Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Control Engineering

Master's thesis

Control System for a Dual-stage High-precision Electromechanical Motion Platform

Bc. Adam Polák

Supervisor: doc. Ing. Zdeněk Hurák, Ph.D

24th May 2019

Acknowledgements

First I would like to thank my supervisor Zdeněk Hurák for giving me the opportunity of working on this project and inviting me to his research group AA4CC. Without his guiding and support, I would not be able to progress in this thesis to the final point, since I would probably get stuck on the first problem and procrastinate forever. I would also like to mention my colleagues and friends from the AA4CC for a friendly atmosphere and their willingness to help in every situation.

Last but by no means least, I would like to thank my parents for a lifelong support. Without them, I would literally not be able to do anything at all.

Declaration

In Prague on 24th May 2019 . . . .

Citation of this thesis

Polák, Adam. *Control System for a Dual-stage High-precision Electromechanical Motion Platform.* Master's thesis. Czech Technical University in Prague, Faculty of Electrical Engineering, 2019.

Abstrakt

Tato diplomová práce popisuje vývoj řídicího systému pro dvoustupňovou platformu přesného polohování, která je součástí průmyslového osazovacího robota pro optické součástky. Úloha obsahuje návrh budiče pro lineární DC motor a jeho následné použití v řídicím systému pro dvoustupňovou polohovací platformu a jako případná náhrada za současný komereční budič. Následující část práce spočívá v návrhu řídicí strategie pro úlohu měkkého dopadu za účelem zrychlení oproti konzervativnímu průmyslovému postupu. K tomu je přistoupeno vývojem hybridního modelu dvoustupňové platformy a jeho použití v hybridním modelovém prediktivním řízení v simulaci, na základě které je navrhnuta suboptimální strategie modelového prediktivního řízení s heuristikou a následně úspěšně použita na dvoustupňové platformě. Řídicí program je realizován na za použití platformy dSpace MicroLabBox.

Klíčová slova dvoustupňová platforma, hybridní systém, hybridní MPC, MPC, voice coil

Abstract

This thesis describes the development of a control system for a high precision dual-stage platform, which is a part of an industrial assembly robot of optical components. The task consists of a driver electronics design for a voice coil motor and its subsequent use as a part of control system for the dual stage and possible future replacement of the curret industrial driver. A consequential part of the thesis resides in designing the control strategy for the task of soft landing as an improvement over the conservative industrial approach. This is achieved by developing a hybrid model of the dual stage system used in a hybrid model predictive controller in simulation, based on which a consequent suboptimal model predictive control strategy with heuristics is designed and successfully applied on the dual-stage system. The control program is realized on the dSpace MicroLabBox platform.

Keywords dual-stage, hybrid System, hybrid MPC, MPC, voice coil motor

Contents

Introduction

This thesis describes a design of a control system for a dual positioning stage, based on an assembly robot for optical sensors in the company *EZconn Czech a.s.* in Trutnov. A collaboration project between the company and the Czech technical university in Prague is currently running and this work is a part of it. The dual-stage experimental setup is a physical model of a part of the original robot. In the production, the assembly robot is moving components from one wafer – the source, and solders them on another wafer, the target. The goal of my work was to get the dual-stage into the state in which it would be possible to conduct experiments with control design concerning the task of soft landing – getting the end effector in contact with the wafer and prevent any potential damage by keeping the impact force under $F_{\mathrm{imp}} = 0.3\,\mathrm{N}$, which is a requirement given by EZconn. After that, the goal is to design a control strategy that would perform better than the conservative industrial approach, in which the individual stages move sequentially and very slowly in order not to cross the constraints.

The structure of this thesis can be divided into two parts. The first part is a design of a control hardware and software for a linear voice coil (VC) motor (chapter 3) and creating an interface for an industrial permanent magnet synchronous motor (PMSM) with a driver, so that everything can be controlled from of the dSpace MicroLabBox platform (chapter 2). The second part (chapter 4) consists of developing a model and designing a control algorithm for the dual-stage as a whole. Afterwards, experiments are conducted both as a simulation and with the physical model and the designed algorithm is evaluated.

Chapter **1**

System Structure

In this chapter, the overall structure of the system will be described in a form of basic information about its individual components and their interconnection, together with a commentary on the state of the project at the beginning, i.e. the hardware and software available to start with, and overall picture of what had to be done in order to get the project into the present state.

1.1 dSpace MicroLabBox

The MicroLabBox by dSpace is a platform for rapid prototyping with many functionalities and interfaces. The computing power is based on the NXP QorlQ P5020 processor for a real-time computation and NXP QorlQ P1011 for a communication with a host PC. In terms of connectivity it has Ethernet, CAN, Serial and LVDS interfaces, several digital and analog I/O channels. It supports the connection with Matlab®/Simulink via the Real-Time Interface (RTI), programming in C using the real-time library RTlib, and the real-time applications can be accessed and operated through the software ControlDesk. Programmable FPGAs are available for fast computations as well.

Following features are used in this project:

- RTI, RTlib and ControlDesk software
- Ethernet I/O with the RTlib C library
- PWM generation with 10 ns resolution
- Analog I/O – either 14-bit channels with 10 Msps or 16-bit channels with 1 Msps, both differential with range of ±10 V

Figure 1.1: dSpace MicroLabBox[1]

[1]https://www.dspace.com/en/inc/home/products/hw/microlabbox.cfm

- Digital I/O with incremental encoder sensor input interface
- Sensor supply 2-20 V

The standard workflow with the dSpace platform has been to design a control scheme within the Matlab®/Simulink framework and then use the Matlab®/Simulink Coder to generate the C/C++ code, compile it and send it to the platform, where it is saved in the flash memory and can be run independently on the connection with Matlab®. The External Mode for the direct interaction with the generated real-time application through Simulink® is not used, as the dSpace PC software ControlDesk is made for this purpose. The ControlDesk offers a variety of options for the MicroLabBox which can be adjusted online. Among the general settings and operational features, it can access the the real-time application variables through the variable description (format `.sda`), which is generated together with the code. The accessed variables can be logged, plotted in a graph or have their value adjusted. Since there is an extra processor for the communication with the host PC, everything mentioned can done very smoothly when compared to the Simulink® External Mode connection.

On top of that, there is a wider selection of the online-tunable parameters compared to the External Mode, such as controller constants. This makes controller hand-tuning or a model parameter hand-fitting much faster process, since one can observe the changes in dynamical behaviour immediately, as opposed to the necessity of building and loading the application again with the External Mode because of some untunable-parameter change, or any change in case of no External Mode.

1.2 Actuators and Drivers

Two motors are being used in this project. The first one is a permanent magnet synchronous motor (later as PMSM), or an AC servo brushles motor in the manufacturers terminology. The motor is made by a German company ESR Pollmeier GmbH and a driver from the same manufacturer is being used in this project and the assembly robot as well. There is another branch in this project, where the possibility of developing a new driver as a replacement for this one is being researched, but currently it is not in a phase when it would be applicable for my experimental setup, therefore I use the driver and the

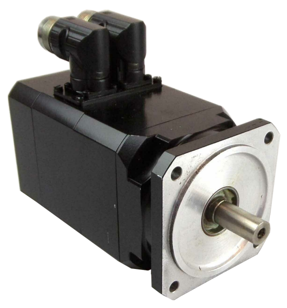


Figure 1.2: ESR AC servo brushless motor[2]

[2]https://www.esr-pollmeier.de/

motor as they do in the company. EZconn provided
us with a linear stage, which is driven by the motor. More about these components is written in the chapter 2.

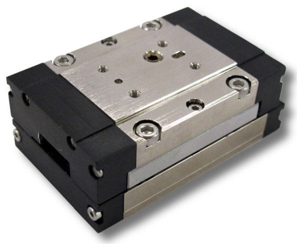


Figure 1.3: SLA25 linear slide actuator[3]

The second motor is a voice coil linear slide actuator (later referenced as VC) by the U.S. company SMAC. The manufacturer sells a driver as well, but for reasons explained in the chapter 3, it had been decided to develop our own driver, which I did as a part of this thesis and the plan is to use it with the VC motor not only in laboratory experiments but it should be provided as a prototype for the company as well. By no means is it intended as a complete product for deployment in a production, since it has no certification whatsoever, it is considered merely as a prototype on which the company could build the possible final product, which would replace the commercial driver. Since the company does not have access to a dSpace platform, there is a need to provide a solution without it. For this purpose, another prototyping platform has been chosen – Launchpad board by Texas Instruments, which is briefly described in the next section.

1.3 TI Launchpad

Compared to dSpace MicroLabBox platform, LAUNCHXL-F28379D by Texas Instruments is a budget development tool for a fraction of a cost of Microlabox. It is based on the processor from C2000 processor family with support from Matlab®/Simulink for the code generation. Together with the orientation on motor control prototyping, this has been the reason for picking this platform as a base for the second variant of VC driver.

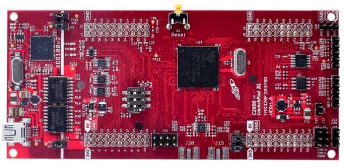


Figure 1.4: TI Launchpad C2000 Delfino MCU F28379D[4]

The important features of LAUNCHXL-F28379D for this project are:

- Embedded Coder Support Package for TI C2000 Processors in Matlab®/Simulink with libraries for the specific processor
- Analog I/O –16 bit 1.1 Msps differential ADCs or 12-bit 3.5 Msps singe-ended, range $0 - 3.3\,\mathrm{V}$
- Digital I/O, 3.3 V logic level

[3]https://www.smac-mca.com/

[4]http://www.ti.com/tool/LAUNCHXL-F28379D

- PWM generation modules
- Quardrature incremental encoder interface modules
- Serial communication interface, CAN module

1.4 Experimental Setup

The whole experimental setup can be seen on the figure 1.5. The main control program runs on the dSpace MicroLabBox platform, which communicates with the ESR driver through Modbus I/O. It sends the control word, mode set, velocity (or possibly a position or torque) setpoint and recieves the data from the sensors – position, velocity and torque. The ESR driver recieves the data from two sensors – resolver and a linear encoder. It uses the resolver as a sensor for the the speed feedback loop and the linear encoder as a sensor in the position control. The motor is driven by three phases U, V and W. The rotational motion of the motor is translated to the motion of the linear stage by a leadscrew. This stage is originally made for horizontal motion, but very similar mechanism is used for the vertical motion on the assembly robot, therefore it does not play any role.

The dSpace sends a pulse width modulation (PWM), direction (DIR) and disable (DIS) signals to the VC driver. The PWM signal drives the VC motor by setting the average voltage through the duty cycle of the switching. In reality, the voltage fluctuates between zero and the source voltage – this is called a power PWM (Figure 3.5), but the current stays, with some ripple – as discussed in Chapter 3, around the value corresponding to the average voltage value over the Ohm's law. The DIR signal assigns the direction of the current and the DIS signal disables the driver when HIGH. Since the current feedback loop runs on the dSpace, it receives the data from the current sensors on the VC driver as analog signals, as well as the pulses from the linear incremental encoder on the VC motor. Finally, the dSpace recieves a measurement from a force sensor, which is later used for verification of the soft landing task.

The diagram on the figure 1.6 describes the second setup with the Launchpad and VC driver in BoosterPack variant, which will be later handed to EZconn. Here, the PC serves not only as measurement visualizer and operation command transmitter, but as a main control station with the control program runs, instead of the dSpace platform in the experimental setup. The control program which would run on the PC, as it does in EZconn, is not a part of my thesis. In my experiments, I have tested the Launchpad board with my driver design using the External Mode in Simulink®.

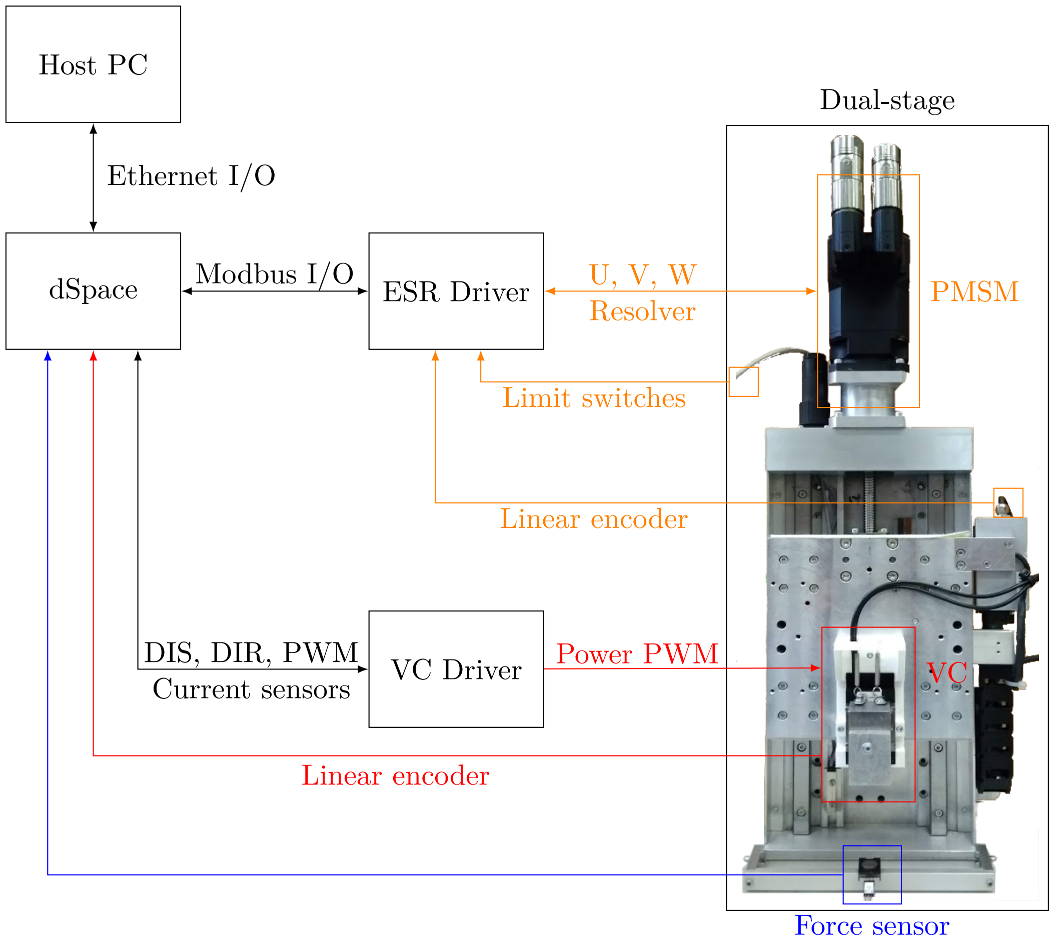


Figure 1.5: Experimental setup

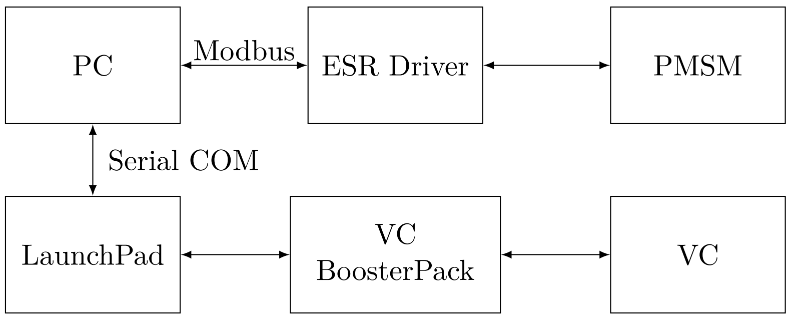


Figure 1.6: Launchpad setup

Chapter 2

PMSM Stage

As mentioned earlier, the motor and the driver from the ESR company together with the linear stage from EZconn are being used "as is" with no electronic or machinery modifications, apart from the 3D printed mounting modul for the voice coil that I had to design in order to hang the moving part of the voice coil motor on the two springs to mimic the situation on the assembly robot. This chapter will thus serve as a brief description of the motor and driver characteristics, control settings and the communication interface between the driver and the MicroLabBox platform.

2.1 Linear Stage

The linear stage is depicted on the previous schematic overview of the whole experimental setup on the figure 1.5, where it is clearly visible what it comprises of. Using a leadscrew, the rotational motion is transformed to the translational one, during which is the moving part of the stage supported by two linear guides. A linear optical scale is placed alongside the guides, which is sensed by the optical encoder on the moving part, providing the position measurement in this axis. In order for the stage not to crush into the margins, there are two adjustable limit switches, which switch off the control when activated. There is also one identical switch between them marking the index position for the linear encoder, used during the homing procedure at the startup in order to synchronize the position and get rid of the possible unwanted offset – potentially a cause of a catastrophic scenario for the force sensor placed at the bottom of the stage in the soft landing task, as it could crash into it in full speed and inflict its destruction due to the overforce.

Concerning other possible failures, in case when the limit swich is for some reason not functional and the stage crashes to the margin, the plastic shaft coupler providing the connection between the motor shaft and the leadscrew could break when the motor applies sufficient torque.

2.1.1 Technical Parameters of Stage Parts

Several application-important parameters of the individual parts on the stage are worth mentioning:

- Force sensor Honelywell FSAGPDXX001RCAB5:
  - Nominal Force: 1 Kg or 9.81 N
  - Overforce: 6.804 Kg or 66.72 N
  - Error: ±5 % of the output range 0-5 V
- Linear encoder Heidenhain 12R:
  - Resolution: 1 µm

2.2 ESR Driver

The exact label of the driver is Servo Drive New Generation, Size 1, with marking `BN6774.6671-B1-RA-A2-F8-ZL1-K1`, in which among other less important information the main parameters are decoded:

- 74 in 6774 describes that the power input is the mains connection with 230 V 50/60 Hz and the output RMS current is up to 6 A.
- B1 marks that there is a cascade controller with torque, speed and position control modes
- RA means the type of the encoder interface. There are more options for this code word, but the particular version in the experimental setup operates with a motor containing resolver
- F8 stands for the communication interface – in our case an ethernet communication as a physical and datalink layer of the OSI model, above which is the Modbus interface at the application layer, together with a custom ESR interface used for communication with the command and commissioning software SPP Windows provided by ESR. Using this software, it is possible to configure the adjustable parameters and settings of the driver.
- ZL1 marks that there is an additional incremental encoder for 5 V signals over RS422 present. In the experimental setup, this is the Heidenhein 12R linear encoder for measurement of the stage motion.

Throughout the text, I refer to the driver as ESR driver for simplification.

2.3 dSpace Interface

The interface with the ESR driver is realized through a Modbus protocol, since there is no other usable possibility for communication with this driver version, apart from the aforementioned ESR protocol, which is not standardized and poorly documented. The Modbus communication is realized through the TCP/IP protocol over the standard port 502.

On the side of dSpace MicroLabBox, we have written a C/C++ S-function[5] for Modbus communication together with my colleague Lukáš Černý. The S-function is than wrapped by a simulink block, displayed on the figure 2.1.

Modbus protocol communication is based on the client-server scheme using transactions, which are either a request from the client or a response from the server. There are two types of messages for the communication with the driver – service data object (SDO) and process data object (PDO). SDOs transmit parameters and settings and are used for non-frequent parameter changes, while the PDOs contain the process data transmitted frequently with constant period, such as references (position, velocity, torque), control word, operating mode in the request message and sensory readings, status word etc. in the response message.

The structure of the PDO request message is depicted on the figure 2.2. First, there is the header with a transaction identifier (TI), protocol identifier (PI), length of PDO and unit identifier (UI). The TI increments with every message, while the PI and UI are set to zero in this case, since there are no other units present or protocols used in the experimental setup. The PDO length differs with the type of the message – for practical reasons, we have divided the PDO messages on two types. The first type is the normal routine, in which we write the references for position, velocity and torque with requests for reading of actual position, velocity, torque, status word and mode. The second type is send only with a change of the control word or operation mode. This has been necessary because after rewriting of the control word or operation mode register, the ESR driver responds with an error packet. For this reason, it is necessary to restart the communication by setting the input of the the CommEnable port of the Simulink®block to zero and then again to one to initiate the communication again. After the Header, the PDO request

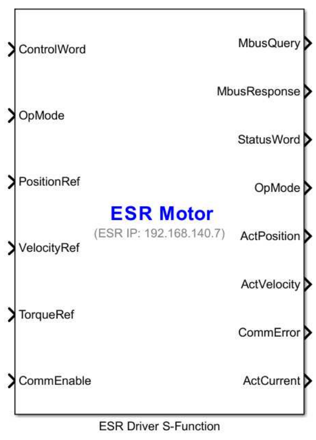


Figure 2.1: Simulink® block using S-function for MicroLabBox/ESR driver Modbus communication

[5]https://www.mathworks.com/help/simulink/s-function-concepts-c.html

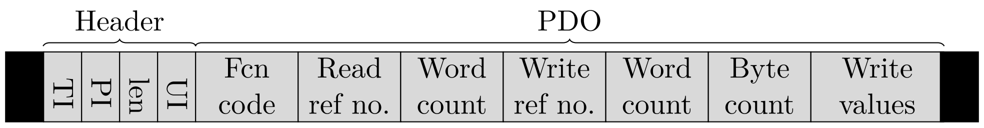


Figure 2.2: Structure of a process data object (PDO) request packet

message itself follows by the Function code, which defines if we are going to write to registers, read them, or both. Then there are reference numbers, which contain the indexes of the registers we would like to start reading from or writing to. After that the Word/Byte counts – the size of the actual data we would like to read or write. In case of write operation, there is the actual data at the end, in sequential order starting from the register with the reference number.

The server response packet is very similar to the request packet in structure, the header is the same and the PDO part contains the registers which were requested to be read. The Simulink® block is intended just for PDOs, as it would make no sense to include the SDOs Simulink® signal diagram. The parameters in the driver can be changed using the SPP Windows program instead.

The sampling period of the is set to 5 ms, and so the frequency of Modbus communication and therefore the reference sending and measurements receiving is 200 Hz. It can be set higher, but then the buffer can get full and the communication quality gets worse. Additional information for usage of the block is included in the block description.

2.4 Modeling and Control

For modeling purposes, the PMSM stage is represented in a model in section 4.4.2 just as a velocity source. This simplification is possible because we use the system in closed-loop setting with the driver, which assures that the velocity reference is the actual velocity of the stage after a short transient time. The transient could be modeled for example as a first order system, but then it would not be ideal as well, because the controller includes nonlinearities. Apart from the obvious ones such as saturations, there are acceleration ramps, which serve as smoothing element for the velocity transients. Therefore I decided to skip any modeling of the transient behaviour and see how the model performs without it first, which it did reasonably well.

Chapter 3

Voice Coil Stage

In this chapter, the upper linear actuator is discussed from the control point of view. Both the drive electronics design and the control algorithm are described here, together with the discussion about the particular challenges that had to be overcome and the design decisions I have made.

3.1 Voice Coil Motor

The upper stage of the dual-stage setting consists of the voice coil linear slide actuator SLA10-010-55-1[6] by the company SMAC. The parameters of this motor are mentioned in the table 3.1. It is worth noting that neither the resistance nor the inductance are present in the datasheet, probably because the strategy of the manufacturer is to mention the least information possible, so that the customer decides to buy their driver rather than designing and producing it on his/her own. Since these two parameters offer some insight usable for the the current feedback control design, I had to acquire them by measurement. The process of measuring these parameters is described in the following subsection.

3.1.1 Measurement of Parameters

I measured the resistance using a digital multimeter under standard laboratory conditions. Since the resistance is relatively high, the heat dissipation is rather significant and the resulting increased temperature of the winding causes the resistance to grow. This varying parameter problem is solved by using an integral term in the current feedback loop and does not influence the control, unless the maximum current is applied to the motor constantly for a time period of approximately 30 s, after which the control action saturates.

[6]https://www.smac-mca.com/sla25-series-linear-slide-actuator-10mm-stroke-volt-single-phase-micron-encder-double-p-200002.html

Table 3.1: Parameters of the SLA10-010-55-1

Parameter	Marking	Unit	Value
Voltage	V_s	V	24
Stroke	l_{vc}	mm	10
Peak Force	F_{peak}	N	4
Continuous Force	F_{cont}	N	1.6
Force Constant	k_f	NA^{-1}	2.7
Max. Current	$I_{\max}$	A	1.5
Moving Mass	m_{vc}	Kg	0.05
Total Mass	m_{tot}	Kg	0.16
Load Mass$^+$	m_{load}	Kg	0.390
Inductance*	L	H	**
Resistance*	R	Ω	18

*measured, missing in the datasheet
**figure 3.1
$^+$load installed on the motor

As for the inductance, it is generally frequency dependent, therefore it depends on the PWM switching frequency of the driver, which is fixed, and so just one inductance value can be picked and used for modeling of the system. In my case, the PWM frequency is 250 kHz. I have measured the inductance using LRC meter SR720 by Stanford Research Systems[7] and obtained the data on the figure 3.1. The LRC meter has an available frequency range up to 100 kHz and other methods were not too precise, therefore I decided to use the 100 kHz value for modeling.

3.1.2 Physics of Linear DC Motor

The magnetic force is defined as follows:

$$\vec{F_m} = q\vec{v} \times B, \tag{3.1}$$

where q is an electric charge of a point, $\vec{v}$ is its velocity and $\vec{B}$ is the magnetic field. For a current-carrying conductor in a magnetic field as displayed on the figure 3.2, the force can be written in a following form:

$$\vec{F_m} = i\vec{l} \times \vec{B} = i(-l\hat{y}) \times (-B\hat{z}) = ilB\hat{x}. \tag{3.2}$$

For a coil with N turns, the formula can be written as:

$$F_m = NilB = k_f i, \tag{3.3}$$

[7]https://www.thinksrs.com/products/sr715720.html

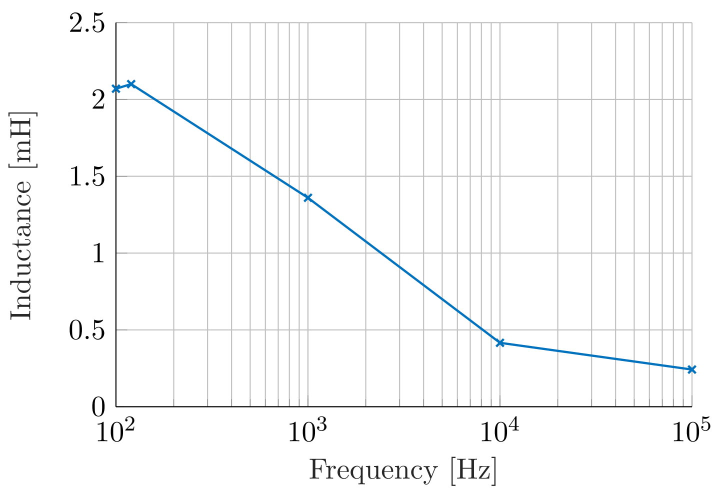


Figure 3.1: Measured inductance and frequency dependence

where $k_f = NlB$ is the force constant of the motor commonly provided in a datasheet by the manufacturer of the specific motor. To be rigorous, this constant is by no means always of a constant value, as the number of active coil turns can change with the position of the coil and the induction can depend on the amount of current going through the coil winding, i.e. $k_f(x,i) = N(x)B(i)l$. The real value should not vary too much from the constant though, therefore we usually neglect this for engineering purposes.

The magnetic flux induced by the external magnetic field $\vec{B} = -B\hat{z}$:

$$\Phi_p = \int_S \vec{B}\, d\vec{S} = \int_0^l \int_0^x -B\hat{z}\,(dxdy\hat{z}) = \int_0^l \int_0^x -B\,dxdy = -Blx. \tag{3.4}$$

In the field of engineering, flux linkage $\lambda = N\Phi_p$ is often defined for multi-turn coils. It has practically the same meaning as the magnetic flux, the distinction relies merely on the definition of the surface S over which we integrate in the equation 3.4. The electromotive force according to the Faraday's law is:

$$\xi_p = -\frac{d\lambda}{dt} = -\frac{d(-NBlx)}{dt} = NBlv = k_f v. \tag{3.5}$$

Now, the flux resulting from the current i flowing in the circuit can be written as:

$$\Phi_c = Li, \tag{3.6}$$

and the resulting self-induced electromotive force:

$$\xi_c = -\frac{d\Phi_c}{dt} = -L\frac{di}{dt}. \tag{3.7}$$

The final formula for the voltages with both EM forces:

$$Ri = u_s - \xi_p - \xi_c = u_s - k_f v - L\frac{di}{dt}. \tag{3.8}$$

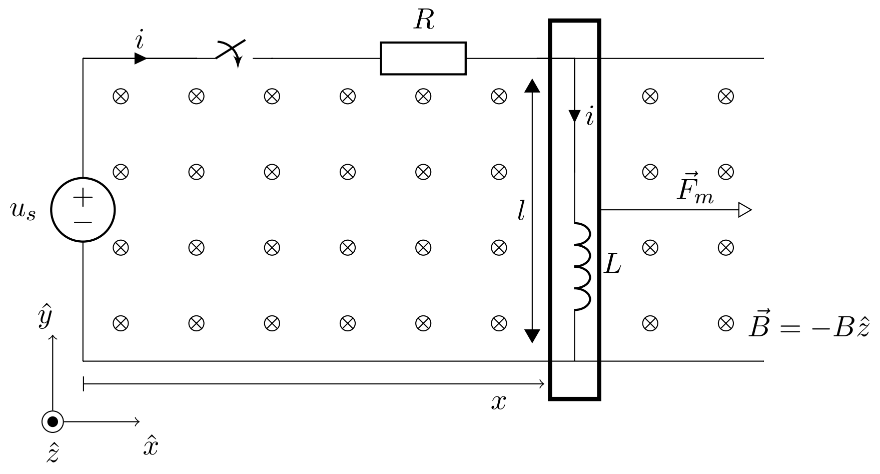


Figure 3.2: Linear DC motor[8]

The second equation needed for modeling of the Voice coil motor is the equation of motion:

$$m\frac{d^2x}{dt^2} = k_f i + b\frac{dx}{dt}. \tag{3.9}$$

We can now consider this two-equation system as state equations with the state vector $\vec{x} = (x, v, i)$ and voltage u_s as input u. The equations 3.8 and 3.9 transform then into following system of three state equations:

$$\frac{dx}{dt} = v, \tag{3.10}$$
$$\frac{dv}{dt} = \frac{k_f}{m}i - \frac{b}{m}v, \tag{3.11}$$
$$\frac{di}{dt} = \frac{1}{L}u_s - \frac{k_f}{L}v - \frac{R}{L}i. \tag{3.12}$$

This is a linear system, which can be written in a state-space form:

$$\frac{d\vec{x}}{dt} = \mathbf{A}\vec{x} + \mathbf{B}u, \tag{3.13a}$$
$$y = \mathbf{C}\vec{x} + \mathbf{D}u, \tag{3.13b}$$

with state-space matrices:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\frac{b}{m} & \frac{k_f}{m} \\ 0 & -\frac{k_f}{L} & -\frac{R}{L} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{L} \end{bmatrix}, \mathbf{C} = \mathbf{I}(3), \mathbf{D} = \mathbf{O}, \tag{3.14}$$

where $\mathbf{I}(n)$ is an identity matrix of order n and $\mathbf{O}$ is the zero matrix.

3.2 Driver Design

The available commercial controller for the voice coil motor[8] is a servo motor controller with three possible operation modes - position, velocity and torque, implemented as a cascade control with trapezoidal trajectory generator. It outputs maximum 3 A continuous current at max. 50 V, through PWM at frequency 19.531 kHz. The communication is realized over RS232 using a specific macro language. Overall, this is a standard industrial controller made to satisfy the requirements for driving most brushed DC motors within the given specifications.

A reliability of their design (or rather a conservative nature of the industry?) can be proven by the fact, that it had been introduced to the market back in 1997 and without any notable changes is being sold and purchased today as well. However, its industrial purpose as well as the age of the design, leads to caveats when a more experimental application is considered. The most limiting problem is the necessity to use the macro language, which, together with the limited bandwidth of the communication protocol and general closeness of the design, rule out any (possibly wanted) customization of the control algorithm or even outputting of some internal signal/variable, which might be needed for some control schemes.

All this has been a motivation for considering a custom design of the driver electronics with an interface for easy connection with the dSpace MicroLabBox prototyping platform, which would allow for a fully customized control design and experimentation. The following paragraphs contain the reasoning behind the design decisions during the process of developing the driver electronics.

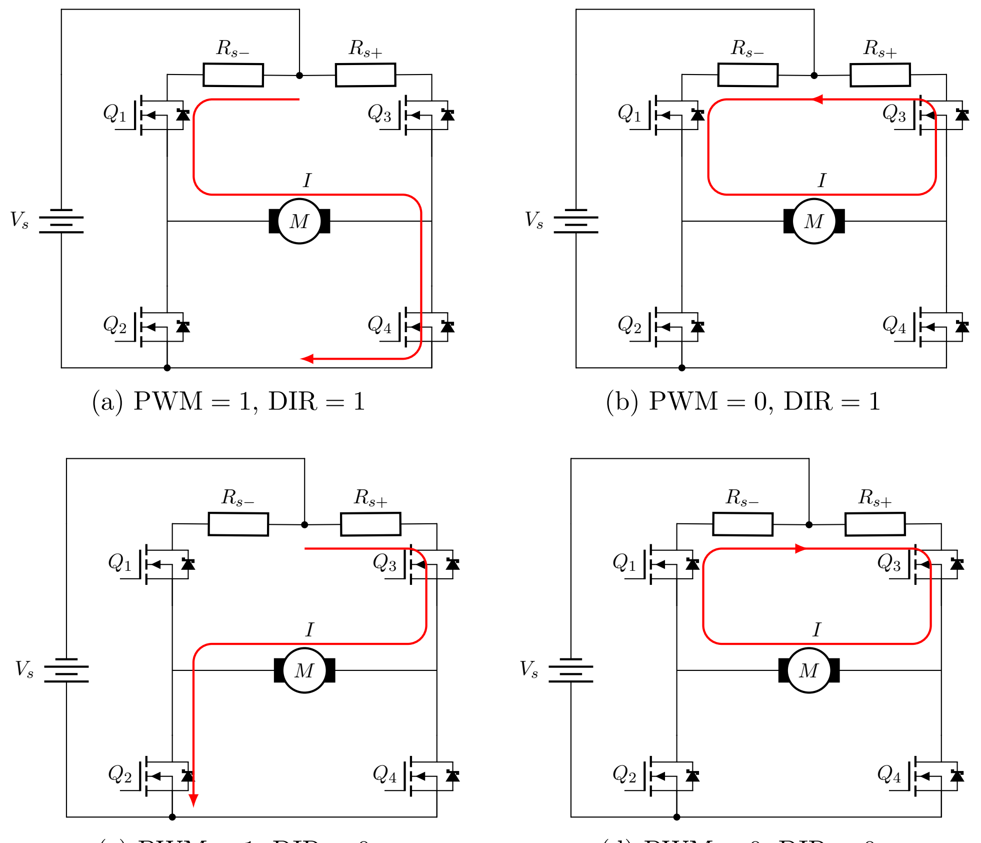
3.2.1 Design description

The most common way to drive DC motors and coils is by PWM switching. This can be done by using two transistors for one direction drive and four for bidirectional drives, connected in a way commonly referred to as H-bridge circuit. There are several possible switching scenarios. I have chosen the so called sign-magnitude drive, where the transistor Q3 is constantly turned on, Q1 is switched by the PWM signal and Q2 by its negation $\overline{\text{PWM}}$ for one direction. For the other direction, the switching scheme is changed in a axially symmetrical way along the up-down axis - now the Q1 is constantly turned on and so on.

The functionality can be divided into two phases. In the first phase, the current goes from the source through the motor to the ground, in the second phase it circulates in the upper loop of the H-bridge, that as displayed on the figure 3.3.

As switching transistors, MOSFETs are frequently used. Their decisive advantages are that they require very little current for turning on and offer

[8]SMAC LAC-1 Single axis controller, https://www.smac-mca.com/



(a) PWM = 1, DIR = 1

(b) PWM = 0, DIR = 1

(c) PWM = 1, DIR = 0

(d) PWM = 0, DIR = 0

Figure 3.3: H-bridge switching, sign-magnitude drive

very fast switching possibilities. Although the MOSFETs can be theoretically switched directly by the logical inputs, they are usually not, because of several difficulties. First, not all MOSFETs are made as logic level - that is they can be switched by 5 V signals at the gate input. But most importantly, there would be a wild occurrence of the so called shoot-through, caused by athough very short, but nevertheless non-instantaneous turn-on and turn-off times of the MOSFETs. By shoot-through, we describe a situation, when the upper and lower MOSFETs (Q1 and Q2 for example) are turned on simultaneously for a short instance of time, practically creating a quick short-circuit between the supply voltage and the ground. This results in a huge voltage spikes throughout the whole drive circuit and extensive MOSFET heating. The spikes can inflict a noisy current measurement (unless sampled in between the switching events) and therefore worsens the control performance, while the heating generally lowers the electronics lifetime. One way how to solve this would be something like phase-shifting the PWMs a little and assuring that in the selected switching scheme this situation never happens, which would mean an increased overhead at the PWM generation. A far better solution is to choose an integrated circuit which takes care of this. In my case I have chosen HIP4081A by Renesas Electronics.

3.2.2 Current Measurement

The current measurement is typically carried out by measuring voltage across the shunt resistor. Other possibilities such as hall sensors do not usually have sufficient bandwidth - around few hundred kilohertz [10]. With the chosen PWM frequency 250 kHz, the bandwidth of the current sensor should be at least ten times more, in order to fully perceive the current changes. The current sense amplifier, which does comply to this requirement is for example LT1999 by Linear Technology[9].

There are several possibilities, where to put the shunt resistor, as depicted on the figure 3.4. The R_2 option is in series with the motor, which means that during both phases (PWM = 1 and PWM = 0, 3.3a + 3.3b or 3.3c + 3.3d) the true current going through the motor is sensed. But there is a catch - the voltage changes constantly with the PWM, in our case it goes constantly from 24 V to 0 V and the voltage on the V_{IN+} and V_{IN-} sense inputs floats in the meantime. Not every current sense amplifier is able to withstand these conditions, only those labeled as bidirectional might, which is the case of LT1999. In practice, it did not perform very well, huge spikes were present in the measurement (not on the oscilloscope) and couple of times the amplifier even stopped working and had to be replaced. It is not absolutely clear whether this happened because of the R_2 connection or something else during the prototype testing, but after a design change to a different shunt resistor placement, it did not happen again.

The R_3 possibility does not require bidirectionality of the amplifier, but in the phase 3.3b it senses zero instead of the real current through the motor. The R_4 is the same situation. For this reason, I have not tested these possibilities.

To solve the problem with R_2, I decided to employ two shunt resistors and amplifiers, R_{s-} and R_{s+}. In this connection, the shunt resistors have a very stable voltage without any huge leaps. The only disadvantage is the slightly increased overhead - the need to switch between the measurements based on the direction. This can be taken care of directly in hardware by using a switch IC with the DIR as the switching signal, or use two ADCs and switch between the measurements in software.

The next important thing is to set the shut resistor resistance value and amplifier gain appropriately to use the ADC range in full extent to obtain good signal to noise ration. dSpace MicroLabBox has wide ADC operating range of -10 V to 10 V, which is actually wider than what the LT1999 is able to output, since it has limited output voltage by the supply voltage of 5 V. It outputs a differential signal shifted by 2.5 V, which yields 2.5 V for positive and negative

[9]https://www.analog.com/media/en/technical-documentation/data-sheets/1999fd.pdf

range. It is not exactly necessary to sense the negative currents in the chosen manner of measurement with two shunts, since in for each direction, voltage across the respective shunt resistor is always positive. In the future, it is possible to change the current sense amplifier for a unidirectional one, which would offer a better resolution. Anyway, in this setting I have chosen the amplifier with gain 20 and shunt resistor of 0.1 Ω. This results in maximum sensed current of 1.25 A, which is slightly below the maximum continuous current of the voice coil motor 3.1. It is not a big problem, considering that in our application, the current rarely reaches 1 A. If needed, it can be solved immediately by using a shunt resistor of slightly lower value, or better by using unidirectional current sense amplifier, but this might require a PCB design change in case of different pinout and package footprint.

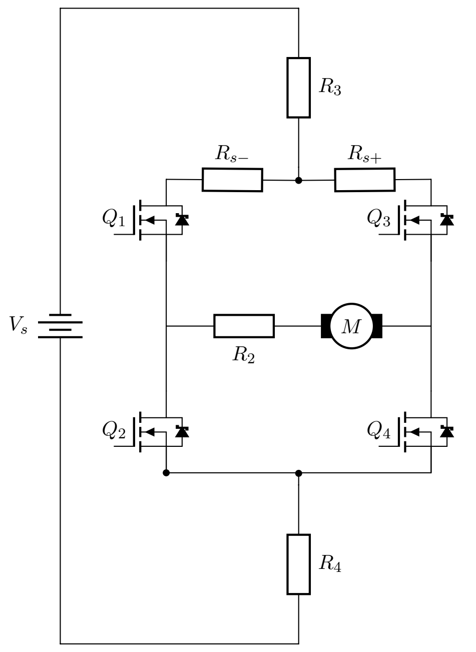


Figure 3.4: H-bridge circuit with shunt resistor placement options

3.2.3 PWM frequency

The choice of the PWM frequency directly influences several aspects of the design and resulting behaviour of the driver. On one hand, we want the frequency to be high enough to sufficiently smoothen the current going through the motor, to keep the actuating force smooth and improve the current loop control performance. To illustrate the difference what effect on the current different PWM frequencies have, on the figure 3.6 is the comparison of the resulting current ripple in cases with frequencies of 20 kHz and 250 kHz. 20 kHz

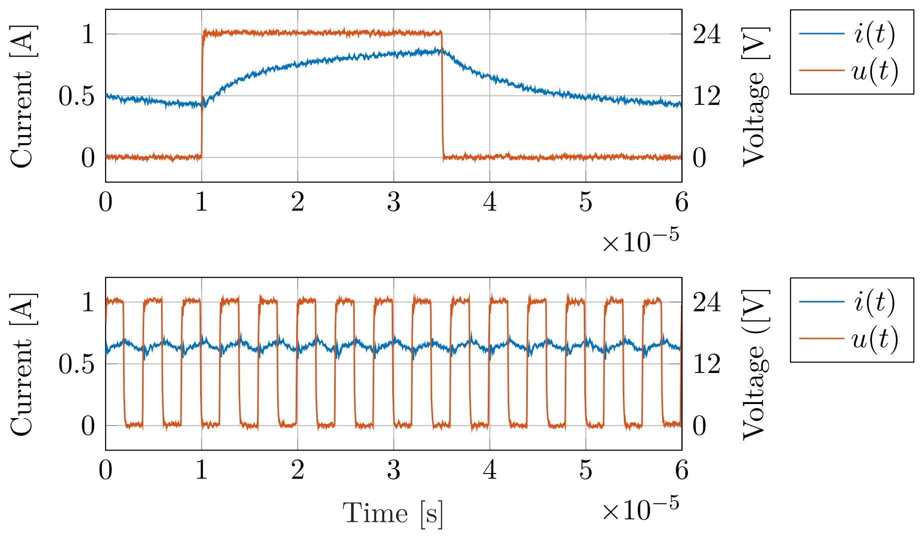


Figure 3.5: Power PWM switching and resulting current, first at $f_{\mathrm{PWM1}} = 20\,\mathrm{kHz}$ and on the second figure $f_{\mathrm{PWM2}} = 250\,\mathrm{kHz}$

is a very common frequency on which many DC motor drivers operate, since it is at the frequency limit of human ear therefore we do not hear the noise coming out of it. The figure 3.6b shows a simulation using the voltage/current transfer function of the electrical dynamics of the motor 3.17 with the measured resistance and inductance values from section 3.1.1. The simulation differs a bit with the 20 kHz from the real system, which is probably due to the measurement error of the LRC meter and the linearity of the first order model. However, for 250 kHz the simulation is very close to the real system. In the oscilloscope measurement, the current ripple is around 450 mA with the 20 kHz PWM and 100 mA with the 250 kHz PWM, not counting the switching spikes, which is a considerable improvement. It should be noted that the current does not smoothen up linearly with the rising frequency, because the inductance gets lower as well, which causes the electrical time constant to be shorter, which means faster charging/discharging.

On the other hand, higher PWM frequency means faster switching, which leads to higher heat dissipation. Other effect is connected to the limits of the switching speed of the MOSFETs and the switching circuit. Suppose we have a H-bridge comprised of transistors with turn-on time 75 ns. We want to drive a motor with a very low inductance, therefore we decided to use much higher frequency than usual in order for the current to smoothen out, say 500 kHz. Now, if we set the duty cycle to 4 %, the transistors should stay on for 80 ns, which is slightly above the minimum swiching speed, but if we set the duty cycle to 3 %, it is already below the limit and transistors would not even manage to switch at all. In our case, the deadzone would be from 0 % to 3.75 % of the duty cycle. With this setting, we would maybe

