |