Guide: Interfacing with CubeMars AK Series Motors

Contents

Introduction

This document introduces the control of CubeMars AK series motors using CubeMars software and STM32 microcontrollers. Below is a collection of useful resources, and on the following pages, the hardware setup will be described. This is followed by a quick-start guide to using the CubeMars software and, finally, a description of the developed code for STM32 microcontrollers.

Resources

CubeMars

AK80-9 specs and graph: https://www.cubemars.com/goods-982-AK80-9.html

AK80-9 specs and graph: https://uav-en.tmotor.com/html/2021/A_0106/640.html

AK Series Q&A: https://www.cubemars.com/article-262-AK+Series.html

AK Series Module Driver User Manual (included in the *Motor Control* folder): https://www.cubemars.com/images/file/20240611/1718085712815162.pdf

Software and video guides: https://www.cubemars.com/article.php?id=261

Download “CubeMars upper computer” from the last link and follow the guide in *CubeMars Software* to connect the motor. If you are unable to connect, try installing “CH340 Driver” on your PC. To get the motor running, follow the guide in *CubeMars Software* or watch video 2, “MIT mode control”. Make sure you follow all the steps.

STM32

NUCLEO-L476RG website: https://www.st.com/en/evaluation-tools/nucleo-l476rg.html

For easy access to convenient references, the following files have been collected in the “NUCLEO-L476RG” folder:

- Data brief for Nucleo-64 boards (nucleo-l476rg.pdf)
- User manual for Nucleo-64 boards (um1724-stm32-nucleo64-boards-mb1136-stmicroelectronics.pdf)
- Pinout for NUCLEO-L476RG (NUCLEO-L476RG Pinout.png)
- Screenshot of default pinout configuration (IOC Config Default.png)
- Screenshot of current pinout configuration (IOC Config 2024-10-28.png)
- Datasheet for STM32L476xx microcontrollers (stm32l476rg.pdf)
- Reference manual for STM32L4xxxx microcontrollers (rm0351-stm32l47xxx-stm32l48xxx-stm32l49xxx-and-stm32l4axxx-advanced-armbased-32bit-mcus-stmicroelectronics.pdf)

Hardware Configuration

Setup for CubeMars Software Control

To control the motor, the R-LINK is connected to the PC using a USB cable. The R-LINK is then connected to the CAN and Serial ports on the motor using the supplied cable.

The motor should then be connected to a power supply set to 24-48 V (also tested successfully at 18 V). The AK80-9 motor is rated for 12 A and a peak of 24 A, but a significantly lower current limit of a few amperes can be used as a precaution.

The setup with the AK80-9 motor and R-LINK is shown in Figure 1.

