

Motion Control Unit Design for Control Prototyping of Modern BLDC/PMSM Drives and Piezo Actuators

Květoslav Belda¹^a, Pavel Píša^{1,2}^b and Štěpán Pressl^{1,2}^c

¹*Department of Adaptive Systems, The Czech Academy of Sciences, Institute of Information Theory and Automation, Pod Vodárenskou věží 4, 182 00 Prague 8, Czech Republic*

²*Department of Control Engineering, Czech Technical University in Prague, Faculty of Electrical Engineering, Karlovo náměstí 13, 120 00 Prague 2, Czech Republic
belda@utia.cas.cz, {pisa, pressste}@fel.cvut.cz*


Keywords: BLDC/PMSM Drives, Motion Control, Power Supply, Mechatronic Systems, PysimCoder, Real-Time Generated Code, NuttX RTOS, Piezoelectric Bender Actuators, Rapid Prototyping.


Abstract: This paper deals with the design and construction of an open-software and open-hardware motion control unit intended for experimental development and rapid prototyping of advanced drive control for mechatronic systems and robotic applications such as smart production lines, manipulators, and robotic machine tools. The control unit is designed for high dynamic drives, both modern permanent magnet synchronous motors and piezoelectric drives and actuators promising for the near future. The concept and construction of the considered control unit are presented both from the hardware and software point of view. This includes custom printed circuit boards, electronic components for communication and power outputs, the microcontroller, firmware as well as software used to generate control application code. The presented experimental research and development is illustrated by figures and records of measured data.


1 INTRODUCTION

Many motion controllers work in industrial and home applications such as robots-manipulators, production lines and machines, unmanned vehicles and others. Therefore, the motion control matter is well known among engineers and the amount of motion controllers in industry is vast. On the market, the supply ranges from controllers working only with a specific actuator (motor) with closed firmware to more affordable and available controllers.

However, in the early stages of developing or researching new control algorithms, it can be advantageous to have a generic prototyping platform. Thus, for fast application development, controllers are required to enable rapid prototyping using model-based design, capable of running the designed control algorithms (Tachiquingutierrez et al., 2023). In this paper, we propose a modular and extensible, yet relatively low-cost, embedded platform with a strong emphasis on running control applications built using model-based design tools that generate real-time code. Firstly, we discuss the existing solutions and then we introduce the main parameters of a proposed control unit.

^a <https://orcid.org/0000-0002-1299-7704>

^b <https://orcid.org/0000-0003-3476-5151>

^c <https://orcid.org/0009-0006-5364-8961>

1.1 Existing Tools and Platforms

Let us analyse available solutions like ODrive (ODrive Robotics, 2024), VESC (Vedder, 2022) and SOLO (SOLO Motor Controllers SRL, 2024) controllers that all focus on the control of Permanent Magnet Synchronous Motors (PMSM) and Brushless Direct Current (BLDC) drives. These controllers share a few similarities: *i*) they use mostly the same interfaces, such as UART, SPI, I²C, CAN, and USB; however, for rapid prototyping, the Ethernet interface can be useful for convenient data logging and configuration, even from all over the world; *ii*) the other similarity is that each controller features its API to configure control algorithms; on our proposed platform, we intend not to come with a custom API but rather provide a nonrestrictive environment, where the generated code of control algorithms could be run; *iii*) the last similarity is the lack of more than three phases per axis, making control of stepper motors or other types of actuators not possible.

There are already several solutions available for *Rapid Control Prototyping* (RCP), such as Speedgoat (Speedgoat GmbH, 2024) and dSPACE (dSPACE GmbH, 2024). However, these systems are expensive and are tightly linked up to MATLAB/Simulink. In contrast, we propose a unit based on an open and free code generation tool.

1.2 Proposed Platform

The platform is intended to be fully open-source and open-hardware to remain maintained and technologically sustainable. A large number of communication interfaces are also required for communicating with sensors, interfacing with the microcontroller and logging purposes, such as Ethernet, I²C, SPI, USB, CAN, etc., as well as functional parts such as *Pulse Width Modulation*, *Analog-to-Digital Converter* and *Timer* peripherals for the specific design of *field oriented control* BLDC/PMSM drives. The generated code adds computational overhead, so a microcontroller with a powerful core is required.

The main focus is on the control of PMSM/BLDC, Stepper (even with feedback sensors), and Brushed DC motors. However, the platform can also be used for special actuators e.g. using inverse piezoelectric effect and others in mechatronics (Gao et al., 2020).

We require the controller to be able to drive at least two stepper motors, therefore eight controllable phases are needed. Eight phases are also enough for BLDC/PMSM motors with more than three phases. With a higher number of phases, a utilised motor can be still operated when one or more phases are failed. For these advantages, the multi-phase BLDC motor is taken a good candidate for energy saving or military application (Moon et al., 2015).

The controller is intended as a generic prototyping platform with a large number of interfaces, the size and weight are not a matter. If the control algorithms are designed and verified on this platform, the hardware can be redesigned to fit certain criteria, such as size, weight, safety-critical, or space compliance, keeping the same algorithms. Despite all the proposed features, we strive for an affordable solution within motion controllers such as ODrive controllers.

Finally, to make the generated code portable across various platforms, a *Real-Time Operating System* (RTOS) is considered. It provides a runtime environment for multithreaded applications, network stacks and a unified API for accessing hardware dependent peripherals. In this paper, NuttX RTOS is selected as a suitable OS. Furthermore, as an experimental but open source code generation tool, *pysimCoder* is used, capable of generating NuttX-compatible code from block diagrams and remotely tuning its internal parameters over the network via the Silicon Heaven protocol (Lenc et al., 2023).

The paper is organised as follows. Hardware and its implementation are in Sections 2 and 3. Then, the hardware adaptation to NuttX is described alongside the *pysimCoder* suite in Section 4. Finally, the *pysimCoder* applications are shown in Section 5.

2 HARDWARE SPECIFICATION

Hardware (HW) design is inspired by the LX-RoCon controller (PiKRON, 2014). One of the design rules is to divide the motion controller into a *Micro-Controller Unit (MCU)* board and a *Power stage* board as two printed circuit boards. The complete motion control unit then comprises of these two connected boards.

This design rule allows to swap the less complex *Power stage* board for a different one while keeping the more complex *MCU* board the same. Since the *MCU* board accommodates most of the used connectors, it is placed above the power board. The power board acts as the 5 V voltage source for the *MCU* board as well.

2.1 Hardware Analysis

To design and specify motion control unit, a basis for selecting a micro-controller, the intended peripherals and their use cases were summarised as follows: TTL UART for Simple console user interface; RS232, RS422 or RS485 for Communication with other devices, sensor data acquisition; (Q)SPI for Ext. SPI memory, sensor data acquisition; I²C for Ext. I²C memory, sensor data acquisition; Counter for position measurement from quadrature encoders; JTAG or SWD for program flashing and debugging; CAN for communication with other devices; Ethernet for data logging, remote access, *pysimCoder* parameter tuning; SD slot for data logging; PWMs for Control of power switches; ADCs for current measurement for motor *field oriented control (FOC)* and GPIOs for reading from hall sensors.

The motion controller should be capable of driving at least two stepper motors simultaneously since we require the unit to be used for SCARA-like or Cartesian robot topologies and axles with two independently driven wheels.

Since a stepper motor requires 4 half-bridge switches, 8 half-bridge switches in total are required. BLDC/PMSM requires 3 half-bridge switches to be controlled and a DC brushed motor requires 2 half-bridge switches (forming a H-bridge). Choosing 8 half-bridge switches is enough for controlling 2 BLDC/PMSM and 4 DC brushed motors as well. Furthermore, at least 8 PWM outputs for power switches and at least 8 ADC channels for current measurement are necessary with two counter peripherals for encoder pulse counting and all the communication interfaces in paragraph above. Finally, all components should operate in industrial temperature range of at least -40 to 85°C .

2.2 Microcontroller

ATSAMV71Q21B from Microchip, hereinafter abbreviated SAMV71 (Microchip, 2023), was chosen as a suitable microcontroller (alternatives are e.g. series STMicroelectronics STM32H7 or the NXP imxRT). According to this selection, the control unit has the label *SaMoCon*.

The microcontroller core is ARM Cortex M7 with support for single-precision and double-precision floating point unit, making it feasible to use alongside *pysimCoder* generated code with *double* 64bit type in calculations. The core is running at 300MHz and features 2MB of Flash memory, alongside 384kB of SRAM.

This microcontroller features all the desired peripherals, ranging from Ethernet, to two PWM peripherals featuring eight channels in total with complementary outputs with configurable deadzones, two *Analog Front End Controller* (AFEC) peripherals each having up to 11 AD channels at disposal and four TC (Timer/Counter) peripherals with encoder counter logic. Besides that, CAN peripherals, the HSMCI peripheral for SD interfacing and many UARTs, SPIs, TWIHSs (Microchip’s I²C) are present too.

3 HW IMPLEMENTATION

AS for the first power stage board prototype, Infineon IFX007T half-bridge switches (Infineon, 2018) were selected, see Figure 1. Since the maximum rated voltage is 40 V, this first power board has a nominal rating of 24 V. To reduce the component cost, the common ground is shared with the MCU board, as no galvanic isolators are needed.

IFX007T is an all-in-one switch featuring two N power-switching MOSFETs alongside gate drivers. It includes built-in high and low-side current limitations alongside over-temperature protection. This

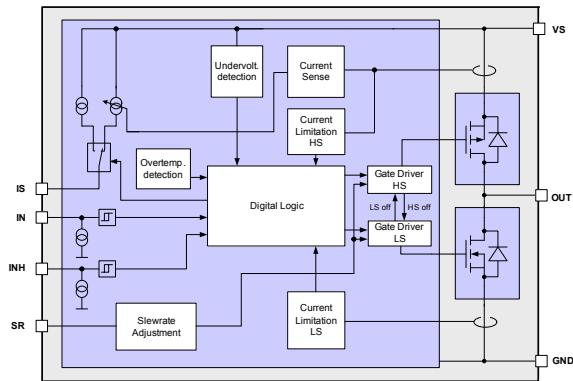


Figure 1: IFX007T structure (Infineon, 2018).

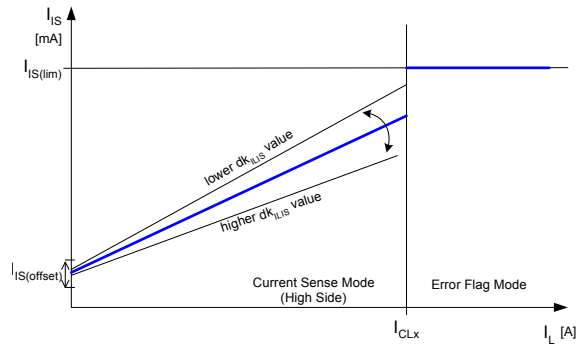


Figure 2: IS output current (Infineon, 2018).

makes the switch very easy to use, as it only requires one PWM driving signal (IN) and an inhibit (INH) input controlled by a GPIO.

The half-bridge also has an IS (current-sense) output whose current is proportional to the current flowing through the high-side transistor to the load. The high-side current can then be measured by measuring the voltage drop across the connected resistor. The IS pin also serves the purpose of a fault indication in case of over-current or high temperature. The characteristics of IS output current are shown in Figure 2.

Even though the high-side current can be measured using the IS output, a decision was made to measure the low-side transistor current using a shunt connected to ground. However, IS high-side value was used as an indicator of average current by connecting a parallel capacitor, and the indicator of a fault.

As Figure 2 suggests, the fault output current is bigger than the expected current during normal operation. This way, a fault voltage drop can be compared with a reference on a comparator. Therefore the comparator with open drain/collector outputs was chosen for easy chaining with other comparators.

The low-side shunt is used for more precise measurement of the motor winding current. The current measurement and fault generation scheme is depicted in Figure 3. If the voltage drop on the R_{IS} resistor is bigger than V_{ref} , the comparator output goes low.

Since the operational amplifiers must be connected to the analog ground, the measurement of the shunt R_S is done using a differential amplifier for R_S is connected to the power ground, which is affected by voltage drop caused by high currents in the power stage.

As the current passing through the transistors can be negative, a bias voltage is applied to the differential amplifier, shifting the output voltage of the amplifier. Fast voltage spikes on the R_{IS} resistor are filtered by the parallel C_{IS} capacitor.

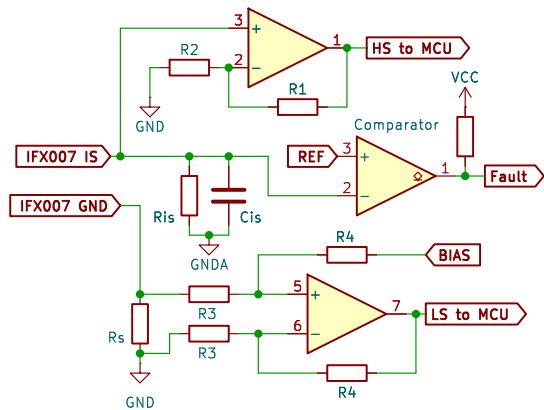


Figure 3: HS and LS current sensing with fault generation.

The output voltage formulas of the amplifiers are as follows according to Figure 3:

$$V_{HS} = \left(1 + \frac{R_1}{R_2}\right) R_{IS} I_{IS}, \quad (1)$$

$$V_{LS} = \frac{R_4}{R_3} R_S I_{RS} + V_{Bias}, \quad (2)$$

where I_{IS} is the current sourced by the IFX007T IS pin and I_{RS} is the current passing through the R_S shunt, which is in fact the low-side current.

3.1 MCU Board Implementation

This section briefly describes the implementation of the upper control *MCU board*, with all the peripherals shown in Figure 4. The connection of two boards is placed on one of the longer sides of the PCB. Extra GPIOs, SPI, and I²C are useful for future extensions. There is another connector on top of the MCU board with the same mirrored pinout as the lower interconnection which can be used for expanding boards placed above the MCU board and as an oscilloscope probing place. In the following sections, the peripherals routed to the separate peripheral connectors on the MCU board will be referred to as *main*

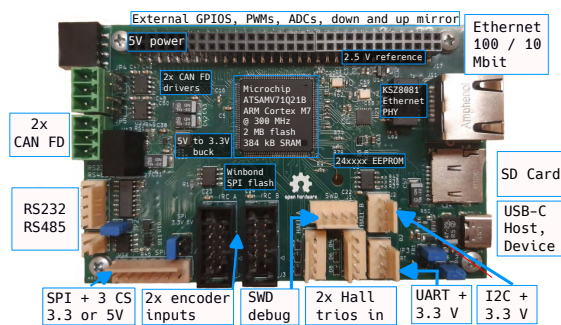


Figure 4: Upper control MCU board with peripherals.

Table 1: Routed peripherals.

<i>SaMoCon Peripheral</i>	<i>SAMV7 peripheral</i>
2x CAN	MCAN0, MCAN1
SPI main & extra	SPI0, SPI1
TTL Console	UART3
RS232/RS485	USART2
Ethernet	GMAC in RMI mode
USB-C	HSUSB
2x IRC	TC[0,2] (Timer/Counter)
SD Card	HSMCI
H PWM out, Motor A	PWM0 CH0-3
L PWM out, Motor A	PWM0 CH0-3
H PWM out, Motor B	PWM1 CH0, CH1, CH3
L PWM out, Motor B	PWM1 CH0-3
8 ADC chn., Motor A	AFEC0
8 ADC chn., Motor B	AFEC1
3 extra ADC chn.	AFEC1
2x 3 Hall inputs	GPIOs on the same PIO
I ² C main & extra	TWIHS0, TWIHS1

peripherals whereas the peripherals routed to the interconnection/expanding connector will be referred to as the *extra peripherals*.

3.1.1 Microcontroller Pinout

There are many combinations of pin configurations due to alternative functions of microcontroller pins. However, for a high count of the desired peripherals, conflicts began to occur – two peripheral outputs or inputs were on the same pin. See Table 1 for routed peripherals and the corresponding SAMV71 peripheral. Each IFX007T half-bridge is controlled by one PWM H (high-side) signal and one GPIO pin. Even though we need 8 PWM H outputs, we also route all remaining PWM L (low-side) outputs, because some gate drivers require complementary PWM outputs. In the case of IFX007T, the L output can be configured as a simple output pin.

Unfortunately, the only one H output of PWM1 CH2 could not be routed due to collisions with the HSMCI peripheral. A decision was made to route the L channel output as the H channel, because the PWM polarity can be configured in software, resulting in a total of 7 complementary and one non-complementary output.

3.1.2 MCU-Board Voltage Supplies

All the components on this PCB are powered from the 5V or 3.3V rail. MCU board is designed to be powered from the power board through the interconnection. The 5V rail can also be supplied from the USB-C connector. A buck regulator

(Texas Instruments TPS562207) is used to convert 5V to 3.3 V. The CAN and RS232/RS485 peripherals are galvanically isolated, so 5 V to 5 V small power supplies are used. Also, a stable 2.5 V reference (Microchip MCP1525) was added for analog measurements.

3.1.3 Ethernet

A Physical Layer (PHY) component was connected to the microcontroller GMAC peripheral. The Microchip KSZ8081 IC was used, working in the RMI mode with specific bit rates up to 100 Mb/s. The design uses the recommendations from the datasheet (Microchip, 2019) with compatible magnetics. Communication activity is indicated by two LED.

3.1.4 USB

The USB-C's CC1 and CC2 pins are connected to the Texas Instruments TUSB321 integrated circuit that negotiates host or device operation, given voltage levels at certain pins. The data pins are connected to the HSUSB SAMV71 peripheral. The board was designed to act as a host or a device, using an AP2171W power switch. If the controller acts as a host, the enable pin (EN) is set high, enabling the power switch. If over-current happens, an open collector FLG output is used to indicate an error. If connected to a USB power source, the current passes through a Schottky diode with a low voltage drop and bypasses the USB power switch, see Figure 5.

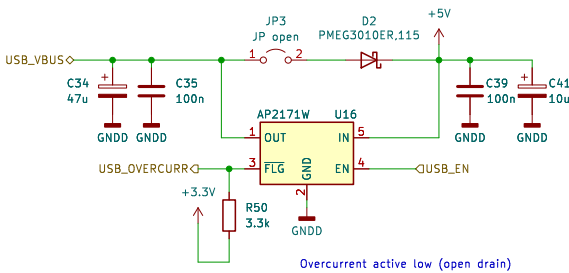


Figure 5: USB power components.

3.1.5 Other Communication Interfaces

The MCU board features two CAN-FD peripherals with MCP2562FD transceivers. The board features a circuit that either functions as the RS232 or the RS485 interface. The RS232 interface uses a MAX232 converter and the RS485 uses the ST485 converter. CAN and RS interfaces are both isolated by the ADuM1201AR isolators.

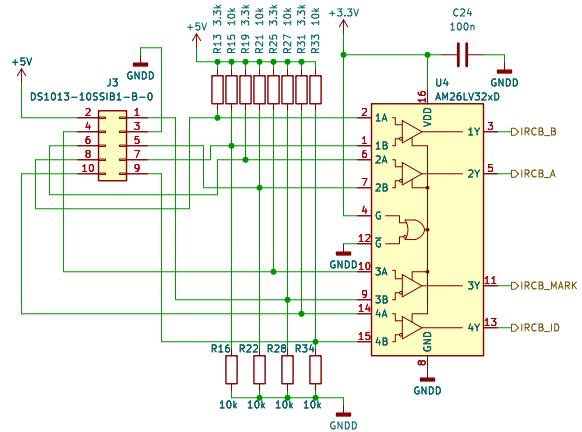


Figure 6: Routed optical encoder with HEDL series pinout.

There are also main and extra TWIHS (I²C) interfaces routed. The main TWIHS peripheral has a Microchip 24AA64 EEPROM connected and the signals are routed to a 4 pin Molex connector alongside 3.3 V and GND. There is another separate 4-pin Molex connector for a basic LVTTL console user interface, with 3.3 V, GND and RX and TX signals.

3.1.6 Feedback from the Actuators

The board has 2 connectors for incremental encoders and 2 Hall input trios at its disposal. The optical encoder connector is compatible with the HEDL encoder series with a 2x5 pin socket. The HEDL series connector incorporates differential two optical phases, a differential zero-cross index, grounding and a 5 V power output too.

The differential signals are decoded by the Texas Instruments AM26LV32 receiver. A so-called *mark* signal is used too, acting as another user-defined helpful synchronising signal. The scheme of the decoder circuit is in Figure 6.

Since the voltage level for the inverted signals in the disconnected state is defined by voltage dividers, then an encoder with single-ended outputs can be connected to the positive inputs. All the inputs have a pull-up in case of open drain/collector outputs. The decoded signals from the receiver are connected to the SAMV71 TC0 and TC2 (*Timer/Counter*) peripherals.

The Hall outputs can be read by a GPIO pin. In our implementation, a standard 5-pin Molex connector (5 V power, GND, and 3 Hall outputs) is used. In general, the outputs can be either push-pull or open-drain. However, the 5 V push-pull output could potentially destroy the microcontroller, meaning precautions must be taken. The way Hall inputs are routed is shown in Figure 7.

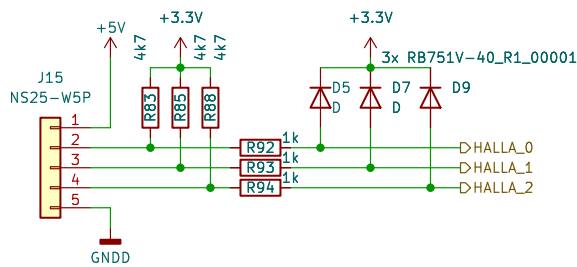


Figure 7: Open drain/collector output: the output is 3.3 V or GND due to the pull-up to 3.3 V. Push-pull: the voltage limitation is done by shorting the output to 3.3 V through diodes. The shorting current is limited by series resistors.

3.1.7 Interconnection Pinout

The interconnection between the two boards is realized by a standard 2×32 header pin connector with a 2.54 mm pitch. The MCU board contains the pin connector heading down towards the power board and a socket heading up. The pinouts of these connectors are the same, mirrored only, and are of SMD type, allowing us to route tracks in the inner layers.

Some pins provide the 5 V source and the power digital ground to power the MCU board. In general, the power board may contain 3.3 V powered circuits, so we need to route 3.3 V too, as well as the 2.5 V reference. The lower board may also contain analog circuitry so we route the analog ground too. The connection of the analog and digital ground is done on the MCU board nearby the ADC reference and ground pins.

All 15 (7 complementary and one non-complementary) PWM outputs must be routed alongside 16 ADC channels (IFX007 high and low side sensing) for two motors. An extra I²C, GPIOs and SPI or 3 extra ADC channels are routed too.

3.2 Power Stage Board Implementation

As already discussed during the analysis, the power board also serves the purpose of a step-down to 5 V. The rated current per phase is a continuous 10 A minimum, achieving a 20 A peak is possible too. The board must perform the current measurement and amplification. These analog signals are then processed by the MCU board placed above. The power board is in Figure 8.

The power to the half-bridge switches is routed on the PCB upper side, while the outputs are routed on its back side. The IFX007 output with an exposed solder pad is connected to a high area copper, on which a heatsink can be placed to cool down the switches.

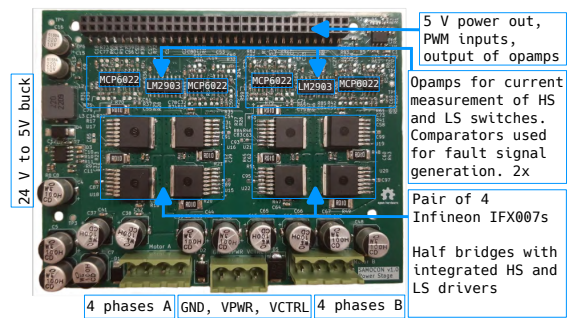


Figure 8: Power stage board

3.2.1 Power Components

The power board features a 3-pin power connector, one pin being the common ground pin, the other ones are the positive voltage rails, one for the MCU board, and the other for the motors. The used buck regulator IC is AOZ1284, used in accordance with the datasheet's recommendations. The total power consumption of the MCU board is supposed around 0.5A, the MCU board's USB power switch handles 1A continuous at maximum and we suppose the expanding board consumes another 1A. The IC provides up to 4A, enough for our requirements.

All the analog circuitry powered from 5 V or 3.3 V (provided by the MCU board) is first filtered with an LC filter. The power rail for the IFX007 switches contains decoupling electrolytic capacitors.

3.2.2 Low Side Shunt Current Measurement

Let us consider the formula (2). Let V_{Bias} be 1.25 V, in the middle of the voltage reference. This voltage is obtained by choosing a 1:1 voltage divider with a buffer to lower the output impedance of the divider. R_s was chosen to be 10 m Ω because higher resistance values would lead to big power losses. The power loss of this at 10 A is 1 W, so a 3 W 2512 resistor has been chosen. To use the whole ADC bandwidth, we amplify the voltage by at least $10 \times$. This can be achieved by setting $R_3 = 1 \text{ k}\Omega$ and $R_4 = 10 \text{ k}\Omega$.

3.2.3 IS Current Measurement

Since the IFX007T IS output indicates a fault, we need to set the threshold value first which should equal to the point where $I = 20 \text{ A}$. Referring to Table 11 in (Infineon, 2018), the maximum analog sense current in fault condition is $I_{IS(lim)} = 6.1 \text{ mA}$. To be sure, the voltage drop on R_{IS} must not be higher than 3.3 V to not damage the operational amplifier, powered from the 3.3 V rail. We choose a $R_{IS} = 510 \Omega$ resistor from the E24 series. The maximum voltage

drop across this resistor is

$$V_{IS(lim)} = 3.111 \text{ V} < 3.3 \text{ V}. \quad (3)$$

The differential current sense ratio in static on-condition is on average (Infineon, 2018)

$$\frac{dI_L}{dI_S} = 19500. \quad (4)$$

Then, a threshold value for the comparator at $I = 20 \text{ A}$ can be computed as follows:

$$V_{th} = \frac{20}{19500} 510 \text{ V} = 0.52 \text{ V}. \quad (5)$$

Since the differential ratio varies significantly and it also has a non-negligible offset, we set $V_{th} = 0.655 \text{ V}$, provided by a divider made out of $2.2 \text{ k}\Omega$ and $6.2 \text{ k}\Omega$ resistors. This compare value is again amplified by a buffer to lower the output impedance.

The last value that needs to be set is C_{IS} . This value was supposed to be in the range of hundreds of pF, so only the high-frequency noise would be filtered. Unfortunately, during testing, we spotted a high fault rate on the comparator's output due to fast voltage spikes on R_{IS} which were always caused after switching off the PWM signal. The reasons of this problem are unknown to us and will require further investigation. Currently C_{IS} is 22 nF , as a trade-off between suppressing the spikes and fast-rising times.

3.2.4 Analog Components

Used operational amplifiers are Microchip MCP6022, a low offset and rail-to-rail operational amplifier. The used comparator is Texas Instruments LM2903, a dual differential comparator with open-drain outputs. That means all comparators from 4 half bridges can be chained together. If at least one switch faults, the line goes low. Finally, the complete motion control unit is shown in Figure 9 assembled of MCU and power boards. This ends the issue of HW design and implementation.

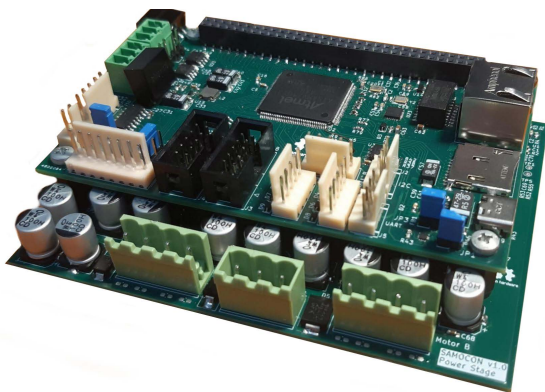


Figure 9: Connected boards forming the controller.

4 NuttX AND pysimCoder

This section briefly introduces NuttX and its adaptation to the *SaMoCon* hardware and pysimCoder suite.

NuttX is an open-source real-time operating system written in C and released under Apache 2.0 license was introduced by Gregory Nutt in 2007, (ASF, 2023). Currently, the project is maintained on GitHub (ASF, 2024c). The primary governing standards in NuttX are POSIX and ANSI standards (ASF, 2024a).

Since NuttX aims to conform to Portable Operating System Interface (POSIX) (Group, 2024), all the interaction with the microcontroller peripherals is done by accessing the peripherals' registered `/dev` files (for example, an AD converter may be registered as `/dev/adc0`). The files are accessed using system calls like `open`, `close`, `read` or `write`. As many peripherals do not share many similarities with each other, `ioctl` calls are commonly used to change the underlying device parameters. The project's directory structure is similar to the *Linux Kernel's* one (Torvalds, 2024), as well as the process of configuration and building. The project must be first configured by choosing `Kconfig` options and then built using `make` or `CMake`.

4.1 NuttX Adaptation

To build NuttX, a custom BSP (*board support package*) is needed. The `samv71-xult` BSP for the *SAMV71 Xplained Ultra Evaluation Kit* was used as the baseline. Multiple source files were added to initialize and register the peripherals, the most important are AD converters (`/dev/adc0,1`), PWM (`/dev/pwm0,1`), quadrature decoders of rotary encoders (`/dev/qe0,1`) and GPIOs for HALL reading and IFX007T INH activation.

Furthermore, the NuttX build needs to be configured so all the peripherals work properly, i.e. console driver to output on the SAMV71 UART3 peripheral; SAMV71 GMAC peripheral to communicate with the KSZ8081 PHY and enabling the UDP/IP and TCP/IP, alongside helper applications like `ping` or `telnet`. Also the SAMV71 quadrature decoder, PWM and ADC drivers needed to be configured. Changes to SAMV71 quadrature decoder and PWM drivers were made, which also made it into the GitHub mainline repository.

By default, a NuttX configuration uses a periodic timer interrupt that drives all system timing. The timer is provided by architecture-specific code that calls into NuttX at a rate controlled by `CONFIG_USEC_PER_TICK`. The default value

of `CONFIG_USEC_PER_TICK` is 10000 microseconds which corresponds to a timer interrupt rate of 100Hz ((ASF, 2024b), Tickless OS). This configuration must be turned on for control applications because it introduces a *freerunning timer*, where each counter increment corresponds to the smallest measurable time slice. If time slice is small, high precision delays can be made, useful in a fast periodic sampling of the `pysimCoder`-generated real-time applications.

4.2 Introduction to `pysimCoder`

`PysimCoder` is an open-source rapid control application development tool designed by Professor Roberto Bucher from the University of Applied Sciences and Arts of Southern Switzerland, (Bucher, 2024). It uses an extended python-control library for the integration of methods as full or reduced state space observer, anti-windup mechanism and discrete linear quadratic regulator. Apart from that, `pysimCoder` offers a graphical editor with a code generator. This combination allows the user to design the desired application graphically using predefined blocks and then generate a C code that can be run on target hardware. Details and `pysimCoder` setup and compilation is in (Lenc et al., 2023).

`PysimCoder` supports code generation for various platforms using custom APIs, like STMicroelectronics' STM32H7, Microchip SAMD21, or Linux. Simulations of control systems can be done if the code is generated for Linux. In the context of this paper, the ability of code generation for the OSIX-compatible NuttX RTOS is more important. As with NuttX, the generated code runs in a high-priority task, with a periodic control loop. The sampling time of the control loop can be set in `pysimCoder` GUI before the code generation. Runtime monitoring and tuning of model parameters that allow the user to display and edit parameters, inputs, and outputs of individual blocks, is solved using Silicon Heaven infrastructure (SHV) (Elektroline, 2024) and (Lenc et al., 2023).

5 `pysimCoder` APPLICATIONS

This section presents the control applications and its `pysimCoder` diagrams. The behavior of the control system can be remotely tuned using the Silicon Heaven protocol, like the references or the values of the PID controllers. Unfortunately, the current build of NuttX is only capable of achieving sampling rates below 1kHz, probably due to the NuttX scheduler limitations or a long-lasting interrupt, mak-

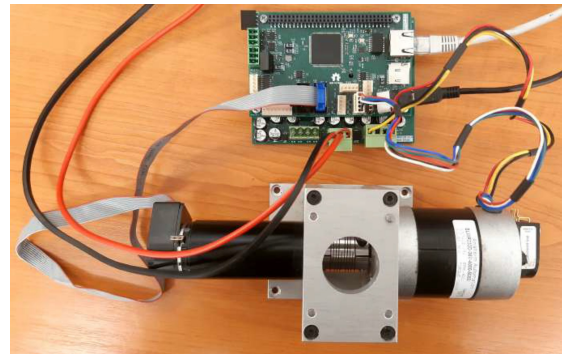


Figure 10: The control unit with a PMSM and an unused DC motor serving as a brake.

ing it unusable for the control of fast-moving systems or the *field oriented control*. This issue will require further investigation in the future. Fortunately, there are ways to debug the control applications, either using the low-level ARM Cortex-M profiling features using the *Instrumented Trace Macrocell* or using NuttX internal profiling tools (ASF, 2023).

The purpose of this section is to show that `pysimCoder` is ready to create control applications, even the ones used for PMSM *field oriented control*, as well as its logging capabilities.