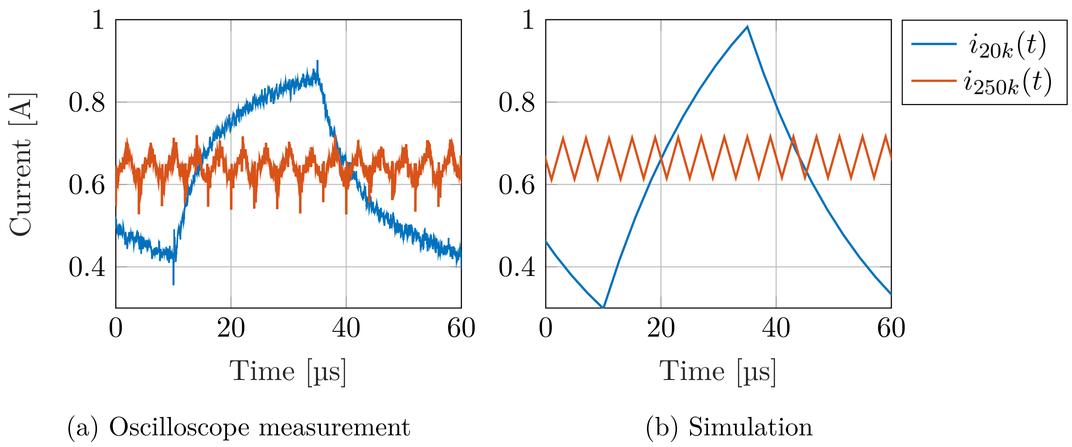


Figure 3.6: Comparison of the current ripple in the real system and simulation. $f_{\mathrm{PWM1}} = 20\,\mathrm{kHz}$ (blue), $f_{\mathrm{PWM2}} = 250\,\mathrm{kHz}$ (red), DUTY $= 50\,\%$.

improve the control in high current area, but worsen in the low current are. This would be tolerable for an application where higher currents are expected, but in a case where the current is mostly around zero, a deadzone this big is undesirable, since it would disturb the control performance. In the particular example of the voice coil motor – when the linear drive is moving horizontaly with no significant load in this direction (such as a spring), the current stays mostly around zero, so the deadzone has a considerable effect on the control. Conversely, when the motor moves verticaly then the weight is acting in this direction, therefore the current is going to be mostly nonzero, since it has to counteract it.

In the end, we want the PWM frequency not too low for the current to be undesirably sawtooth-like and not too high for the deadzone to be acceptable. In my case, I have chosen the frequency 250 kHz, with resulting deadzone of 1.88 % of duty-cycle, which results in a current around 20 mA. This is already approaching the limits of the current measurement, which makes it tolerable.

3.2.4 Driver Boards

I have produced two driver PCB boards – one with the terminal blocks for easy wire connection to the digital and analog I/O channels of MicroLabBox and a smaller one with a LaunchPad platform pin to pin compatibility, so called BoosterPack module [10]. This module can be easily mounted on the LaunchPad without any need for wiring. The dSpace version is a bit larger, because there was no need for shrinking the PCB down, as it had been with the BoosterPack version. It is a third iteration of the drive circuit, which is denoted by the letter c in the version marking. The previous versions had a

[10]http://www.ti.com/tools-software/launchpads/boosterpacks/build.html

different shunt resistor placement, which proved not to be a good solution, as already mentioned in the section 3.2.2.

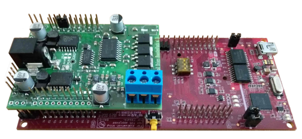
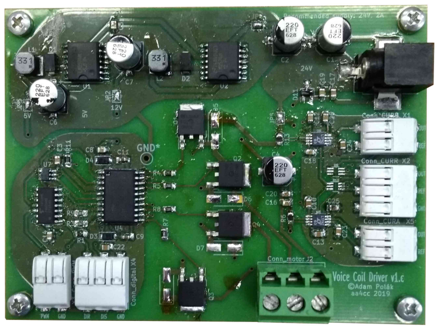
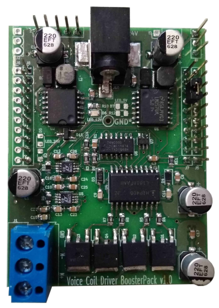


Figure 3.7: VC Driver BoosterPack mounted on a TI Launchpad

The curent version functions well enough and it performed well with the motor on the assembly robot when I used it in a current/force mode. However, a problem arose during an operation where the incremental position encoder of the voice coil motor had been used – in the position/speed feedback loop. It turned out that in EZconn they had been fixing the wires inside their voice coil motor and now the switching noise disrupts the encoder measurement. My speculation is that they might have connected the encoder ground to the motor ground, which would cause the noise to spread to the encoder wires and subsequent changes in TTL logic of the pulses during the decoding process. They did not notice this, because the SMAC driver uses differential encoder signals, which makes the decoding much more immune to noise. My testing setup included the MicroLabBox and the RTI library version of quadrature decoder, which unfortunately does not take into account the possibility of differential connection. The Launchpad platform also does not include this possibility, therefore I have to come up with a solution in the future version of the driver. It is possible to use the programmable FPGAs in the MicroLabBox, however this would not solve the problem, because it would not be of any use for the company, as they require a solution without the dSpace platform. The most realistic solution is probably to use some quadrature decoder IC with differential signal connection possibility.



(a) VC Driver dSpace version



(b) VC Driver BoosterPack

Figure 3.8: Comparison of the two driver boards

3.3 Control Algorithm

While many approaches for designing the control of a linear DC motor can be taken, for an application where the force control plays an important role, the current feedback loop is a logical choice, since the force is approximately proportional to the current going through the winding, therefore it is sometimes called a force feedback, or torque feedback in rotational system. Other control structures can be then build above it, under the condition that it is fast enough so that the controllers above it could consider the transfer function of the closed-loop system be $T_{\mathrm{cl}}(s) = 1$. I have chosen the sampling period of the current loop to be $T_s = 10^{-4}\,\mathrm{s}$. In case of the control program on the dSpace platform, this makes it the fundamental sampling period of the control program, because nothing else requires to run faster, except for the modules implemented in dSpace Real-Time Interface (RTI) library such as ADCs, PWM generation which run on the FPGAs independently of the main control program.

3.3.1 Current Feedback

Let us now analyze the electrical part of the VC motor, described by the equation 3.8. We will now omit the back EMF described by the component $\xi_p = k_f v$, because we are interested in a linear behaviour around an operating point with zero velocity, alongside the fact that the back EMF is not very significant in a linear drive with range of 1 cm, since it cannot reach high velocities for a time period longer than just a fraction of a second, therefore it results only in a negligible current ripple. The equation then turns into:

$$Ri = u_s - L\frac{di}{dt}. \tag{3.15}$$

We would like to get now the transfer function from the input voltage to the output current, therefore we apply the Laplace transform:

$$RI(s) = U_s(s) - LsI(s), \tag{3.16}$$

and the resulting transfer function is:

$$G_i(s) = \frac{I(s)}{U_s(s)} = \frac{1}{R + sL}. \tag{3.17}$$

Now we would like to design a controller for this first order system. We would like to get the closed-loop transfer function into the following form, as mentioned in [11] or [14]:

$$T_{cl}^{\mathrm{d}}(s) = \frac{\omega_b}{s + \omega_b}, \tag{3.18}$$

where ω_b is the bandwidth and $\tau_{cl} = \frac{1}{\omega_b}$ the time constant of the resulting system. The transfer function of the closed-loop system from the figure 3.9 is:

$$T_{cl}(s) = \frac{C_i(s)G_i(s)}{1 + C_i(s)G_i(s)}, \tag{3.19}$$

If we now put 3.18 and 3.20 equal and solve express the C_i, we get:

$$C_i(s) = \frac{\omega_b}{s}(sL + R) = \omega_b L + \frac{\omega_b R}{s}, \tag{3.20}$$

which is a PI controller with constants $k_p = \omega_b L$ and $k_i = \omega_b R$. This is the reason, why the PI controller is commonly used for the current control – it is very easy to implement and we can directly set the closed-loop bandwidth.