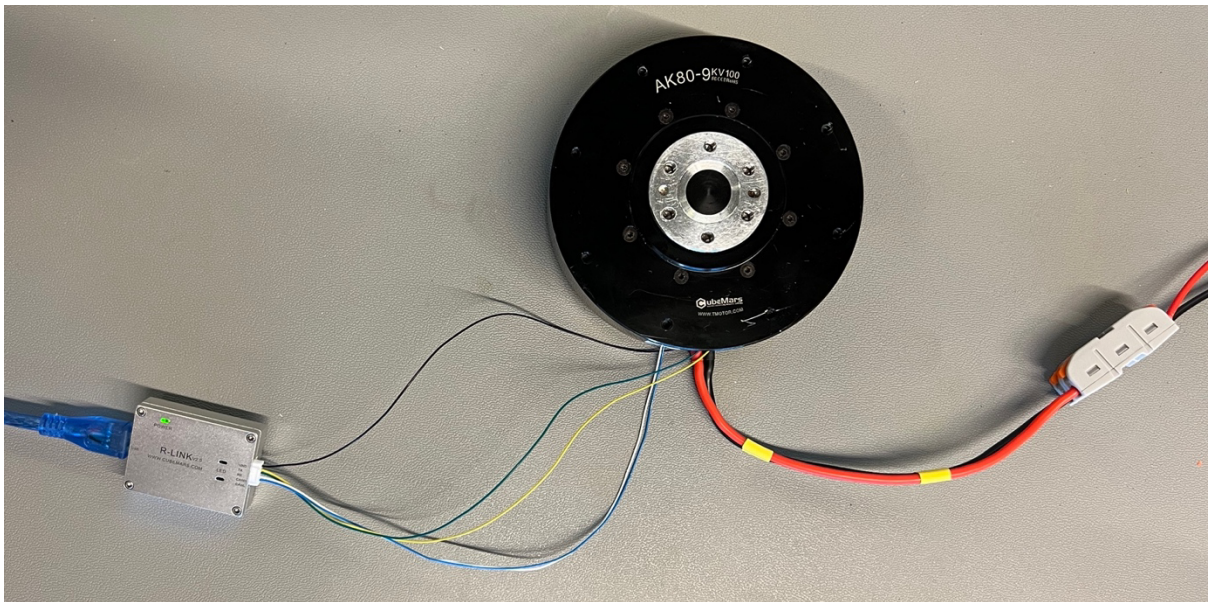

Figure 1: Setup of motor for test with CubeMars software.

Setup for STM32 Communication

The NUCLEO-L476RG development boards are connected to the PC using USB Mini-B cables. When working with multiple MCUs, it can be helpful only to connect one of them to the PC using a data cable while the other one is just connected to a power source. This avoids confusing the MCUs with each other when debugging.

The current input pinout configuration in the initialisation and configuration (IOC) tool (referred to as STM32CubeMX) is shown in Figure 2. Of main interest are the CAN RX and TX (upper left) and the Potentiometer (left).

The physical pinout of the Nucleo development board is shown in Figure 3.

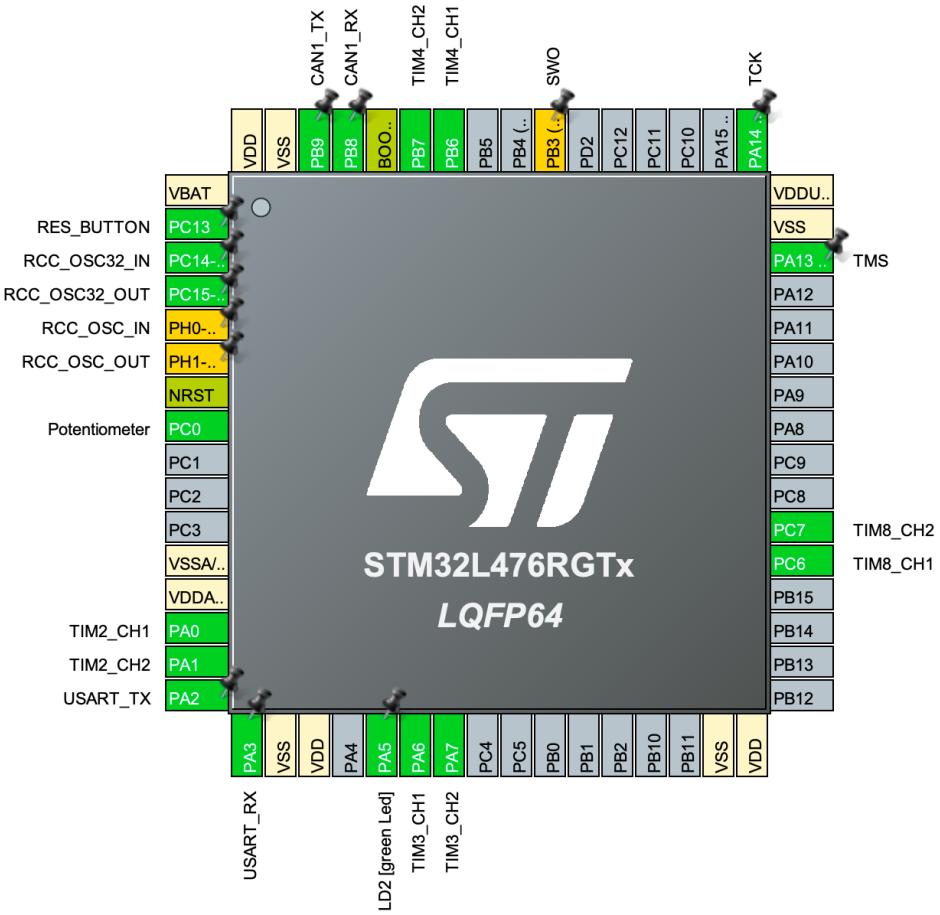

Figure 2: IOC pinout configuration viewed in STM32CubeIDE.