5.1 Control of PMSM

The unit connected to PMSM is shown in Figure 10. The parameters of the used PMSM are in Table 2. The PMSM was tested simply in (Belda and Píša, 2021). Despite the sampling issues, a diagram of the current PI controller of dq axes is presented, showing that `pysimCoder` is ready to create FOC applications. Afterward, we present a simple non-vector PMSM control.

5.1.1 Current Controllers

To verify and tune current PI control, synchronous current-constant mode is used (see diagram Figure 11). The rotor is rotated in a synchronous way by generating an angle from an integrating angular ve-

Table 2: Parameters: PMSM BLWR233D-36V-4000

<i>Symbol</i>	<i>Description</i>	<i>Value</i>
P	rated power	95W
R_S	stator resistance	0.64 Ω
L_S	stator inductance	0.0021 H
Ψ_M	PM rotor magnetic flux	0.0200077 Wb
B	viscous friction coef.	$\approx 0 \text{ kg m}^2 \text{ s}^{-1}$
p	number of pole pairs	4
J	moment of inertia	$1.2 \cdot 10^{-5} \text{ kg m}^2$

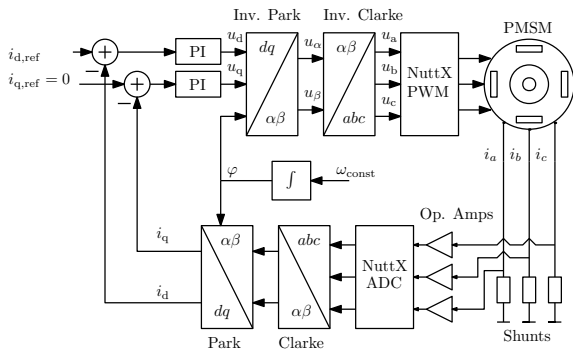


Figure 11: A simplified diagram of current controllers.

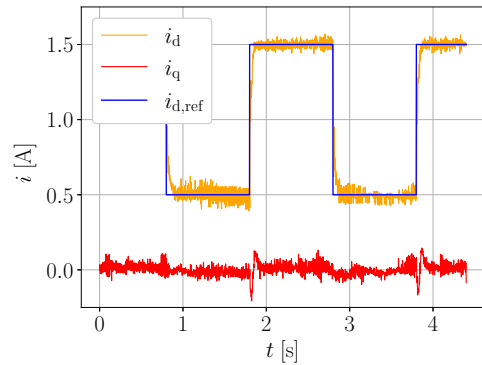


Figure 13: The PI dq currents control, $i_{q,ref} = 0$.

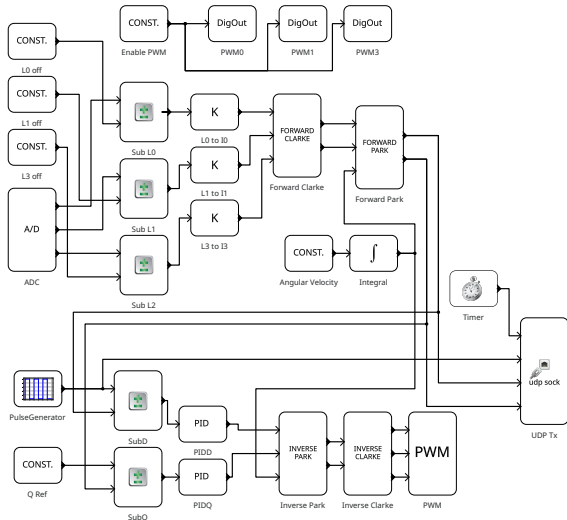


Figure 12: PysimCoder diagram of dq current control.

locity (see the integrator) while applying action only in the d -axis, while q -axis action is zero. The angular velocity must be set low. The measured currents i_a , i_b and i_c are acquired by reading the voltage drop on the current shunts with an AD converter. These readings are recalculated to obtain the i_q and i_d currents using the Clarke and Park transformations. The angle used in the Park transformation comes from the integrator.

These quantities are subtracted from the references and passed to inner PI controllers. Afterward, the Inverse Park and Inverse Clarke transformations are used to obtain the actions used to control the IFX007 switches. The angle used in the Inverse Park transformation is also from the integrator. The implementation in pysimCoder is depicted in Figure 12. In this figure, the ADC outputs are also passed to subtract and gain blocks, mapping an ADC number to the real currents using linear regression. The calibration procedure was done beforehand by fitting measured data using a custom Python script.

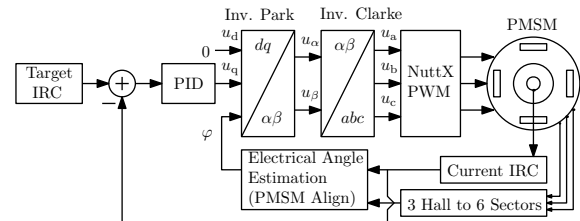


Figure 14: A simplified diagram of the closed-loop control.

The output of the reference $i_{d,ref}$ and measured i_d and i_q currents is sent by using a NuttX UDP Socket to a given IP address, which can be configured in the block itself. Using the Silicon Heaven protocol, the user can tune the constants of the PI controllers remotely while watching the changing waveforms. Figure 13 presents the captured data visualisation.

5.1.2 Simple Closed-Loop Control

A simplified diagram of the closed-loop control is shown in Figure 14. The difference between the target and the current incremental encoder count (NuttX-specific block) goes into the PID controller, setting the q -axis action for the highest torque while the d input is set to zero. The inverse Park transformation requires the rotor angle to be estimated. This is performed by the electrical angle estimation block using current IRC count and Hall sensors.

The realisation of this control diagram in pysimCoder is shown in Figure 15. The *Hall To 6 Sectors* block reads the GPIOs (NuttX-specific) of the Hall outputs and transforms them into the 6 sectors (a simple integer). The *Encoder* block outputs the current encoder count (C) the encoder count of the last hit index (C_1) and the number of hit indexes. The angle estimation is performed by the *PMSM Align* block. When the number of hit indexes is zero, the estimation is done only using Hall sensors, and the block outputs multiples of $\pi/3$. When a first index is hit,

