UNIVERSITÀ
DI TRENTO

Department of Information Engineering and Computer Science

Bachelor's Degree in
Electronics, Computer and Communications Engineering

Electronics curriculum

Final dissertation

Stepper motor testbench: expansion board and Azure RTOS driver unit based on STM32 microcontroller

Supervisor

Prof. Roberto Passerone

Student

Lisa Santarossa

Academic year 2021/2022

Acknowledgements

I feel that I should dedicate a few lines of this paper to the people who supported me in its realization.

First of all, I would like to thank my supervisor Prof. Roberto Passerone, for leading me in this thesis project and for always being ready to provide me with useful suggestions for writing, allowing me to broaden my knowledge.

Special thanks to the company tutor Dr. Maurizio Rossi, who helped me in the implementation of the project at ProM Facility company, believing and supporting me all the time with professionalism, availability, precision and patience throughout the whole period; he passed on knowledge and advises involving myself and motivated me to overcome the obstacles I encountered, and this will be precious for my professional future. I also want to thank mentor Dr. Luca Herzog, whom I met only in the first internship period, for the unique things he taught me and the time he dedicated to me.

Thanks to my sister Agnese for her closeness and support in facing my path and thanks to my parents, because without them I could never have embarked on this course of study. Thanks to all my friends who were close to me during this time and during this bachelor's path, especially Tommaso for enthusiastically sharing the internship path and the thesis project and Simone for his thoughtful advice and his brilliant ideas. Along with them, my grandmother Rina, for her closeness no matter the distance, especially for her encouraging phone calls.

Trento, August 31, 2022 *Lisa Santarossa*

Contents

Abstract

I attended my academic internship at ProM Facility, a company specialised in creating ad-hoc mechatronics applications, and here I decided to take part in solving one of their needs. Since several of their applications involve stepper motors, ProM needed to have a stepper testbench to test these devices: in this way, it would be possible to precisely know their behaviour in order to achieve a better final product quality.

This thesis discusses the design and development of a **stepper motor testbench prototype** subsection, which includes an **expansion board** to acquire sensors signals and an ***Azure RTOS* driver unit** based on an STM32 microcontroller to physically drive the motor. The system ideated was built from scratch following ProM's design requirements.

The expansion board represents the hardware part of the project. It is a Printed Circuit Board (PCB) designed with Kicad, an electronic open-source CAD. After being realised, the board was analysed with a thermographic camera to verify the power-on components' condition state, with an oscilloscope to test the acquired signals' noise, and with an electromagnetic scanner to measure the electromagnetic compatibility (EMC). On the other hand, the driver unit represents the firmware development of the stepper driver based on *STM-Nucleo F4 series board*, it is implemented and debugged with the environment offered by STMicroelectronics called *STM32-CubeIDE*.

The expansion board and the driver unit firmware work properly: they satisfy ProM's requirements and lay the foundations for future developments of the prototype.

1 Introduction

1.1 Problem Statement

During my last year of studies, I completed my traineeship in the *Research and Development area* at ***ProM Facility***, a company based in Rovereto designed to be a f*ast-prototyping centre* for mechatronics embedded systems solutions. In the projects in which ProM is involved, it often finds itself using actuators such as **stepper motors**, i.e. devices largely used both in consumer and industrial applications, which require a high level of precision. They are electric motors that, thanks to their internal structure, allow the rotor to be moved in pulses or steps; the discrete angular amplitude is fixed a priori. Furthermore, they are able to deliver very high torque at a low number of revolutions, being able to overcome any opposing forces applied to the axis, thus maintaining possible stationary loads in their position without vibrations or large oscillations. Another advantage of their conformation is that it allows to obtain a compact device with reduced dimensions compared to other electric motors with the same nominal power [4] [1].

Since the number of applications involving this kind of motor is uncountable, the producer only provides data concerning nominal and/or canonical work conditions. Therefore, when it is necessary to design ad-hoc reliable and efficient systems, it is important to understand the behaviour of parameters such as torque and power output over time.

One of ProM's needs is to obtain a **testbench for stepper motors**, able to test and **characterise torque and power output over time**, given the fact that they are given by the producers with a small quantity of data regarding their real performances.

1.2 General overview and thesis goal

To face ProM's needs, we have designed and developed a prototype device from scratch with the following requirements:

- *STM32F7 master* microcontroller with dedicated firmware (*main board*)
- signal acquisition board, designed ad-hoc
- *motor driver unit* on *STM32F4* board equipped with Microsoft's *Azure RTOS* (*slave board*)
- Graphic User Interface (GUI) software, developed using *Python* toolkit *dearpygui*

The project goal is hence to validate ProM's idea starting from the proposed requirements, creating a **prototype device for internal use**, then validate the components' functionalities and the system modularity; to achieve such a goal we ended up developing two parallel projects. This thesis project's goal is to realise the **expansion PCB board** and develop the **firmware for the motor driver** based on the microcontroller STM32, by the F4 family.

More in detail:

1. design schematics and layout of a PCB board to handle the testbench sensors, made to be allocated on the main board and to use the available components in the warehouse
2. mount and assemble the prototype, checking the effective functioning by testing
3. develop the firmware for the stepper motor driver on a *Nucleo-STM32F4* board and a *powerSTEP01* shield and an UART interface with main board and GUI.

1.3 Thesis outline

The thesis follows this structure:

- **Chapter 2** explores presentation and technical description of the testbench to design following ProM's requests
- **Chapter 3** illustrates the hardware task developed
- **Chapter 4** discusses the hardware tests and their results
- **Chapter 5** shows the firmware structure and how it works
- **Chapter 6** explores closing thoughts.

2 Testbench

2.1 Overview main structure

The testbench has three main elements which are mounted on the same direct axle (Figure 2.1):

1. the stepper motor (DUT)
2. a calibrated torque sensor
3. an electromagnetic brake

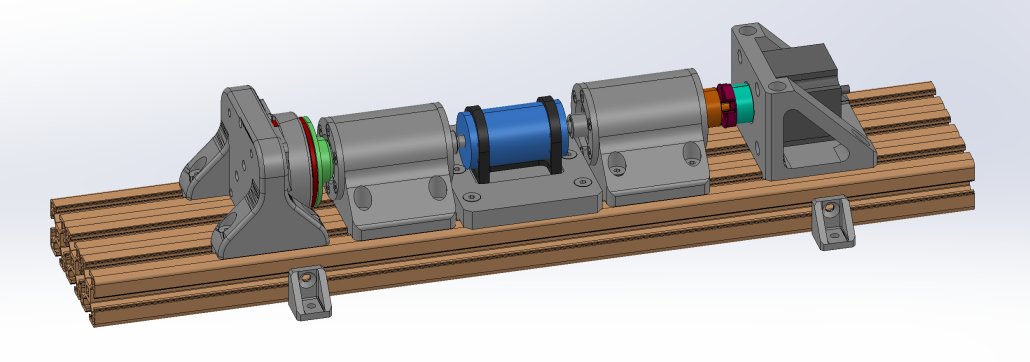


Figure 2.1: Testbench mechanical structure with brake, torque sensor and stepper

The mechanical structure supporting these three objects has been entirely designed and 3D-printed at ProM using a nylon-powder printer.

2.1.1 Stepper motor

This first prototype has been developed in order to test *Nema 23 series* stepper motors, which are very popular and have the same physical dimensions: in fact, the structure has a fixed faceplate slot of 57x57mm that is their common factor (Figure 2.3).

The one currently mounted is a bipolar stepper motor model *Nema 57SH51-4B* and its main characteristics are [7]:

- 1.8° minimum step degree
- 1Nm maximum holding torque (Figure 2.2)
- 2.8A maximum phase current
- 30V nominal voltage

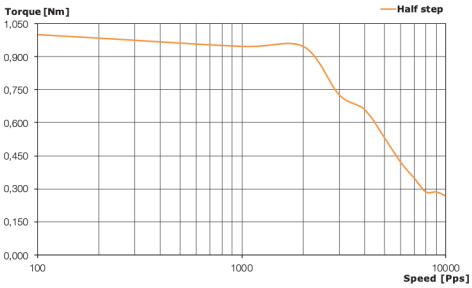
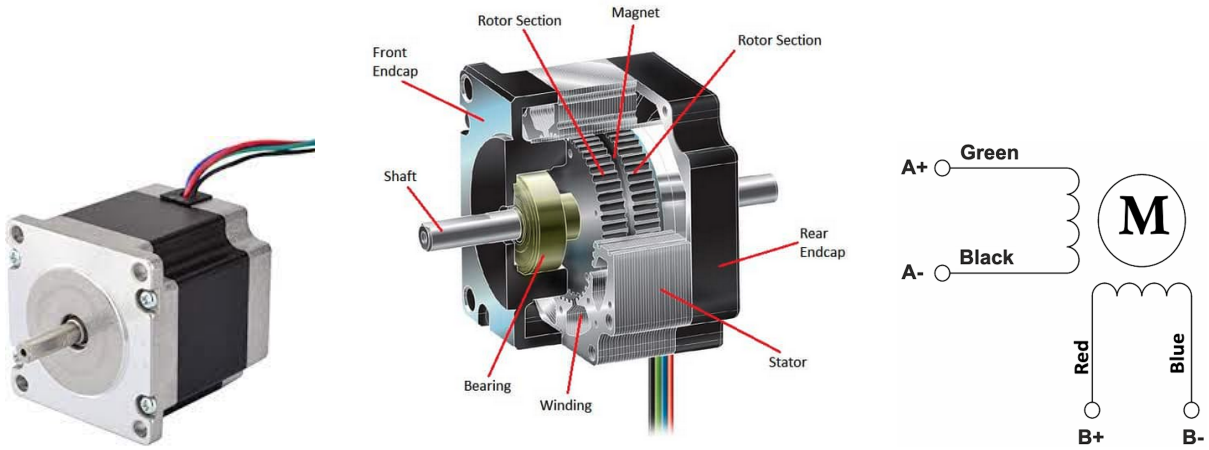


Figure 2.2: Torque vs Speed at 30V, 2.8A



(a) (b) (c)

Figure 2.3: Nema 23 Bipolar Stepper Motor, its construction and its wiring diagram

The stepper is the Device Under Test and it is supposed to be mounted and then tested with running parameters selected by the user.

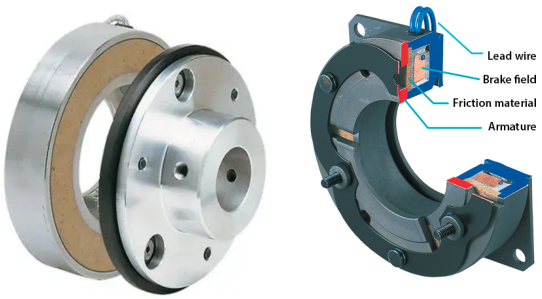
2.1.2 Torque sensor

The torque sensor measures the torque developed at the shaft both dynamically and statically during the motor desired performance. In our case, the already available *NCTE Series 2200-2.5Nm* torque transducer is used (Figure 2.4). It works at 12V and has a nominal bidirectional torque of 2.5Nm, it also gives a 0 to 5V output proportional to the torque detected and it has already been calibrated by the certified manufacturer in such a way that the slope coefficient of its measurement is 781.43mV/Nm [9].

Figure 2.4: NCTE 2200 torque sensor

2.1.3 Brake

An electromagnetic brake is a reversible electrical machine that develops an holding torque when some voltage is applied [10]. In this project the brake is controlled with a variable input voltage in order to obtain different intensity of braking force. The brake used is a *MWN-emsl-08.25.060.02* (Figure 2.5), with a variable supply voltage ranging from 10 to 24V. This specific device is an electromagnetic single-disk brake with flat springs which can apply up to 5Nm torque [8].



(a) (b)

Figure 2.5: Emsl series brake and a generic brake construction

2.2 Hardware and Firmware design

The project needs dedicated solutions for hardware, firmware and software design that are illustrated in Figure 2.6 and schematized in Figure 2.7.

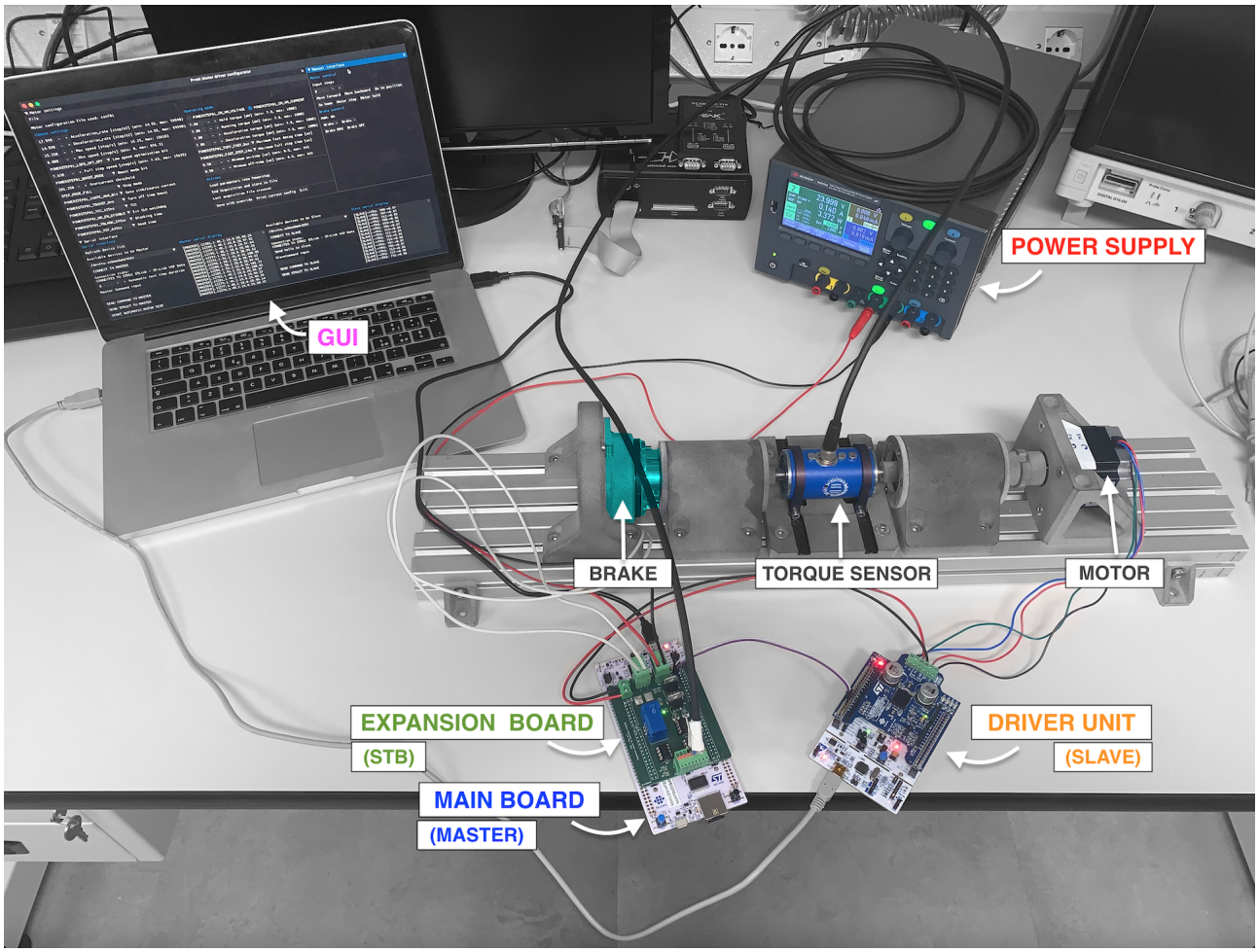


Figure 2.6: Testbench setup

The hardware design requires a PCB (Expansion Board or STB), which has to be physically mountable on the main board for signals acquisition. The parameters to detect are:

- **electrical power consumption** of the DC power supply of driver+motor section, therefore is required a current sensor in series with the power supply line and a voltage sensing circuit;
- **mechanical torque produced** by the motor, i.e. an analog signal made available by the torque meter;

The PCB also requires:

- a circuitry section to properly pilot the electromagnetic brake with a variable breaking effect;
- another circuitry section to regulate the voltage at the needed value, taking in input 24V from an external power supply;
- connectors to connect the external power supply and the peripherals;

On the other hand, the requirements for what concerns firmware and software design are:

- development on a Nucleo STM32H7 board (main board) of:
  - i analog signals acquisition and value conversion from digital to real;

ii set an UART channel to communicate with the GUI to transmit acquired data that needs to be processed;

iii set another UART channel to communicate with the slave and perform start test of driver+motor system;

- development using Azure RTOS on Nucleo STM32F4 board of:

  i an interface with PowerSTEP01 shield;

  ii set an UART channel to communicate with the GUI, in order to receive user values to run the stepper and to get its configuration parameters (such as velocity, acceleration, start mode, etc.)

  iii set a second UART channel to communicate with the main board and to receive the data need to do a test start without GUI information

- development of a Python GUI capable in the first place of initializing the serial communication and take as user input the input configuration parameters to run the stepper and its running commands; then, it has to display the characterization diagrams.

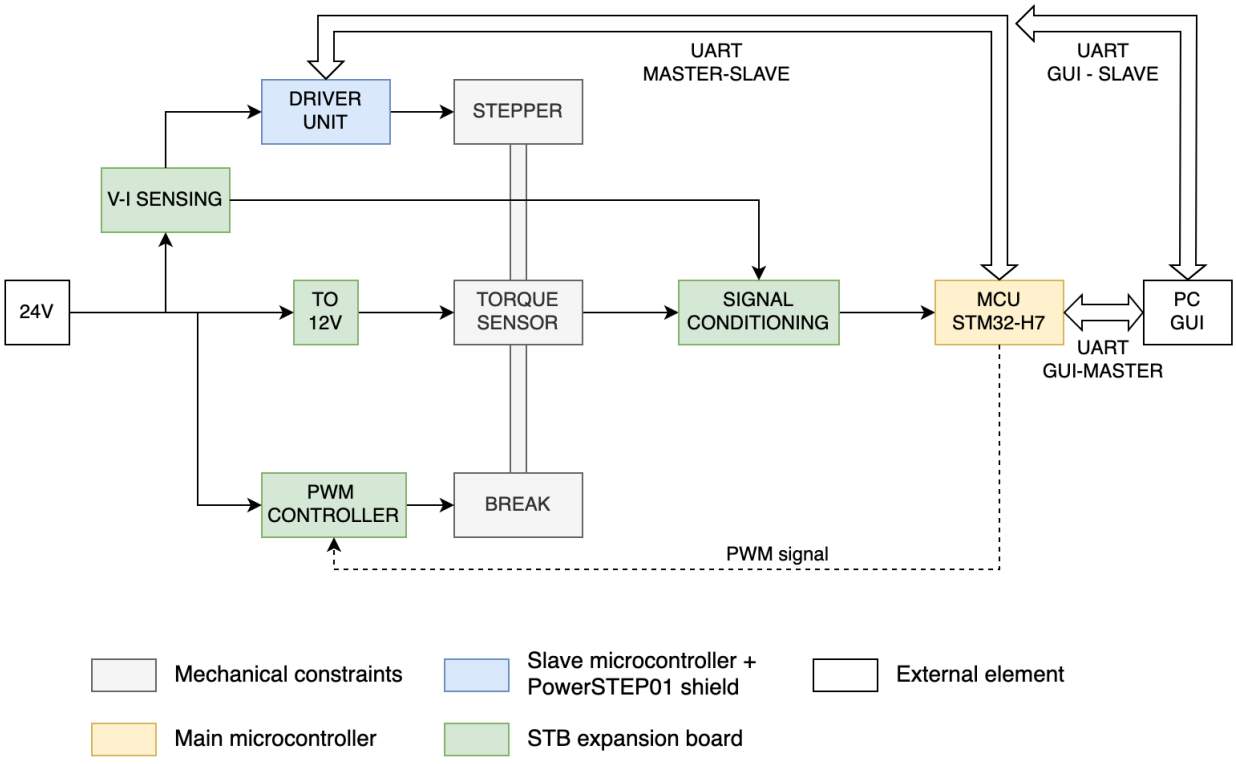


Figure 2.7: System setup blocks schema

3 Expansion Board

3.1 CAD Software

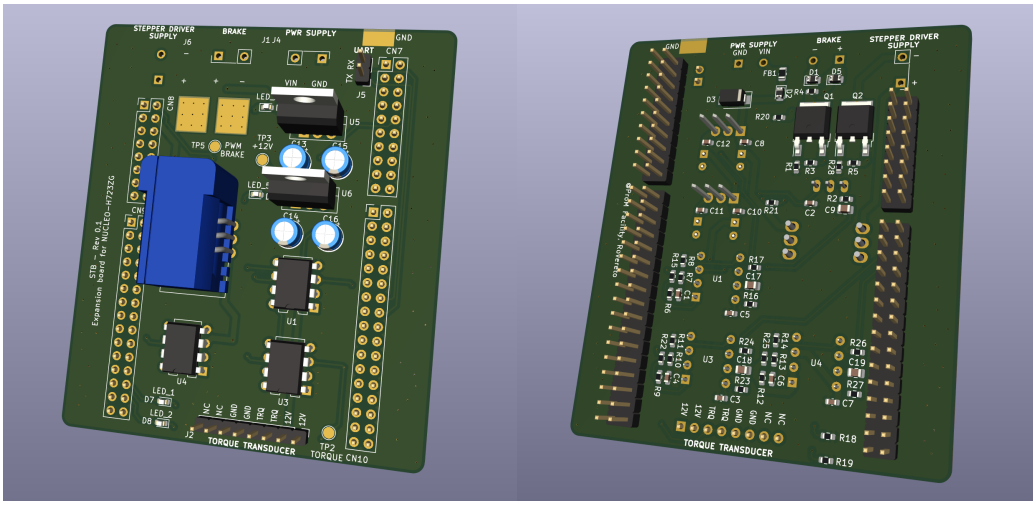
The hardware is designed with the open source software KiCad v.6.0. To be more specific, it is a suite that includes dedicated software, one for every design step; the main are:

- Kicad, to handle the entire project files
- Schematic Editor, to handle the schematics design
- PCB Editor, to handle the layout design i.e. the components placement and the routing of the tracks
- Gerber Viewer, to handle Gerber and Drill files

By installing an additional plug-in it is also possible to add post-design features such as teardrops[1].

3.2 Description

The designed shield, shown below (Figure 3.1), is called Stepper Testbench Board, shortened STB:



(a) (b)

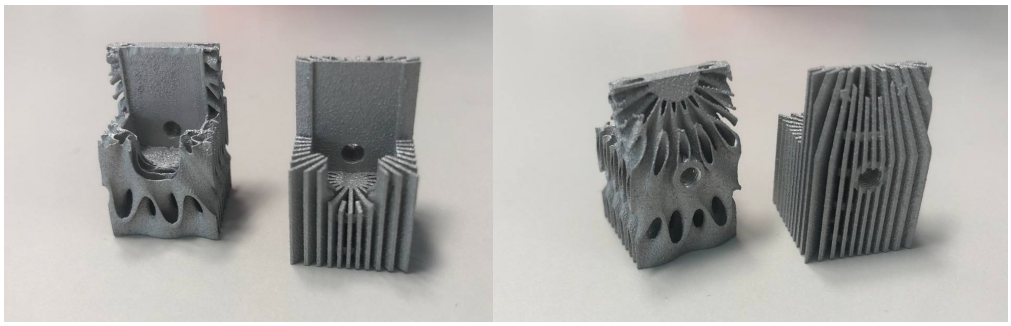
Figure 3.1: STB Expansion Board top and bottom 3D-render

An STB is an expansion board connectable to the top of the main board through its female header connector placed at the bottom. It allows the main board to acquire filtered and conditioned measured signals of current, voltage, and torque and has two circuit sections to drive the stepper driver unit power supply and the electromagnetic brake. On the top are placed the connectors needed to connect the stepper driver unit supply, the main external power supply, and the EM brake.

The STB has a power management section that receives input of 24V from an external supply and converts it to 12V and 5V with a cascade of two linear regulators. Switching ones would be better

[1] **Teardrop**: connections between pad and track to avoid sharp edges (stress concentration points), reducing the risk of breakage of the connections and of the thermal stress on the joints due to welding.

in terms of efficiency, but it could cause noise and EM interference and this has more priority than power consumption. In order to properly dissipate the heat produced by the linear regulators, it is necessary to use heatsinks, otherwise, the increasing temperature could damage the components or make them work incorrectly. Since fitting commercial heatsinks were not available in our laboratory, at ProM we could 3D-print two custom efficient heatsinks (Fig. 3.2).



(a) (b)

Figure 3.2: Custom 3D-printed heatsinks

Other features available on top of the shield are two state LEDs, 24V and 5V LEDs alive, a two-pin connector for the UART channel, and some useful test points.

Note that the design choices have been influenced by the need of ProM to use the already available components because nowadays the purchase of electronic material is very difficult due to the market volatility. However the current sensor has been bought in detail, it is a LEM CAS-6NP AC/DC current transducer with $I_N = 6A_{RMS}$ and it can measure the current from -20A to 20A and maps the value from 0.375V to 4.625V [5]. For the actual version of this board, only a DC transducer would be fine too but in nearly future works the testbench could be extended also to AC motors.

The current transducer is placed in series with the driver unit power supply connector to sense the current sunk by the driver+motor section. In parallel there is a voltage divider designed to sense the actual voltage across this connector to let the main board calculate the total power consumption. Current, voltage, and torque signals are low pass filtered and then conditioned with a suitable resistor voltage divider; these two stages are properly separated with operational amplifiers before and after.

The STB blocks schema is shown in Figure 3.3 and to see the schematic, layout and BOM refer to Attachments A.1, A.2 and A.3.

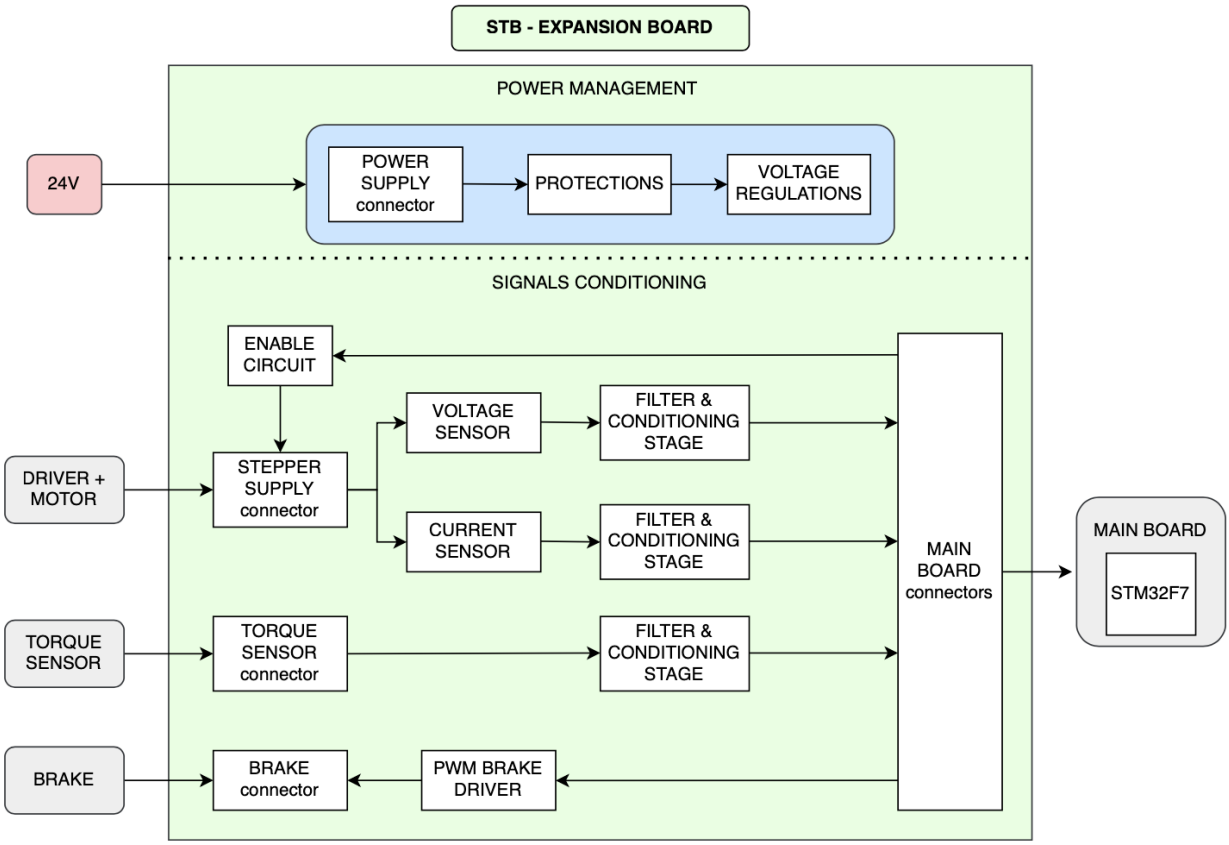


Figure 3.3: STB blocks schema

4 Hardware Analysis

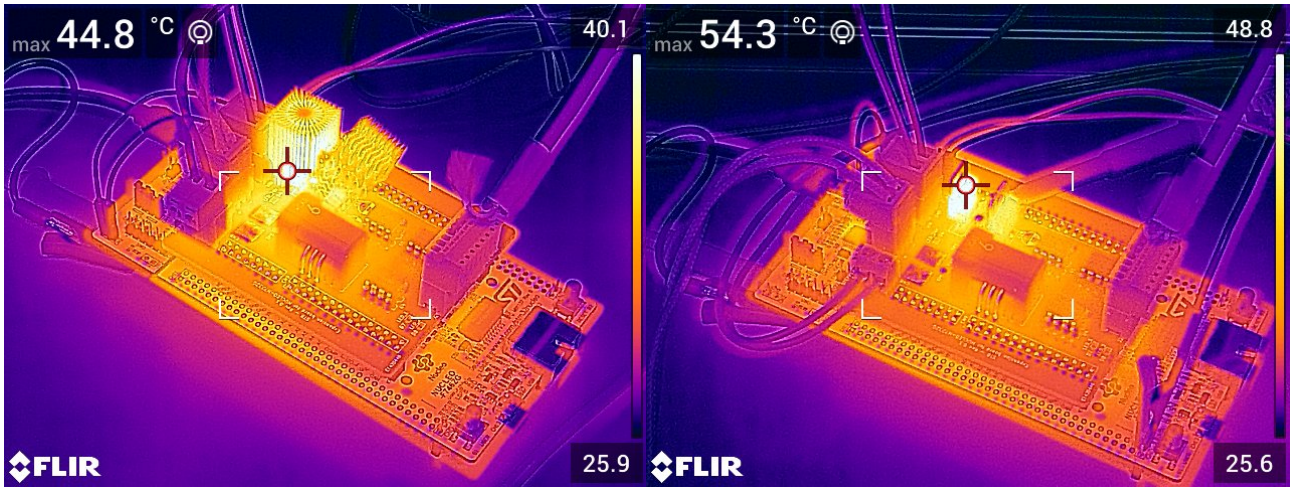
4.1 Thermographic and FFT analysis

The STB has been tested with a thermographic camera *FLIR* in two separate scenarios, to detect the temperature of the linear regulators with or without heatsinks. Their behaviour were observed during the increase of heat dissipated.

A. scenario A is a generic setup with which we normally test the stepper, i.e. when the mainboard is powered via USB and on its top is mounted the STB;

B. scenario B is a simulation of a future work, in which the STB is powered stand-alone with a load of 20Ω applied downstream of the 5V regulator, to simulate the scenario in which the mainboard is powered by STB and not externally by USB.

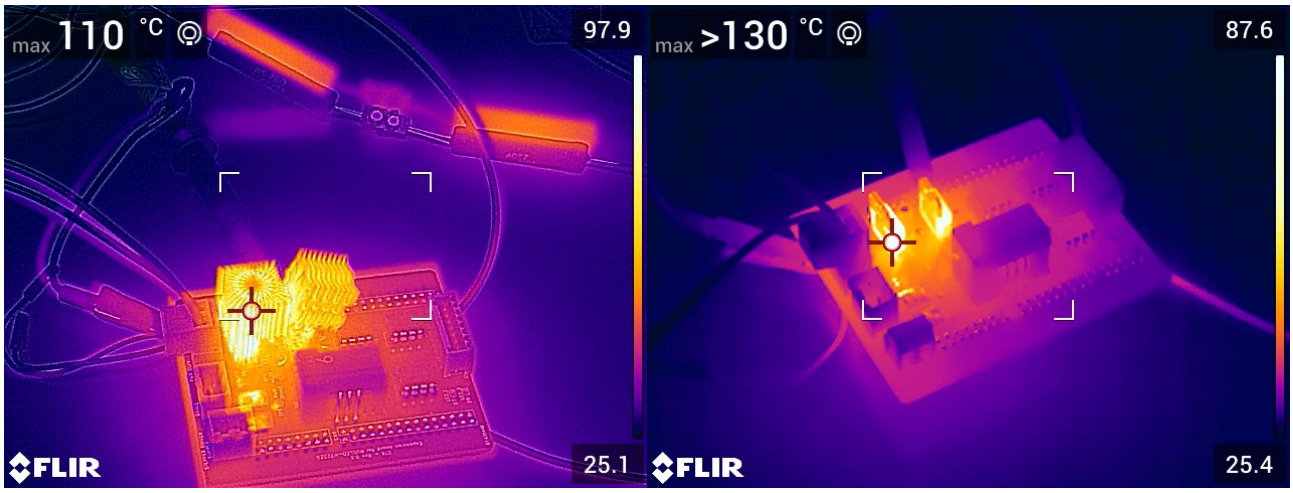
Scenario A: In this setup the actual STB consumption is about 250mA and, with or without heatsinks, the temperature during a powerd-on and 30-minutes test period is limited under 60°C and the functioning of the regulators is not compromised; hence, they correctly regulate on 5V (Figure 4.1).



(a) (b)

Figure 4.1: STB with heatsinks at 45°C and without heatsinks at 54°C

Scenario B: In this setup the total consumption is about 500mA. In fact, the load applied consumes up to $I_L = \frac{V}{R_L} = \frac{5V}{20\Omega} = 250mA$ like the main board request. Here the use of heatsinks becomes fundamental in order to obtain an adequate 5V regulation: using heatsinks the temperature stabilizes at 110°C, which is a little bit high value but works properly; on the other hand, without heatsinks the temperature increases continuously exceeding the 130°C threshold (maximum temperature detectable by the FLIR camera). A few moments later the 130°C overcoming, the components do not regulate correctly (Figure 4.2).



(a) (b)

Figure 4.2: STB with heatsinks at 110°C and without heatsinks at over 130°C

If without heatsinks, when the 12V regulator exceeds the 130°C, it enters the overtemperature protection mode to preserve itself. Using an oscilloscope has been possible to show its behaviour, see Figure 4.3 .

Figure 4.3: 5V regulation not working

In addition, a Fast Fourier Transform (FFT) of the 5V signal has been computed and displayed in white in Figure 4.4; as we can see there, the signal is not perfectly DC (0HZ) but presents noisy frequency at almost 5KHz.

Figure 4.4: FFT (white) of the improper 5V signal (green)

4.2 Noise filtering

The STB aims to condition analog signals in order to let the main board acquire them properly. The filtering and conditioning stage performance were hence tested with an oscilloscope. In Figure 4.5 it is shown in yellow the direct voltage output signal of the current sensor and in green the filtered and conditioned signal which is acquired by the main board's ADC. Here we can see that the yellow signal is clearly noisy while the green one does not present evident ripples and at the same time has a stable trend. Comparing the two peak-peak measurement, there is a 30mV overall reduction of noise overlapping and, considering the current sensor sensitivity of 104.2mV/A, 30mV corresponds at 288mA less variation on the final acquired signal. If needed, the STB allows to place an additional active low-pass filter to filter again the ADC signal.

Figure 4.5: Raw (yellow) and filtered and conditioned current signal (green) and peak-peak measurement

4.3 EMC and EMI tests

At ProM Laboratory we could test electromagnetic radiated emission using *EMxpert EHX Near Field Scanner*. This device is connected to a *spectral analyzer* and it is a real-time EMC and EMI diagnostic tool for very-near-field tests (Figure 4.6).

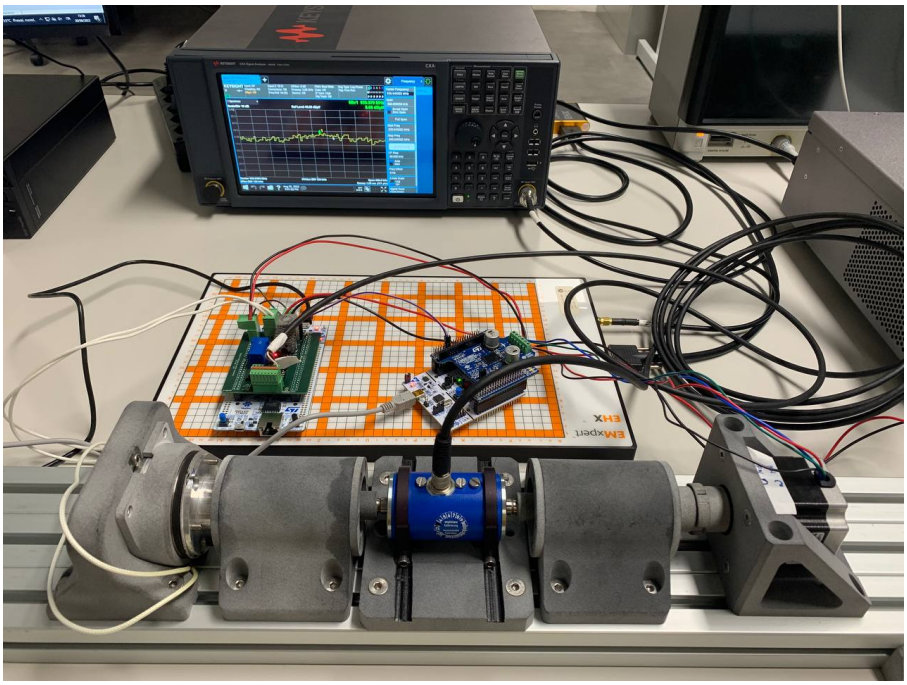
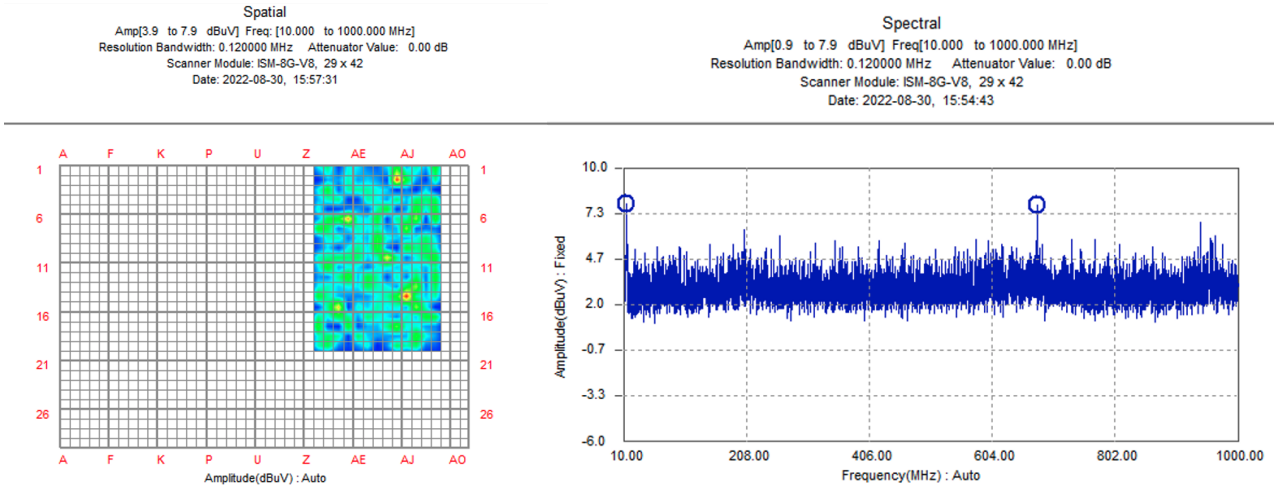


Figure 4.6: EM radiated emission test setup with EMxpert Scanner and Spectral Analyzer

Three different tests were done:

1. With motor in stand-by mode (Figure 4.7)



(a) (b)