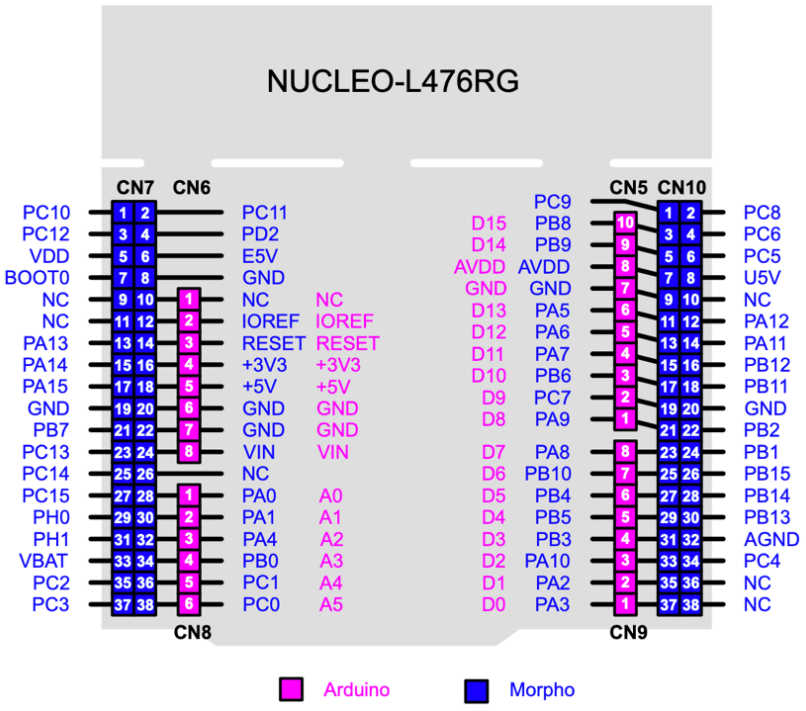

Figure 3: Physical pinout of NUCLEO-L476RG.

Referring to the physical pinout in Figure 3, the connections to be made on the board:

1. CAN transceiver Vcc: +5V (left)
2. CAN transceiver GND: GND (left)
3. CAN transceiver RX: PB8/D15 (upper right)
4. CAN transceiver TX: PB9/D14 (upper right)
5. Potentiometer Vcc (purple): +3V3 (left)
6. Potentiometer GND (grey): GND (left)
7. Potentiometer Signal (yellow): PC0/A5 (lower left)

To facilitate communication between two or more units, they should be connected CANH-CANH (blue-blue) and CANL-CANL (white-white).

The setup for communication between two Nucleo boards is shown in Figure 4.

The setup for testing motor control from a Nucleo board is shown in Figure 5. In this case, the Serial line can remain connected to a PC through the R-LINK if desired.