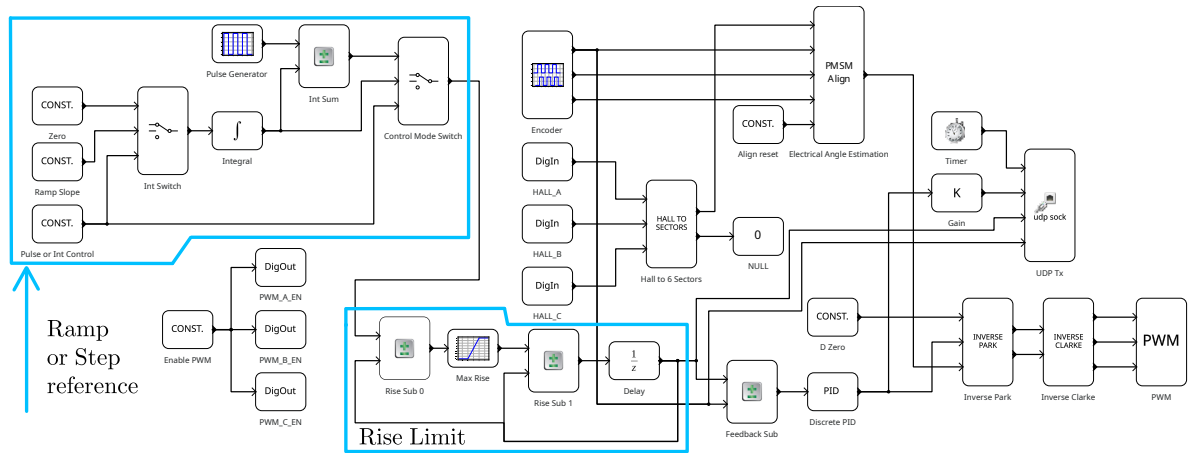


Figure 15: PysimCoder of the closed-loop control diagram with a selectable reference.

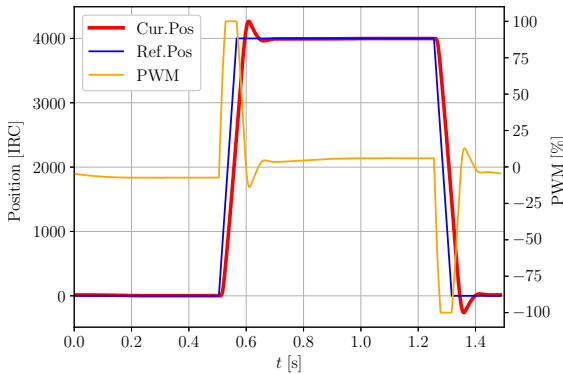


Figure 16: The step response to 4000 encoder counts.

the following formula is used:

$$\varphi_{Est} = \frac{C - C_1}{C_{Turn}} 2\pi + \varphi_{Offset}, \quad (6)$$

where C_{Turn} denotes the number of encoder counts per one electrical turn and φ_{Offset} denotes an offset between the start of electrical angle and index. Values C_{Turn} and C_{Offset} must be set in the *PMSM Align* block.

The diagram in Figure 15 also features a selectable reference between a ramp and a step. The reference is chosen by setting the constant remotely using Silicon Heaven (Lenc et al., 2023).

Furthermore, the rise time of the ramp is limited with a saturated sumator. Again, this diagram features a NuttX UDP Socket sending the PID action, the reference, and the actual position. Figure 16 shows a step response of the used motor.

5.2 Control of a Piezoelectric Actuator

This section presents a simple pysimCoder diagram capable of controlling a piezo actuator P-871.140, (PI,

Table 3: Parameters: PL140 - PIC251

Symbol	Description	Value
l_t	length	40 mm
w	width	11 mm
d	thickness	0.55 mm
c	capacitance	$8.2 \mu\text{F}$
u_{rated}	rated voltage range	$\pm 30 \text{ V}$
z_f	free deflection at u_{rated}	$\pm 1 \text{ mm}$
f_b	blocking force at u_{rated}	$\pm 0.5 \text{ N}$
f_n	first natural frequency	160 Hz

2008). The parameters of this piezo bender actuator are listed in Table 3, where the used piezo material is PIC251 ceramic. The purpose is to demonstrate simple control applications that can be quickly developed in research environments. Despite this actuator having a tensometer, an optical approach has been chosen as proof of the movement. There is a mirror placed at the tip of the actuator. If this place is illuminated by a laser beam, the incidence angle changes, thus the reflection angle changes too. The experiment setup is shown in Figure 17.

The piezo actuator is controlled by a linear voltage. In our case, we have constructed an LC low-pass filter, turning the PWM voltage into a DC voltage which depends on the duty. Although the power board supplies 24 V at most, this voltage was used for initial tests, despite the rated voltage range of the piezo actuator. The pysimCoder diagram is shown in Figure 18. The only block that is to be controlled over Silicon Heaven is the *CONST* block with the *Duty* description in the range of [-1, 1].

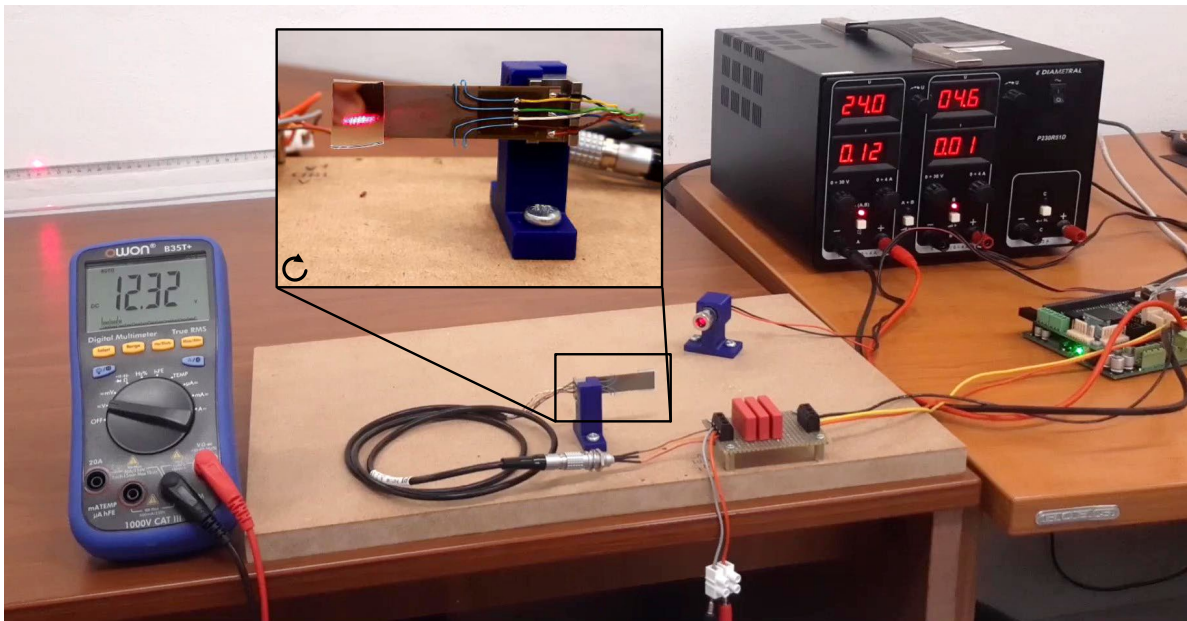


Figure 17: Experimental unit with piezo bender actuator P-871.140.