Figure 4.7: Stand-by test

2. With motor running continuously (Figure 4.8)

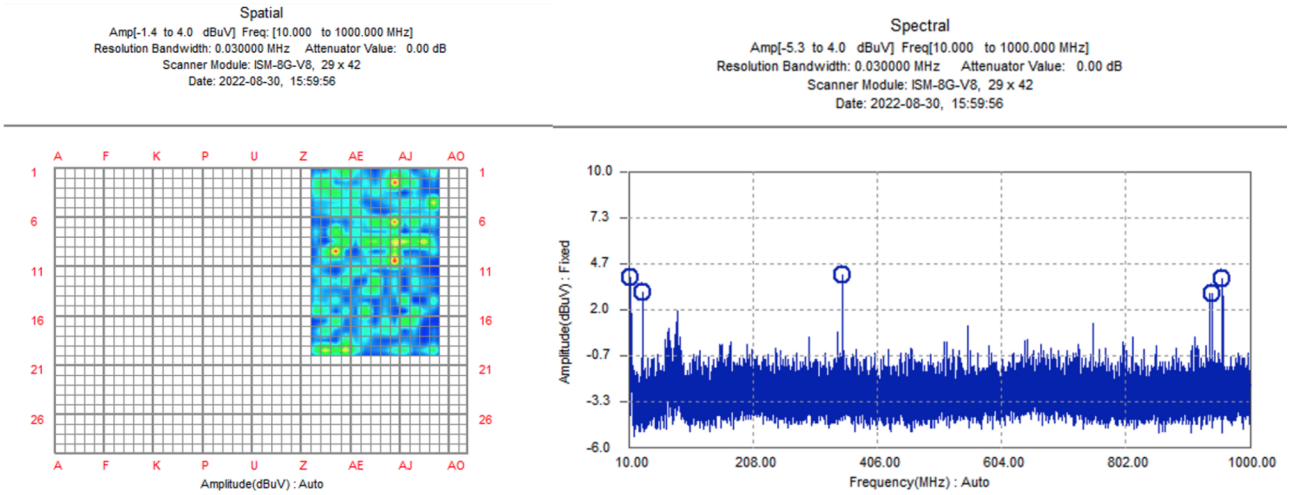


Figure 4.8: Motor running test

3. With motor running continuously with the shield laying on heatsinks side as in Figure 4.9 (Figure 4.10)

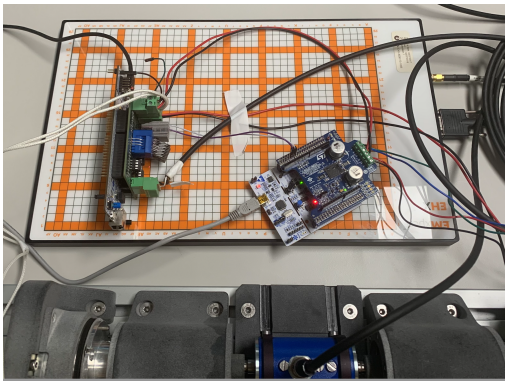


Figure 4.9: Setup for lateral test

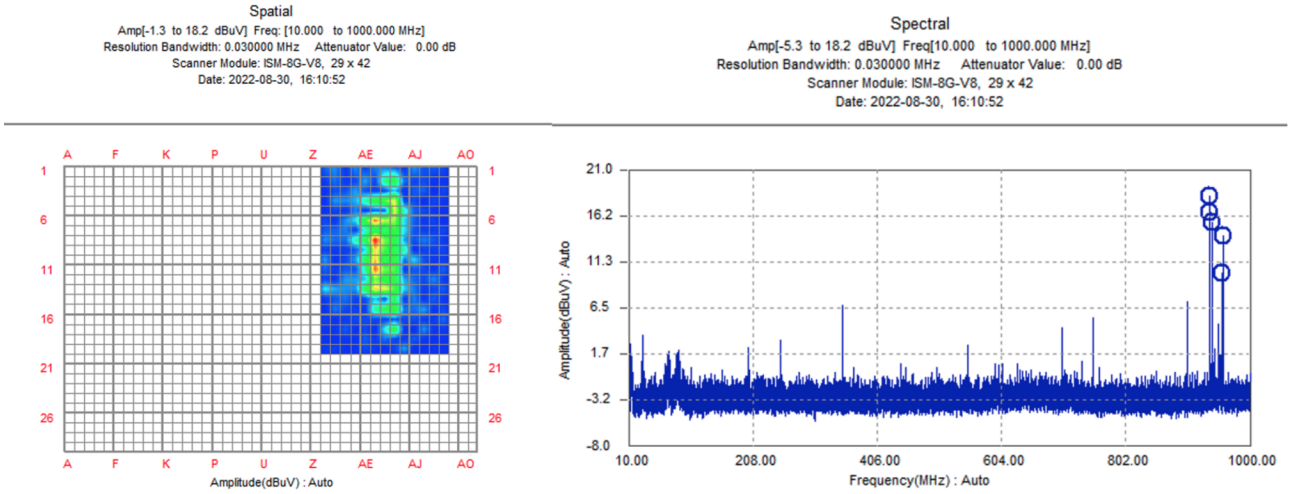


Figure 4.10: Motor running test from heatsinks side

From the charts above, it is possible to ensure that the STB and the main board do not generate any potential interference: in fact, in a range from 10KHz to 1GHz, the frequency amplitude does not reach the -40dBuV along with the *EC 61000* and *CISPR 16* regulations. To sum up, this setup can be recreated everywhere without causing any problematic EM radiated emissions.

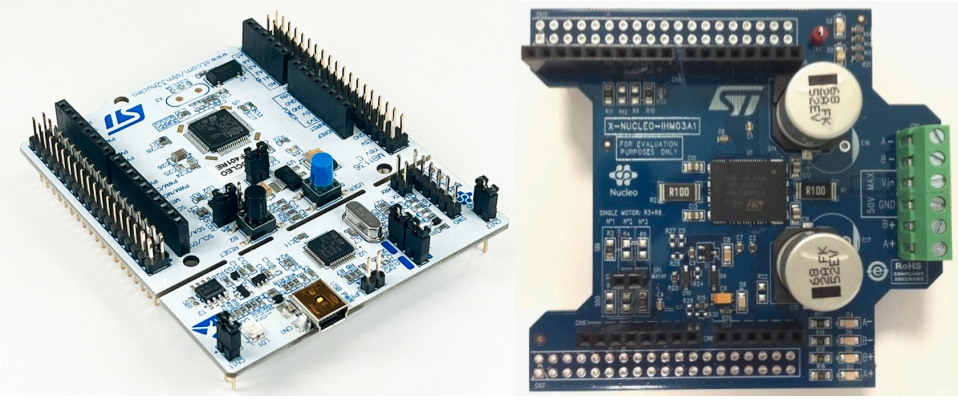
5 Stepper motor Driver Unit

5.1 Software tools

The firmware was developed using *STM32-CubeMX* and *STM32-CubeIDE*. The first one is a graphical tool that allows GPIO and peripherals configuration of STM32 microcontrollers and generates the corresponding initialization C code; the second one is a development environment platform that includes the first one and implements code compilation and feature debugging.

5.2 Hardware

The Driver Unit consists of a Nucleo STM32F401RE board with a PowerSTEP01 shield model *X-Nucleo-IHM03A1* on its top (Figure 5.1).



(a) (b)

Figure 5.1: Nucleo STM32F401RE board and PowerSTEP01 shield

The Nucleo board based on an *ARM Cortex M4* STM32F401RE microcontroller works up to 84 MHz and has all the principal peripherals such as ADC, Direct memory access (DMA) and timers. It is affordable and convenient to use since we can develop on it through an embedded *st-link* [1].

On the other hand, the PowerSTEP01 shield is a bipolar stepper motor driver expansion board that allows full digital motion control. It can deal with up to $10A_{RMS}$ of phase current and is already equipped with overcurrent and overtemperature protection. This board allows to drive the stepper in two separate ways, current mode and voltage mode, and for each one the user has to set different driving parameters.

5.3 Firmware and RTOS

The firmware is not *bare-metal*[2] but it is based on Real Time Operating System to have non-blocking code to autonomously handle the motor piloting and the transmission/receiving of messages via UART.

In general, a **Real-Time Operating System** is system software that manages processor resources for applications and its main purpose is to allocate processing time among various duties the embedded software must perform. This typically involves a division of the software in *tasks* or *threads*,

[1] **St-link**: in-circuit debugger and programmer for the STM microcontrollers
[2] **Bare-metal**: system without a base operating system or installed applications.

and creating a run-time environment that provides each thread with its virtual microprocessor, i.e. (*multithreading*). The Real-Time Operating System controls thread execution, and the accompanying management of each thread's context, for example, each thread is given a priority by the designer, to control which thread should run if more than one is ready to run (i.e.: not blocked) and when a higher-priority thread (compared to the running thread) needs to execute, the RTOS saves the currently running thread's context to memory and restores the context of the new thread. The process of swapping the context of threads is commonly called *context switching*. This transfer of control to another thread is immediate and invisible to the embedded software and this invisibility might be the most fundamental benefit of an RTOS, in fact instead of embedding processor allocation logic inside the application software, it is done by the RTOS. This arrangement isolates the processor allocation logic and makes it much easier to predict and adjust the run-time behavior of the embedded device. It's important to note that an RTOS must offer *preemption*, the action of switching to a higher-priority thread instantly and transparently, without having to wait for the completion of the lower-priority thread [6].

For this application, Microsoft's *Azure RTOS* was chosen, a recent RTOS supported by STM products. A kernel like *FreeRTOS* would have been a better-known alternative since it is more used and supported by 35 microcontroller platforms; nevertheless, we opted for the former to deal with a new challenge and provide the know-how to the company.

5.3.1 RTOS Tasks

The firmware is made of three main tasks:

1. *Powerstep*, manages the driving stepper functions received from the serial tasks that wrap functions of the *powerstep01* library offered by STMicroelectronics. Through this task, commands are then sent via Serial Peripheral Interface (SPI) to the PowerSTEP01 shield which physically drives the motor as desired. The task diagram is shown in Figure 5.2.

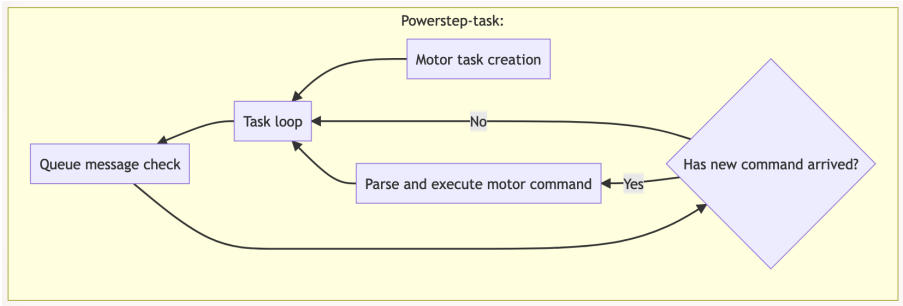


Figure 5.2: Powerstep task diagram

2. *Serial-to-GUI*, configures via the serial channel the Driver Unit when the GUI sends motor configuration parameters and driving commands. The task diagram is shown in Figure 5.3.

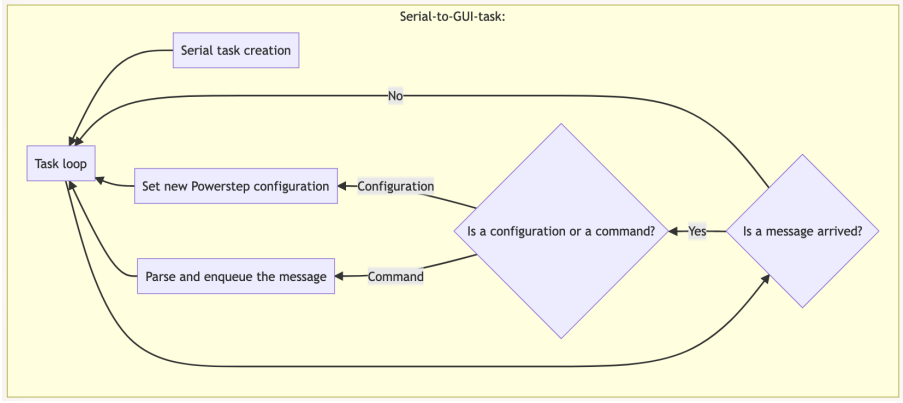


Figure 5.3: Serial-to-GUI task diagram

3. *Serial-to-master*, manages the serial communication between the Driver Unit and main board, where through a dedicated UART peripheral (distinct from the one of the previous task) the driving motor commands are sent for a test start. The task diagram is shown in Figure 5.4.