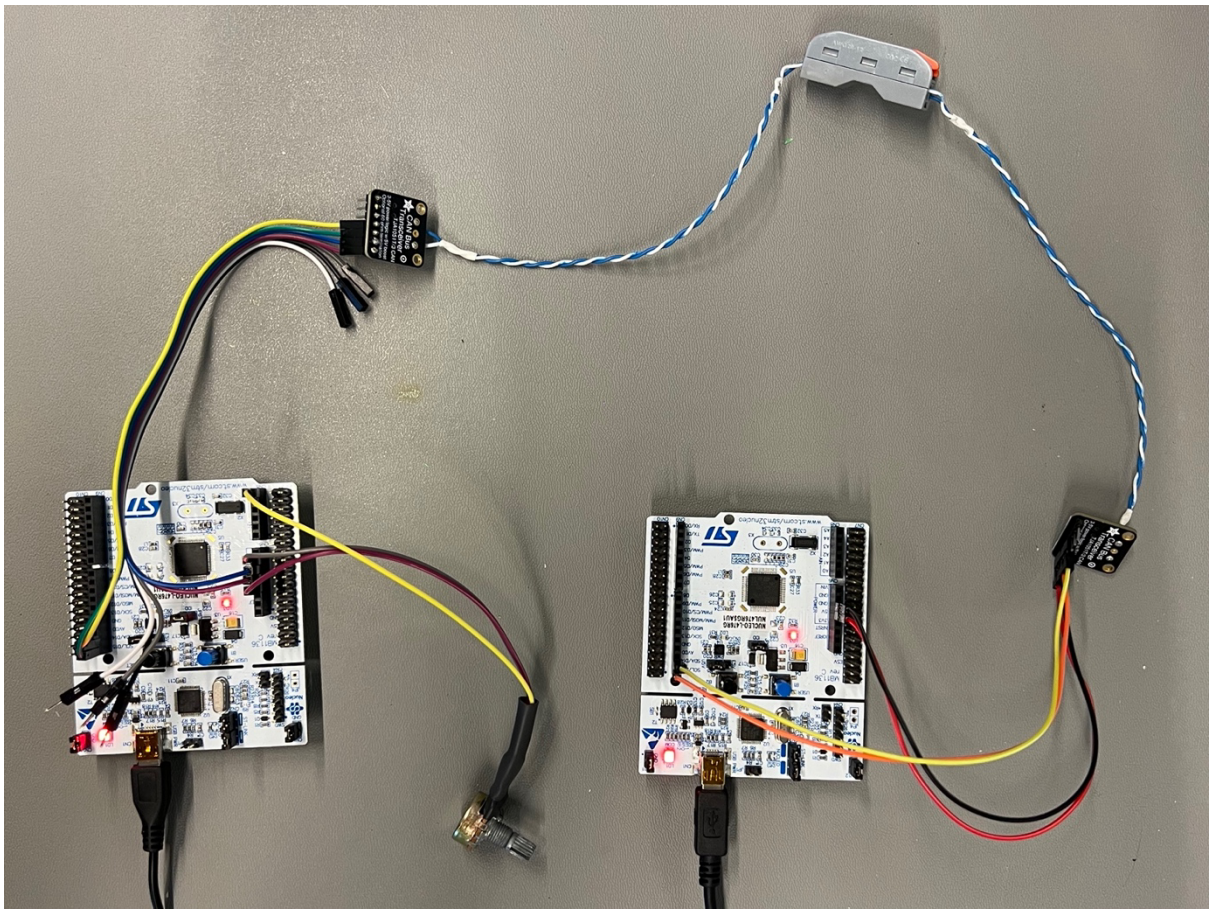

Figure 4: Setup of Nucleo boards for test of CAN communication.