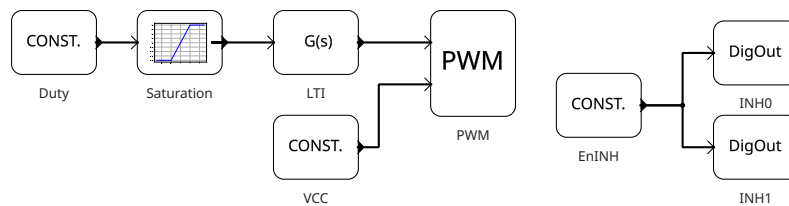


Figure 18: Inputs and outputs of piezo bender actuator.

6 CONCLUSIONS

This paper introduces the design of an extensible prototyping unit with an emphasis on motion control of modern BLDC/PMSM drives and piezo actuators. Proposed power stage is rated up to 24 V.

The hardware was successfully tested with pycsimCoder with generated code running on NuttX real-time operating system including remote tuning model parameters using Silicon Heaven. The adaptations of NuttX peripheral drivers for the platform have been merged by NuttX maintainers into the main-line repository. We have observed limits of maximum achievable sampling frequency of the NuttX scheduler, which requires future investigation.

Despite the indicated issues, we demonstrated the advantage of open-source and block-based graphical environment pycsimCoder that control applications can be created very quickly without writing code. It was proven in the simple PID PMSM control and the piezo actuator control. Compared to fully professional commercial solutions for universal use,

e.g. dSPACE, the presented solution is feasible with components at an affordable cost and free open source software. In addition, other operating systems like Zephyr and RTEMS support the selected family of SAMV71 microcontrollers as well. Thus, they represent another alternatives for real-time control applications. In the future, the adequate selection of the operating system is important for the sampling frequency.

Hardware and pycsimCoder model diagrams are available in the public repository:

<https://gitlab.fel.cvut.cz/otrees/motion/samocon>

ACKNOWLEDGEMENTS

This work was supported by The Czech Academy of Sciences, Institute of Information Theory and Automation under project No. 23-04676J of the Czech Science Foundation: Controllable Gripping Mechanics: Modelling, Control and Experiments.

REFERENCES

- Belda, K. and Píša, P. (2021). Explicit model predictive control of PMSM drives. In *2021 IEEE 30th Int. Symposium on Indus. Electron. (ISIE)*, pages 1–6.
- Bucher, R. (2024). pysimCoder – Block diagram editor and real time code generator for Python. <https://github.com/robertobucher/pysimCoder>. Accessed: 2024-07-10.
- dSPACE GmbH (2024). dSPACE rapid prototyping systems. Accessed: 2024-09-22.
- Gao, X., Yang, J., Wu, J., Xin, X., Li, Z., Yuan, X., Shen, X., and Dong, S. (2020). Piezoelectric actuators and motors: materials, designs, and applications. *Advanced Materials Technologies*, 5(1):1900716.
- Group, A. (2024). POSIX IEEE std 1003.1-2024. <https://posix.opengroup.org/>. Accessed: 2024-07-10.
- Lenc, M., Píša, P., and Bucher, R. (2023). pysimcoder – open-source rapid control prototyping for GNU/Linux and NuttX. In *2023 24th Int. Conf. Process Control*, pages 102–107.
- ASF, Apache Software Foundation. (2023). Apache NuttX, rtos. <https://nuttx.apache.org/>. Accessed: 2024-07-10.
- ASF, Apache Software Foundation. (2023). NuttX task trace. <https://nuttx.apache.org/docs/latest/guides/tasktrace.html>. Accessed: 2023-11-18.
- ASF, Apache Software Foundation. (2024a). NuttX documentation. <https://nuttx.apache.org/docs/latest/>. Accessed: 2024-07-10.
- ASF, Apache Software Foundation. (2024b). NuttX documentation - system time and clock. https://nuttx.apache.org/docs/latest/reference/os/time_clock.html. Accessed: 2024-07-10.
- ASF, Apache Software Foundation. (2024c). Repository of NuttX source code. <https://github.com/apache/nuttx>. Accessed: 2024-07-10.
- Elektroline (2024). The repository of Silicon Heaven Protocol source code. <https://github.com/silicon-heaven>. Accessed: 2024-07-10.
- Infineon, Technologies. (2018). High current pn half bridge with integrated driver. https://www.infineon.com/dgdl/Infineon-IFX007T-DS-v01_00-EN.pdf?fileId=5546d46265f064ff0166433484070b75. Accessed: 2023-09-10.
- Microchip, Technology. (2019). 10BASE-T/100BASE-TX PHY with RMI support. https://ww1.microchip.com/downloads_note. Accessed: 2023-10-10.
- Microchip, Technology. (2023). SAM e70/s70/v70/v71 family data sheet. <https://ww1.microchip.com/downloads/aemDocuments/documents/MCU32/ProductDocuments/DataSheets/SAM-E70-S70-V70-V71-Family-Data-Sheet-DS60001527.pdf>. Accessed: 2023-10-10.
- PI, Physik Instrumente, GmbH. (2008). P-871 PICMA® piezo bender actuators. https://www.pi-usa.us/fileadmin/user_upload/pi_us/files/product_datasheets/P871_Piezo_Bimorph_Bender.pdf. Accessed: 2024-02-08.
- Moon, J.-J., Lee, W., Park, S.-W., and Kim, J.-M. (2015). Fault tolerant control method of seven-phase bldc motor in asymmetric fault condition due to open phase. In *2015 9th International Conference on Power Electronics and ECCE Asia (ICPE-ECCE Asia)*, pages 1591–1596.
- ODrive Robotics (2024). The ODrive website. <https://odriverobotics.com/>. Accessed: 2024-09-20.
- PiKRON (2014). Lx_RoCon – Robot Motion Controller. https://pikron.com/pages/products/motion_control/lx_rocon.html. Accessed: 2024-07-10.
- SOLO Motor Controllers SRL (2024). The SOLO controllers website. <https://www.solomotorcontrollers.com/>. Accessed: 2024-09-20.
- Speedgoat GmbH (2024). Speedgoat rapid control prototyping. <https://www.speedgoat.com/solutions/testing-workflows/rapid-control-prototyping>. Accessed: 2024-09-22”.
- Tachiquingutierrez, R. T., Lay-Ekuakille, A., Chiffi, C., Singh, S. P., and Rao, K. S. (2023). Design, fabrication, and testing of a microelectronic controller for sensing and actuating in robotic neurorehabilitation. *IEEE Sensors Journal*, 23(16):18700–18707.
- Torvalds, L. (2024). Linux kernel. <https://kernel.org/>. Accessed: 2024-07-10.
- Vedder, B. (2022). The VESC project website. <https://vesc-project.com/>. Accessed: 2024-09-20.