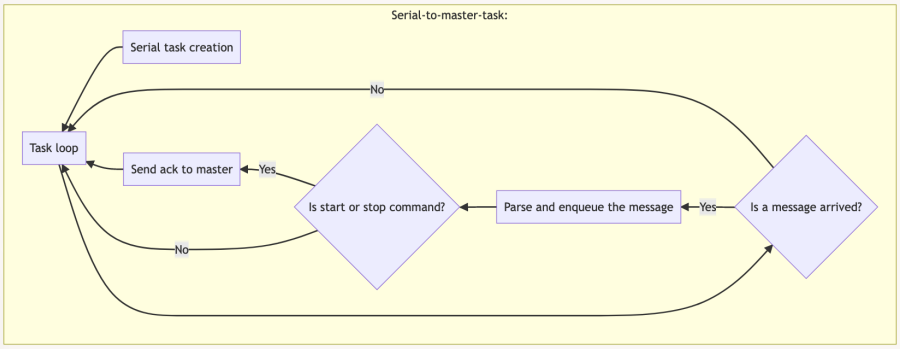


Figure 5.4: Serial-to-master task diagram

In general, a *queue* structure is used to let the Powerstep task receive all the commands coming from the serial tasks while it is piloting the motor.

Each of the above tasks plays a precise role and at the beginning, they need a creation phase when the parameters [3] in Table 5.1 have to be set via firmware:

Parameter	**Type**
Reference to the thread	TX_THREAD*
Custom thread name	char*
Entry function	VOID*
Entry input	uint64_t
Stack pointer	VOID*
Stack size	uint64_t
Thread priority	uint32_t
Thread preemption threshold	uint32_t
Thread time slice	uint64_t
Thread auto-start	uint32_t

Table 5.1: Required parameter to create a task

```
/* Queue creation */
tx_queue_create(&ptr_motor_cmd_queue, "motor command queue", 2, stack_motor_cmd_queue,
QUEUE_STACK_SIZE);
/* Tasks creation */
tx_thread_create(&serial_to_GUI_thread_ptr, "serial to gui", task_Serial_to_GUI, 33,
serial_to_GUI_stack, THREAD_STACK_SIZE, 12, 12, 100, TX_AUTO_START);
tx_thread_create(&serial_to_master_thread_ptr, "serial to master", task_Serial_to_master,
33, serial_to_master_stack, THREAD_STACK_SIZE, 12, 12, 100, TX_AUTO_START);
tx_thread_create(&powerstep_thread_ptr, "powerstep", task_Powerstep01, 30, powerstep_stack,
THREAD_STACK_SIZE, 9, 9, TX_NO_TIME_SLICE, TX_AUTO_START);
```

Listing 1: Queue and Tasks creation

The code regarding queues and task creation can be found in Listing 1. Here we can observe that the Powerstep task priority is 9, while the one of the two serial tasks is 12; taking into account that a lower value corresponds to a higher priority, it is easy to notice that Powerstep task has the highest

[3] https://docs.microsoft.com/en-us/azure/rtos/threadx/chapter4

one because the main goal is to drive the DUT. On the other hand, the Serial tasks have the same priority because they do the same function but with different peripherals, so there is no reason to give them a different one.

In addition, an important parameter is the *time slice*, which is the number of timer-ticks that the thread is allowed to run before other ready threads with the same priority are given a chance to run. This parameter needs to be set to a value different from 0 (*TX_NO_TIME_SLICE*) only when two or more tasks have the same priority. In this case, preemption is negligible because it works only for tasks with different priorities. Given that the serial tasks have the same priority, the definition of the time slice parameter was needed to run the task concurrently. Indeed, if the time slice is not specified the only task allowed to run is the first created, because the *auto-start* value in the creation phase is set to *TX_AUTO_START*.

5.3.2 Command parsing

As mentioned earlier, messages from GUI or main board are sent through the dedicated serial channels and get elaborated by the Driver Unit firmware to pilot the stepper motor. These messages are coded strings that may contain **motor commands** or **motor configuration parameters**. If the test is set through the GUI by the user, both motor commands and motor configuration parameters are sent, otherwise, the developer can run a test through the main board star only sending motor commands. The two serial tasks have common functions, therefore a specific library was planned to include shared functions and structs in order to reduce code redundancy.

Motor commands consist in two fields: *command code* and *value*. They are defined in a dedicated *struct* (Figure 5.5): the first two chars are the code of the preset commands (Table 5.2), than there is a delimiter character '+' followed by a numeric meaningful value (e.g. *fw+500000*). To convert the received string into a real command, the firmware parses the message and then builds it up, thus *fw+50000* is translated and sent to the Powerstep task responsible to "run forward the motor by 5000000 steps". Motor configuration parameters are sent from GUI as coded strings too, similarly to the motor commands, and they are decoded and then set into the Powerstep task. In the serial-to-GUI task, motor commands are discriminated from the motor configurations parameters using the different positions of the termination char of the strings.

```
typedef struct
{
  motor_cmd_code    cmd;
  uint32_t          val;
} motor_cmd_msg;
```

Figure 5.5: Motor command message struct

Command	Code	Description
MOTOR_MOVE_FW	fw	run forward
MOTOR_MOVE_BW	bw	run backward
MOTOR_HOLD	hd	hold current position
MOTOR_STOP	st	stop
MOTOR_GET_POS	ps	get current position
MOTOR_GO_TO	gt	go to set position
MOTOR_GO_HOME	gh	go to home state position

Table 5.2: Motor commands

5.3.3 Motor configuration parameters

The stepper configuration parameters are distinct whether the driving mode is set to current mode or voltage mode. These two modes have both a common section and specific payloads (Table 5.3, 5.4 and 5.5)

Parameters	**Type**
Acceleration rate	float $[step/s^2]$
Deceleration rate	float $[step/s^2]$
Maximum speed	float $[step/s]$
Minimum speed	float $[step/s]$
Low speed optimization	bit
Full step speed	float $[step/s]$
Boost of the amplitude square wave	enum
Overcurrent threshold settings	enum
Step mode settings	enum
Synch. Mode settings	enum

Table 5.3: Common parameters of the motor configuration payload

Parameters	**Type**
Holding torque	float [mV]
Running torque	float [mV]
Acceleration torque	float [mV]
Deceleration torque	float [mV]
Maximum fast decay time	enum
Maximum fall step time	enum
Minimum on-time	float [us]
Minimum off-time	float [us]

Table 5.4: Current mode only parameters payload

Parameters	**Type**
Hold duty cycle (torque)	float [%]
Run duty cycle (torque)	float [%]
Acceleration duty cycle (torque)	float [%]
Deceleration duty cycle (torque)	float [%]
Intersect speed settings for BEMF compensation	float $[step/s]$
BEMF start slope settings for BEMF compensation	float $[step/s\%]$
BEMF final acc slope settings for BEMF compensation	float $[step/s\%]$
BEMF final dec slope settings for BEMF compensation	float $[step/s\%]$
Thermal compensation	float
Stall threshold settings	float [mV]

Table 5.5: Voltage mode only parameters payload

5.4 Memory usage

The Driver Unit firmware occupies about 18% of the RAM available and less than 10% of FLASH, as shown in Figure 5.6. Althoug an RTOS is used, which in general requires more memory space than bare-metal firmware, there is plenty of space left to implement other future features.

Memory Regions | Memory Details

Region	Start address	End address	Size	Free	Used	Usage (%)
RAM	0x20000000	0x20018000	96 KB	78.12 KB	17.88 KB	18.63%
FLASH	0x08000000	0x08080000	512 KB	472.65 KB	39.35 KB	7.69%

Figure 5.6: RAM and FLASH usage

6 Conclusions and Future work

At the current state of the project, the thesis goal was successfully achieved. In fact, following ProM's requirements we created a working and scalable system to realise a reliable testbench prototype from scratch. This testbench allows to measure and plot charts regarding torque and power output over the time of the DUT, which is a Nema 23 stepper motor.

This thesis project regards the realisation of a hardware expansion PCB prototype to acquire the signals needed and the development of the firmware dedicated to physically drive the motor. For what concerns the hardware part, the expansion board, also referred to as STB, works correctly and properly fits on the top of the main board. It consumes about 250mA to filter and condition signals and at the same time it does not emit EM interference; however, the filtering and conditioning stage gain needs to be fixed by the main board's firmware due to the tolerance of the real components. Moreover, we could size and solder the active low pass filter on the board, and observe its effect on signals acquired by the main board's ADC channels. The further reduction of the noise could allow to carry out faster acquisitions since the firmware filtering would be reduced or even canceled. The realised Driver Unit firmware correctly works too; in fact, it manages to receive running and configuration commands and translates them into physical stepper driver signals. To achieve this, an RTOS was necessary and Azure RTOS turned out a good solution that does not occupy too much memory space: in this way, the firmware of the Driver Unit becomes improvable and highly scalable.

In nearly-future works, the STB revisions should provide the power supply for the main board, so it will no more be powered on via USB cable and, to enhance efficiency, the onboard power management should be changed to switching regulators, also trying to properly limit the EMI to keep the device electromagnetically compatible. Moreover, for a strong serial data transmission UART peripherals are not the most reliable, so interface protocols such as *Controller Area Network (CAN)* or *RS-485* or *RS-232* should be implemented. In this way, the STB would include a suitable transceiver and the Driver Unit firmware would be updated too. For what concerns the firmware, we could develop another task to keep monitoring the PowerSTEP01 shield state and then self-controls the running state of the motor: for instance, whenever the DUT "slides" the program should increase or decrease the motor velocity in order to obtain a better performance. An observation regarding the actual brake device is that it is electromagnetic and it is not suitable to brake force modulation, because by nature it is used for on/off applications; hence, a different brake system should be mounted, as well as another motor with variable resistors in series with its phases to modulate the torque generated [3][2]. Some other ideas are to adapt the entire system to test different types of motor, AC ones too, and then to design an embedded system board to replace the main board with its STB on top. In the end, also the external power supply will be removed to use a dedicated switching power supply.

Bibliography

[1] Richard Crowder. *Electric Drives and Electromechanical Systems: Applications and Control*. B&H, 2006.

[2] P. Chandrakand Dr. S. Ganeshkumar, R. Ashwath. Analysis of electromagnetic braking system. *International Journal of Scientific Research and Engineering Development*, 4(Issue 2):1–9, 2021.

[3] Saif Ali Shaikh Dr. S. R. Pawar, Shubham D. Satam. Analysis of electromagnetic braking system. *International Journal of Research in Engineering and Science (IJRES)*, 10(Issue 4):19–26, 2022.

[4] Alexander Gmiterko Ivan Virgala, Michal Kelemen. Control of stepper motor by microcontroller. *Journal of Automation and Control*, 3(3):131–134, 2015.

[5] *CAS/CASR/CKSR series Current Transducers*. LEM, 2008.

[6] *Real-Time Operating System, What it is and why you might want to use one*. Microsoft, 2020.

[7] Fulling Motor. Nema 23 stepper motor 57sh series, 2020, last consultation 10/08/2022. https://www.fullingmotor.eu/cat.pag/categoria-czk338kzpsxzkkzordxcrtzk1.html.