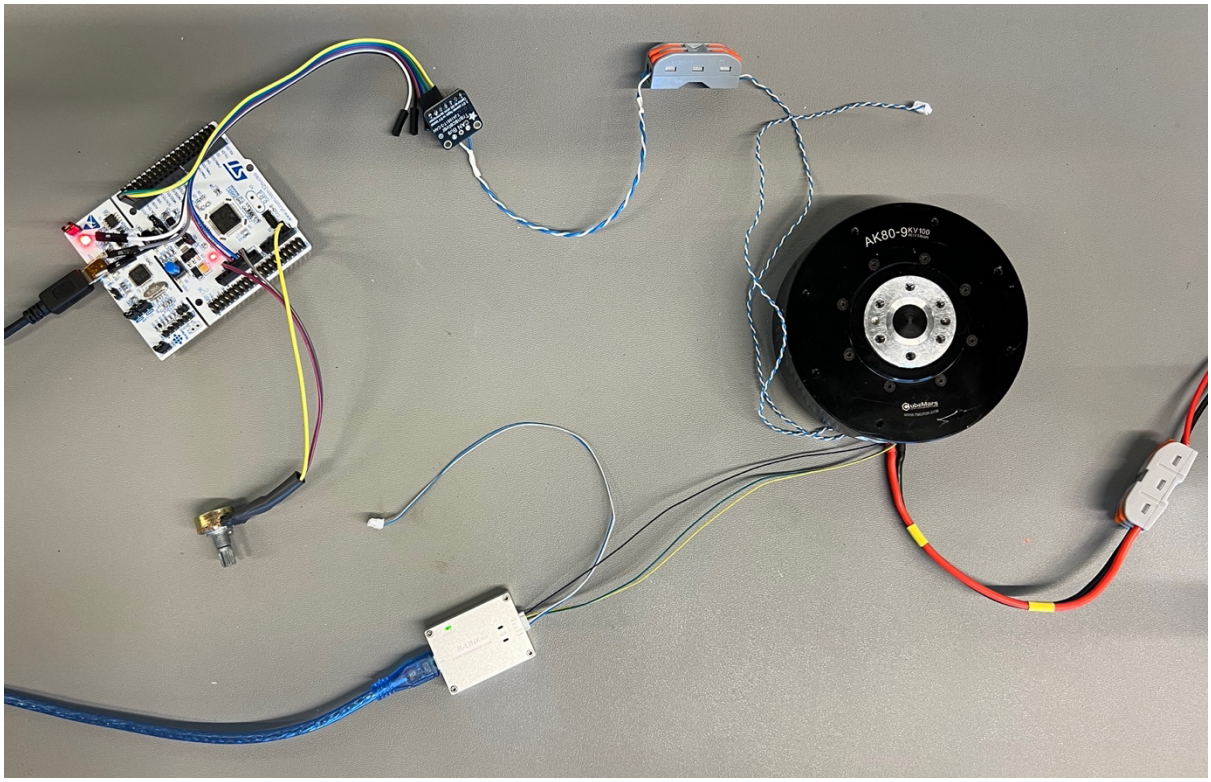

Figure 5: Setup of Nucleo board and motor for test of motor control.

CubeMars Software

The CubeMars software (referred to as the “upper computer”) can be downloaded from the link under *Resources* above. A screenshot of the software is shown in Figure 6.

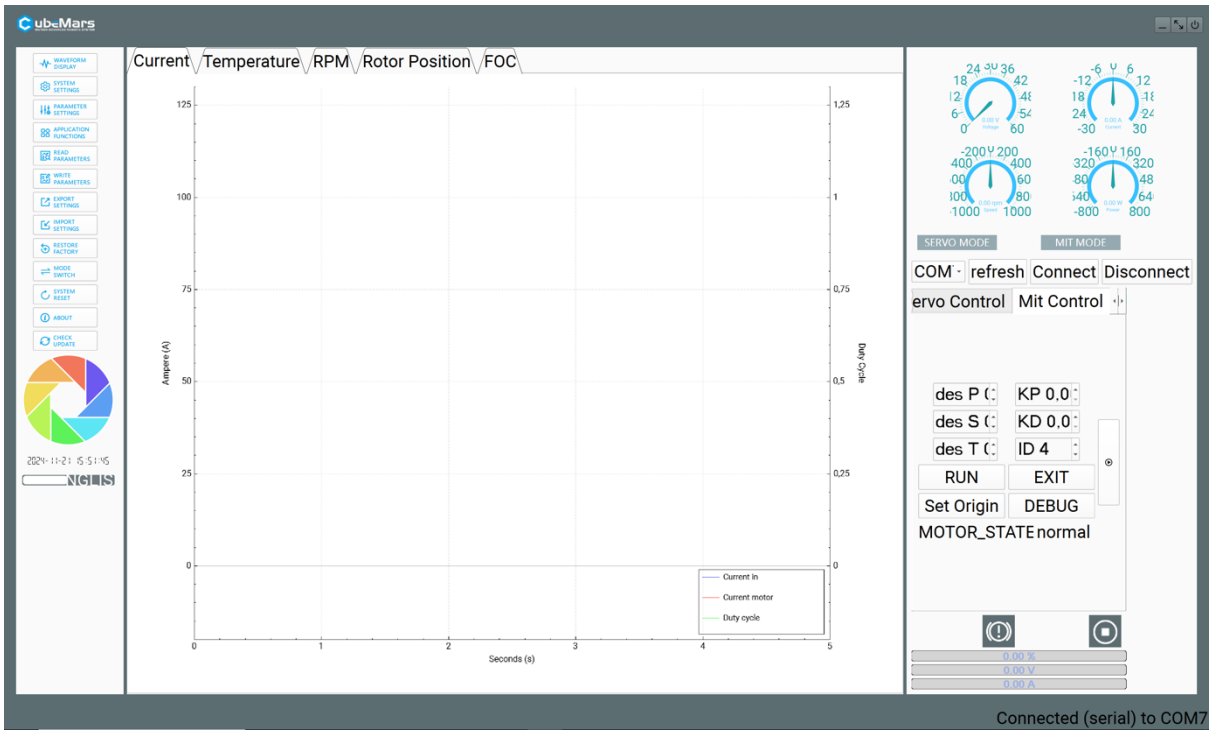

Figure 6: Screenshot of CubeMars software.