[8] *Electromagnetic brake, clutches, tooth type couplings*. MWM, 2020.

[9] *Instruction manual and data sheet Torque Sensor Series 2000*. NCTE, 2020.

[10] Kaushik Ghosh Pritam Chakraborty, Kingshuk Kundu. Application of electromagnetic braking torque and different braking modes programmed with atmega328p microcontroller in electromagnetic braking system. *Reason - A Technical Journal*, 20(Sept):13, 2021.

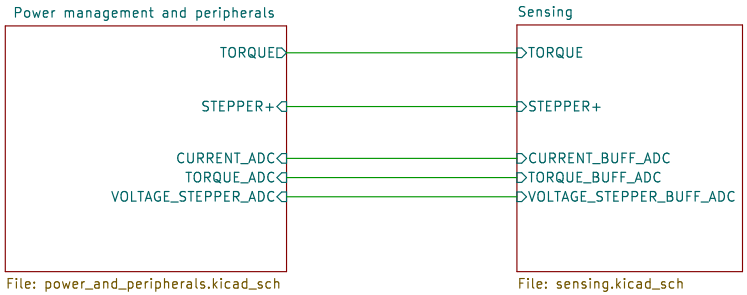
Acronyms

AC	Alternate Current.
ADC	Analog to Digital Converter.
BOM	Bill of materials.
CAD	Computer Aided Design.
CAN	Controller Area Network.
DC	Direct Current.
DMA	Direct memory access.
DUT	Device Under Test.
EM	electromagnetic.
EMC	electromagnetic compatibility.
EMI	electromagnetic interference.
FFT	Fast Fourier Transform.
GPIO	General Purpose Input Output.
GUI	Graphic User Interface.
LED	Light-emitting diode.
PCB	Printed Circuit Board.
RAM	Random Access Memory.
RTOS	Real Time Operating System.
SPI	Serial Peripheral Interface.
STB	Stepper Testbench Board.
UART	Universal Asynchronous Receiver-Transmitter.
USB	Universal Serial Bus.

Appendix A

Attachments

A.1 Expansion Board - Hardware Schematic



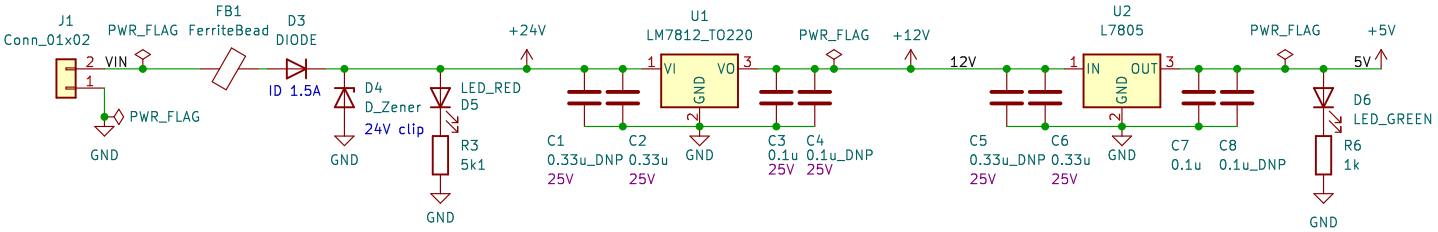
ProM Facility

Sheet: /
File: stepper_testbench.kicad_sch

Title: STB – Expansion Board

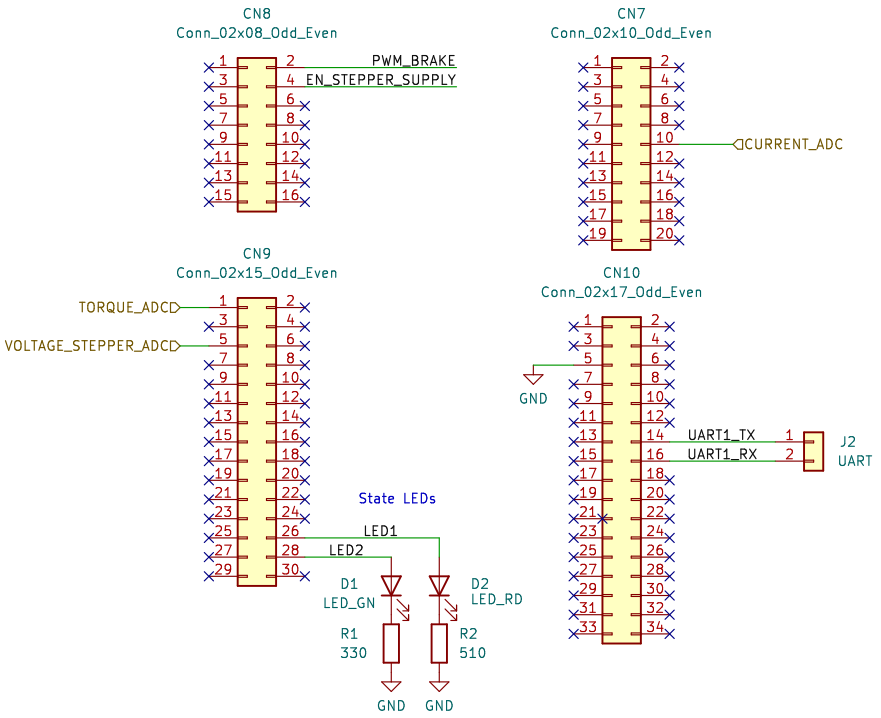
Size: A4	Date: 2022-08-09	**Rev: 0.2**
KiCad E.D.A. kicad (6.0.2-0)		Id: 1/3

POWER MANAGEMENT

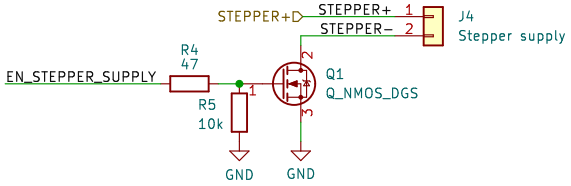


NUCLEO STM32F7 ZIO CONNECTORS

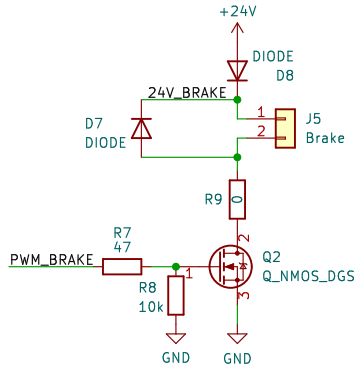
UART, State LEDs, PWM brake signal, ADC channels, GND