General Usage

The software is a bit quirky and will likely take some getting used to. Start by maximising the window and changing the language to English using the switch below the colourful circle on the left. The below procedure is also demonstrated in the video on GitHub.

Most controls are done in the panel on the right. To connect the motor, start by pressing “refresh”. This should display the relevant COM port in the dropdown, which can then be connected by pressing the “Connect” button. The status in the lower right corner should then change accordingly.

Two built-in control modes exist, “Servo Control” and “MIT Control”. Both can be used for position and/or speed control, but the difference lies in the control structure (refer to the user manual for block diagrams). In this guide, MIT mode will be used.

To control the motor in MIT mode, select the “Mit Control” tab on the right. First, set the motor’s CAN ID. (If you don’t know the motor’s CAN ID, see *Debug Panel* below.) Make sure all other values are zero. Then, press “RUN” to start continuous CAN

communication with the motor. At this point, you should see the graphs in the centre starting to display real-time data received via CAN. Press “Set Origin” to ensure the motor’s current position is zero before starting position control (forgetting this step can result in sudden, unexpected movement). Set the Kp (position control) and Kd (speed control) parameters and the position (“P”) and speed (“S”) setpoints. Press the vertical play button to send the command, and the motor should move accordingly. You will need to press the play button every time you want to send a command with new parameters or setpoints. In the centre, switch between tabs to see waveforms of position speed, temperature, duty cycle, DC current, FOC currents, etc. To stop control, press “EXIT”.

As a conservative starting point, Kp = 0.5, Kd = 0.2 and S = 0 can be used for position control. In this case, Kp controls the position to the P reference, and Kd limits the speed. For pure speed control, Kp = 0 and Kd = 0.2 can be used. In this case, the P reference has no effect, and Kd controls the speed to the S reference.

Additionally, it is possible to command a feed-forward torque (“T”), which translates directly to a current without associated feedback control.

Debug Panel

When the serial connection has been established (through the COM port, as described in *General Usage* above), the motor can be controlled using a CAN connection (though the CAN ID). If the CAN ID is unknown, it can be found through the Debug panel. This panel can also be used to read or set other parameters, to monitor encoder data and other things.

To see the configuration of the connected motor, first go to the debug panel by pressing the “DEBUG” button in the “Mit Control” tab on the right. This opens a command line-like interface for serial communication.

To read motor parameters, enter “setup”. This outputs a list of parameters, including the CAN ID. To return to the main debug menu, enter “exit”. Other usable commands are displayed in the main debug menu.

For More Information

In the left panel, graphical menus not covered in this guide can be opened, such as for settings, system parameters and exporting and importing settings. The “Mode Switch” menu can be used to switch between MIT mode and Servo mode.

For more information about using the CubeMars software, I recommend watching the video guides linked under *Resources* and referring to the user manual for detailed information about control modes and communication.

STM32

The code developed for the NUCLEO-L476RG is based on template code from UTS Motorsports. Minimal changes have been made to the existing code and no changes to the IOC configuration. This section will describe the additions to the code in general terms. For details, see the commented code.

Most additions have been made in the created files cubemars_control.c and cubemars_control.h. Many of these functions are based on examples from the CubeMars AK Series User Manual.

Float support for printf

For debugging purposes, printf is used to print values via USB to a serial monitor on your PC. Floating-point support for printf has been enabled to allow the display of floating-point numbers. However, enabling this feature can consume significant flash memory, RAM and CPU resources, which may negatively impact performance in critical real-time operations.

Recommendation: Before conducting critical testing or operation, you should strongly consider disabling floating-point support and generally limiting or eliminating the use of printf to minimise resource usage.

To enable floating-point support in STM32CubeIDE, the flag "-u _printf_float" was added under **Project Properties** > **C/C++ Build** > **Settings** > **Tool Settings** > **MCU GCC Linker** > **Miscellaneous** > **Other flags**. To disable floating-point support, simply remove this flag.

Includes in main.h

In the main.h file, two lines have been added under "USER CODE BEGIN Includes" where math.h and stdio.h have been included.

Structure of main.c

The subsections below will describe the additions made to the supplied main.c file.

Initialisation

In main.c, the cubemars_control.h file has been included. Additionally, some variables have been defined for storing the state of the push-button for debugging purposes, defining CAN IDs for sending commands and interpreting feedback, as well as variables for storing motor commands and feedback data.

While Loop

In the while loop in the main function, different sections of code can be commented out or uncommented to change the functionality.

The first section reads data from the ADC. It was found that it was necessary to call HAL_ADC_Start(&hadc1) after every sample for the next sample to give a valid reading. This is due to the configuration of the ADC in the IOC tool; enabling “Continuous Conversion Mode” would eliminate the need for repeatedly calling HAL_ADC_Start.

The state of the blue push-button is also read. Next are three different methods of sending messages via CAN.

The first method is a simple transmission of data to test the connection. In this mode, pushing the blue push-button on each of the Nucleo boards should light up the LED on the other board(s) if this has been configured in the callback function, as described in *CAN Callback Function* below.

The second mode sends enable and disable signals to the motor with a frequency of 0.5Hz, which should cause it to connect and disconnect the CAN connection, which should be indicated by the LEDs on the motor. However, this was found to not work during testing.

The third mode sends a control command for the motor in MIT mode using the cubemars_send_can_cmd function from cubemars_control.c. However, this was also found to not work during testing.

CAN Callback Function

The HAL_CAN_RxFifo0MsgPendingCallback function is called every time a CAN message is received. The function has been modified to contain different methods of processing the received messages, which can be commented out or uncommented to change the functionality.

The first method prints the raw numerical values of the eight received data bytes.

The second method processes the received data as a feedback message from the motor using the cubemars_get_can_msg function from cubemars_control.c and prints position, speed, torque, temperature and fault message.

The third method emulates a motor receiving a command for debugging purposes. Thus, it processes the received data as a command to a motor using the cubemars_get_can_cmd4debug function from cubemars_control.c and prints references for position, velocity, Kp, Kd and feed-forward torque. The messages were found to be processed correctly, outputting floating-point values that rounded to the same values as the ones transmitted, except for the feed-forward torque, which

returned an incorrect value. Attempts to debug this were unsuccessful, and it is unknown whether the error is on the transmitting or receiving side.

Finally, the “LD2” LED is turned on or off based on the data sent using the simple transmission mode described in *While Loop* above.

Constants and typedefs in cubemars_control.h

In addition to function prototypes, cubemars_control.h declares four special CAN commands used for entering and exiting motor control mode, setting the origin and reading the state (note that the command for reading the state is identical to the command for entering motor control mode).

This header file also declares variables for the ranges of position, velocity, torque and control parameters. The values of the constants are assigned in cubemars_control.c.

Additionally, a typedef for an enumeration (enum) defining different error codes received from the motor, which can be useful for monitoring.

Functions in cubemars_control.c

In cubemars_control.c, values are assigned to the special CAN commands and limits mentioned above. Additionally, it contains functions used for communication with sending commands to CubeMars motors and receiving feedback. Most of these functions are based on examples from the user manual, many of them with modifications. These functions are documented with comments in the file.

Conclusion

The motor can be successfully controlled using CubeMars software when connected to a PC using the R-LINK. The communication between the microcontrollers via the CAN transceivers also seems to work as expected.

However, trying to send commands to the motor did not work, which could suggest that the current CAN configuration on microcontrollers could be incompatible with the motor or that there is a discrepancy in the message format between the motor and the developed code. The test with the debugging function returned an incorrect value for one of the commanded parameters.

Recommendations for Future Work

Using the setup in *Hardware Configuration* and following the guide in *CubeMars Software* should allow for quickly getting the motor running using a PC connection.

The code described in *STM32* provides a foundation for further testing and the development of an API for controlling CubeMars AK series motors using STM32 microcontrollers, following adjustments to make the CAN communication compatible.