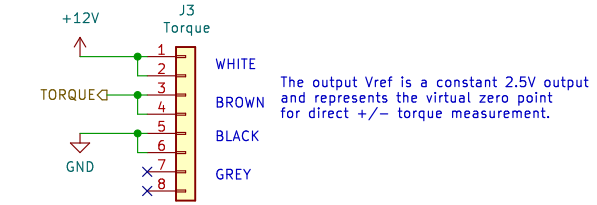
STEPPER DRIVER



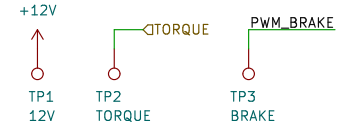
BRAKE DRIVER



TORQUE SENSOR



TESTPOINTS

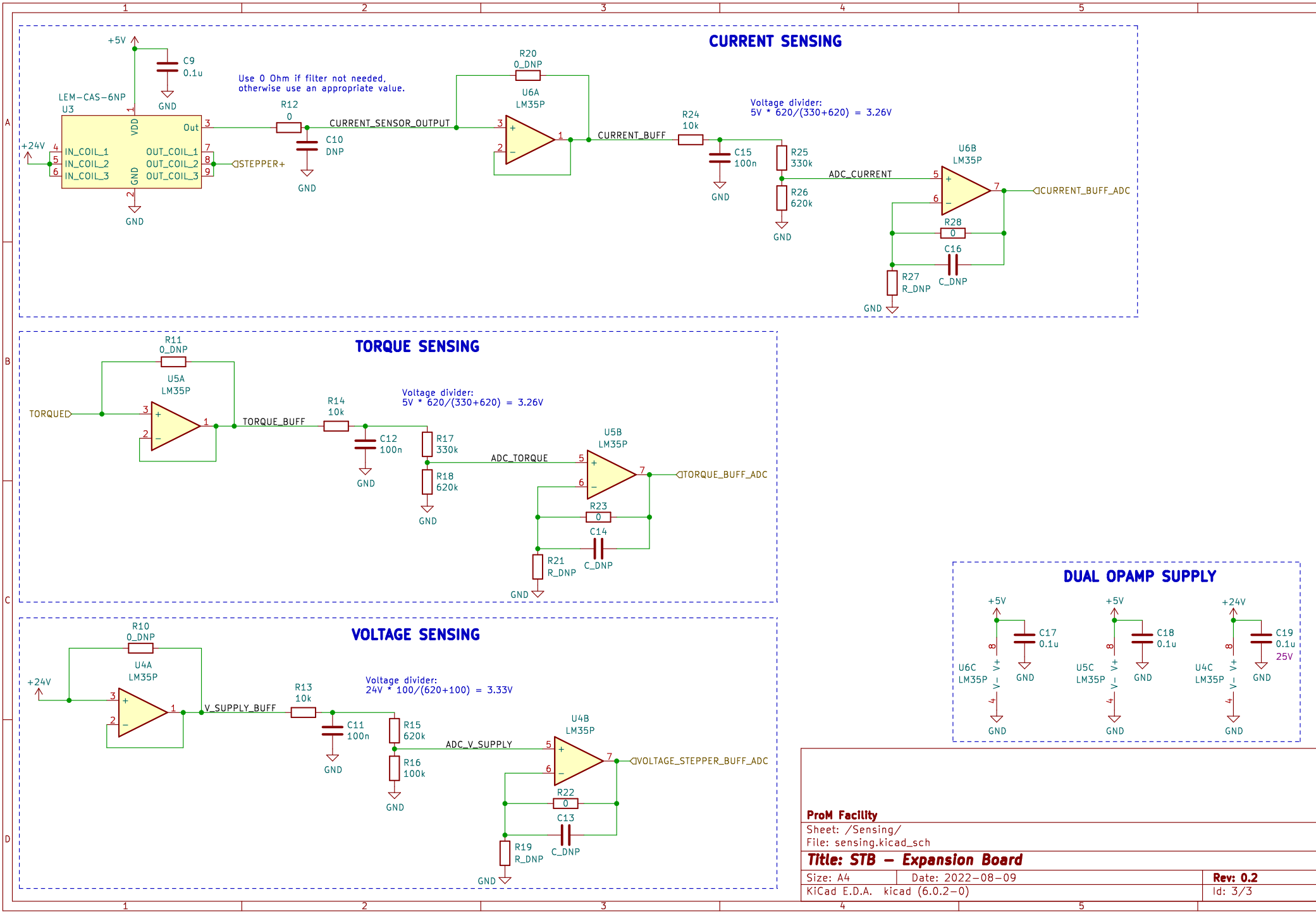


ProM Facility

Sheet: / Power management and peripherals/
File: power_and_peripherals.kicad_sch

Title: STB – Expansion Board

Size: A4	Date: 2022-08-09	**Rev: 0.2**
KiCad E.D.A. kicad (6.0.2-0)		Id: 2/3


CURRENT SENSING
+5V
C9
0.1u
GND
LEM-CAS-6NP
U3
VDD
Out
+24V
IN_COIL_1
IN_COIL_2
IN_COIL_3
OUT_COIL_1
OUT_COIL_2
OUT_COIL_3
GND
STEPPER+
Use 0 Ohm if filter not needed,
otherwise use an appropriate value.
R12
0
C10
DNP
CURRENT_SENSOR_OUTPUT
R20
0_DNP
U6A
LM35P
CURRENT_BUFF
R24
10k
C15
100n
Voltage divider:
5V * 620/(330+620) = 3.26V
R25
330k
R26
620k
ADC_CURRENT
U6B
LM35P
CURRENT_BUFF_ADC
R28
0
C16
C_DNP
R27
R_DNP
TORQUE SENSING
R11
0_DNP
U5A
LM35P
TORQUED
TORQUE_BUFF
R14
10k
Voltage divider:
5V * 620/(330+620) = 3.26V
C12
100n
R17
330k
R18
620k
ADC_TORQUE
U5B
LM35P
TORQUE_BUFF_ADC
R23
0
C14
C_DNP
R21
R_DNP
VOLTAGE SENSING
R10
0_DNP
U4A
LM35P
+24V
V_SUPPLY_BUFF
R13
10k
Voltage divider:
24V * 100/(620+100) = 3.33V
C11
100n
R15
620k
R16
100k
ADC_V_SUPPLY
U4B
LM35P
VOLTAGE_STEPPER_BUFF_ADC
R22
0
C13
C_DNP
R19
R_DNP
DUAL OPAMP SUPPLY
+5V
+24V
C17
0.1u
C18
0.1u
C19
0.1u
25V
U6C
LM35P
U5C
LM35P
U4C
LM35P
V+
V-
ProM Facility
Sheet: /Sensing/
File: sensing.kicad_sch
Title: STB – Expansion Board
Size: A4
Date: 2022-08-09
Rev: 0.2
KiCad E.D.A. kicad (6.0.2-0)
Id: 3/3

A.2 Expansion Board - Layout

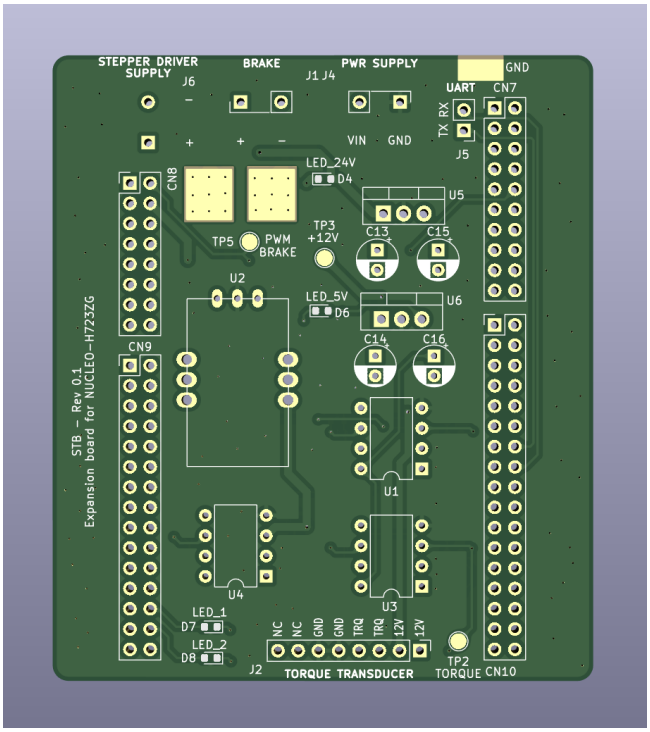


Figure A.1: STB top layout

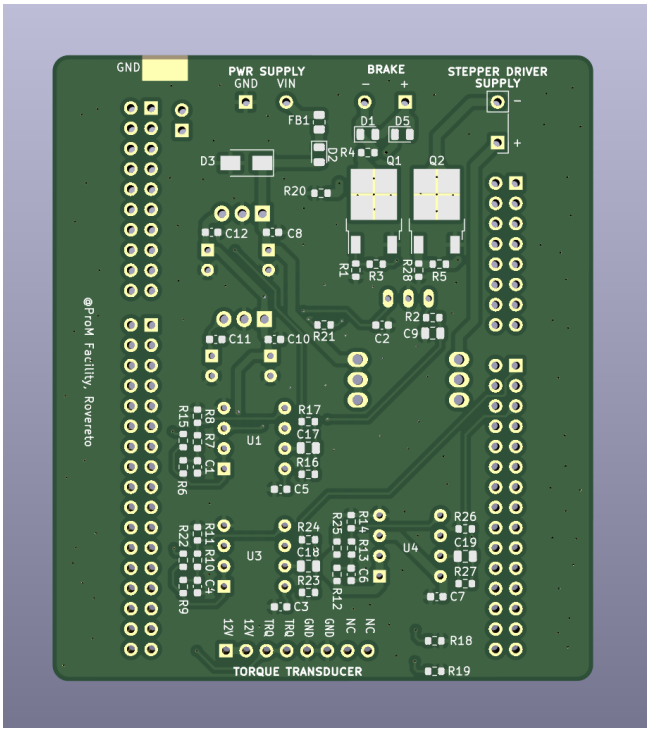


Figure A.2: STB top layout

A.3 Expansion Board - BOM

Reference(s)	Value	LibPart	Footprint
C1, C5	0.33u_DNP	Device:C	Capacitor_THT:CP_Radial_D5.0mm_P2.50mm
C2, C6	0.33u	Device:C	Capacitor_SMD:C_0603_1608Metric
C3, C7, C9, C17, C18, C19	0.1u	Device:C	Capacitor_SMD:C_0603_1608Metric
C4, C8	0.1u_DNP	Device:C	Capacitor_THT:CP_Radial_D5.0mm_P2.50mm
C10	DNP	Device:C	Capacitor_SMD:C_0805_2012Metric
C11, C12, C15	100n	Device:C	Capacitor_SMD:C_0603_1608Metric
C13, C14, C16	C_DNP	Device:C	Capacitor_SMD:C_0805_2012Metric
CN7	Conn_02x10_Odd_Even	Connector_Generic:Conn_02x10_Odd_Even	Connector_PinHeader_2.54mm:PinHeader_2x10_P2.54mm_Vertical
CN8	Conn_02x08_Odd_Even	Connector_Generic:Conn_02x08_Odd_Even	Connector_PinHeader_2.54mm:PinHeader_2x08_P2.54mm_Vertical
CN9	Conn_02x15_Odd_Even	Connector_Generic:Conn_02x15_Odd_Even	Connector_PinHeader_2.54mm:PinHeader_2x15_P2.54mm_Vertical
CN10	Conn_02x17_Odd_Even	Connector_Generic:Conn_02x17_Odd_Even	Connector_PinHeader_2.54mm:PinHeader_2x17_P2.54mm_Vertical
D1	LED_GN	Device:LED	LED_SMD:LED_0603_1608Metric
D2	LED_RD	Device:LED	LED_SMD:LED_0603_1608Metric
D3, D7, D8	DIODE	Simulation_SPICE:DIODE	Diode_SMD:D_SOD-323
D4	D_Zener	Device:D_Zener	Diode_SMD:D_SMA
D5	LED_RED	Device:LED	LED_SMD:LED_0603_1608Metric
D6	LED_GREEN	Device:LED	LED_SMD:LED_0603_1608Metric
FB1	FerriteBead	Device:FerriteBead	Resistor_SMD:R_0805_2012Metric
J1	Conn_01x02	Connector_Generic:Conn_01x02	Connector_PinHeader_2.54mm:PinHeader_1x02_P2.54mm_Vertical
J2	UART	Connector_Generic:Conn_01x02	Connector_PinHeader_2.54mm:PinHeader_1x02_P2.54mm_Vertical
J3	Torque	Connector_Generic:Conn_01x08	Connector_PinHeader_2.54mm:PinHeader_1x08_P2.54mm_Vertical
J4	Stepper supply	Connector_Generic:Conn_01x02	Connector_PinHeader_2.54mm:PinHeader_1x02_P2.54mm_Vertical
J5	Brake	Connector_Generic:Conn_01x02	Connector_PinHeader_2.54mm:PinHeader_1x02_P2.54mm_Vertical
Q1, Q2	Q_NMOS_DGS	Device:Q_NMOS_GDS	Package_TO_SOT_SMD:TO-252-2
R1	330	Device:R	Resistor_SMD:R_0603_1608Metric
R2	510	Device:R	Resistor_SMD:R_0603_1608Metric
R3	5k1	Device:R	Resistor_SMD:R_0603_1608Metric
R4, R7	47	Device:R	Resistor_SMD:R_0603_1608Metric
R5, R8, R13, R14, R24	10k	Device:R	Resistor_SMD:R_0603_1608Metric
R6	1k	Device:R	Resistor_SMD:R_0603_1608Metric
R9, R12, R22, R23, R28	0	Device:R	Resistor_SMD:R_0603_1608Metric
R10, R11, R20	0_DNP	Device:R	Resistor_SMD:R_0603_1608Metric
R15, R18, R26	620k	Device:R	Resistor_SMD:R_0603_1608Metric
R16	100k	Device:R	Resistor_SMD:R_0603_1608Metric
R17, R25	330k	Device:R	Resistor_SMD:R_0603_1608Metric
R19, R21, R27	R_DNP	Device:R	Resistor_SMD:R_0603_1608Metric
TP1	12V	Connector:TestPoint	TestPoint:TestPoint_Pad_D2.0mm
TP2	TORQUE	Connector:TestPoint	TestPoint:TestPoint_Pad_D2.0mm
TP3	BRAKE	Connector:TestPoint	TestPoint:TestPoint_Pad_D2.0mm
U1	LM7812_TO220	Regulator_Linear:LM7812_TO220	Package_TO_SOT_THT:TO-220-3_Vertical
U2	L7805	Regulator_Linear:L7805	Package_TO_SOT_THT:TO-220-3_Vertical
U3	LEM-CAS-6NP	Sensing lib:LEM-CAS-6-NP	Sensing lib:CAS-6-NP
U4, U5, U6	LM35P	Device:Opamp_Dual	Package_DIP:DIP-8_W7.62mm

Figure A.3: Bill of Material of STB Expansion Board