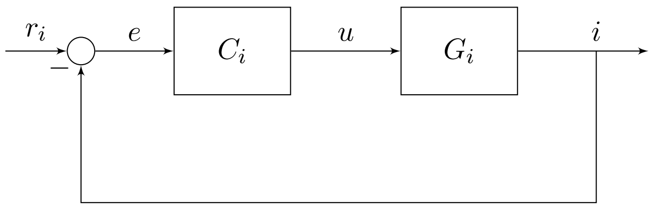


Figure 3.9: Current feedback schematic

The practical realization of the control loop consists of deadzone switch, which sends further zero when the control action is below the limit, which is in my case 2 % duty cycle. If the deadzone of the driver would not be considered in the control algorithm, the unreasonable heating of the transistors would occur, therefore it is a crucial to take care of it.

On the figure 3.9 we can see the dependence of the current loop step response on the parameter ω_b. At the value $\omega_b = 600\,\mathrm{rad\,s^{-1}}$, the measurement noise started to be excessively attenuated, which rapidly deteriorated the the response as well as steady state conditions.

3.3.2 Cascade Controller

One of the most common control configurations for driving motors is a cascade controller structure. Typically, this consists of a current, speed and a position feedback loop, as depicted on the figure 3.12, where the C_i symbolizes the current controller, G_i the electrical and G_m the mechanical of the motor, C_v the speed controller, G_v the computation of speed from the position encoder and finally C_x the position controller. As already mentioned, the necessary condition for cascade control functionality is the bandwidth/sampling period setting of the individual feedback loops, where the inner loop should be faster then the outer loop, so that the dynamics of the inner loop would not interfere with the dynamics of the outer loop. A safe rule of a thumb is to make the inner loop ten times faster than the outer loop in terms of bandwidth and

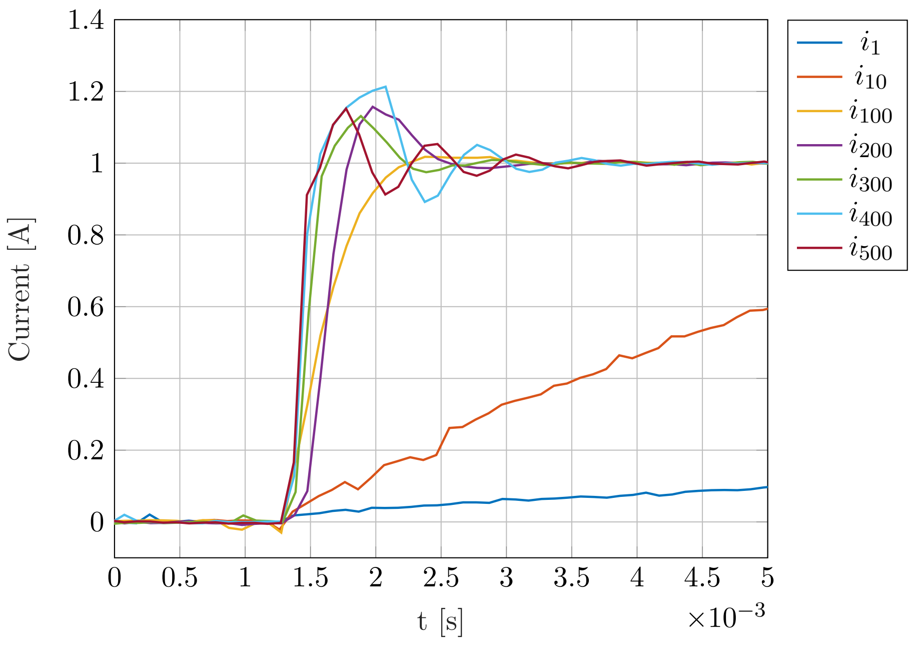


Figure 3.10: Step response of the current feedback loop and its dependence on the controller parameter ω_b, the indices in the legend denote its value

the sampling period. In my case I have set the bandwidth of the velocity and position loop to the infromation obtained from the figure 3.9, where I picked the value $\omega_b = 300\,\text{rads}^{-1}$. After tuning the speed and position PID regulators, the resulting step response is depicted on the figure 3.11.

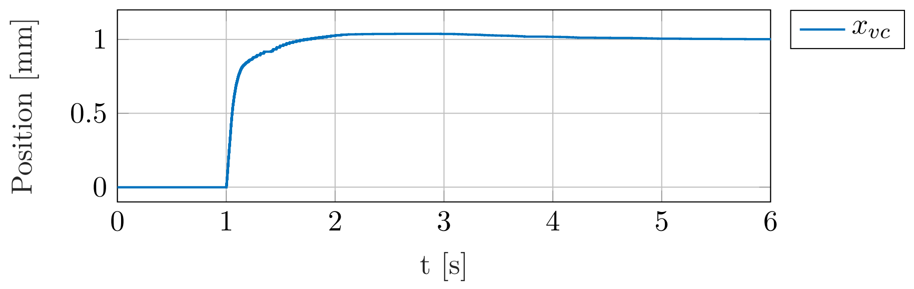


Figure 3.11: Step response of the cascade controller

Other possibility is for example to use a state feedback. This can be done either with the third order system 3.12 with states $\vec{x} = (x, v, i)$ or use the current feedback with the PI controller and consider only two order system with states $\vec{x} = (x, v)$. However, for the latter part of this thesis, just the current feedback loop is of importance, therefore I will omit adding more possibilities of control for the voice coil motor.

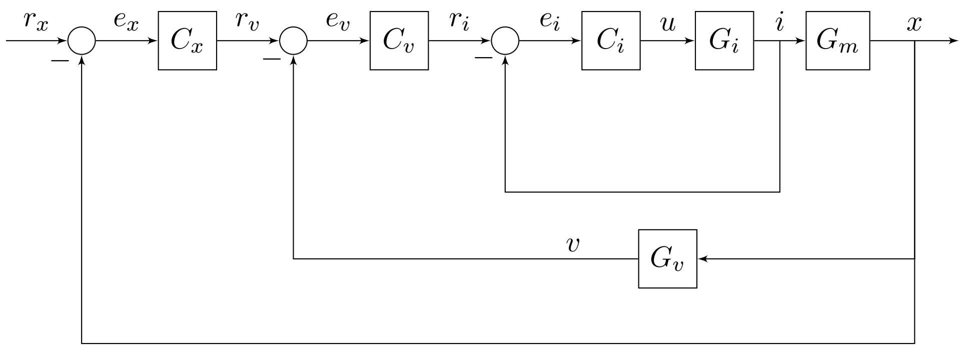


Figure 3.12: Cascade controller

Chapter 4

Dual-Stage Soft Landing

This chapter provides a brief overview of a hybrid system modeling and control, then the mathematical model of the dual-stage system dynamics of the experimental setup is derived using bond graphs. The model is then transformed into a form of discrete hybrid automata for implementation in HYSDEL (hybrid systems description language) and used for generating a hybrid model predictive controller. The simulation of the hybrid system with the hybrid controller is then described and discussed. Finally, I design a sub-optimal MPC controller with switched weights for the real system inspired by the results from the simulation and evaluate the results with respect to the existing solution in the company.

4.1 Soft Landing Problem

The experimental setup (figure 1.5) is intended mainly as an approximate physical model of z-axis motion of the assembly robot. The motion in this axis is executed (apart camera focusing) during so called soft landing task.

The term *soft landing* has been used mainly in a research area concerning valve actuators in camless engines, where the problem is to reduce the impact force between the armature and the valve seat in order to avoid excessive material wear and ensure an acceptable noise levels, while preserving the consistency and speed of the valve opening/closing. Most approaches have translated this problem into limiting the impact velocities as in [13], [16] or [13]. Other usage of the term soft landing is connected to the lunar lander, but that stands rather (literally) far from our objective.

The classical strategy in robotics consist of force control, namely hybrid position/force control as described in many robotics textbooks such as [19] or [9]. The situation is however a bit different to our setup, because I can use the available force sensor only for verification purposes (the robot in EZconn has none), whereas in industrial robots the force sensors are usually present directly behind the end effector in form of Maltese cross force sensor. After frame transformation, the measurement can be used in aforementioned force control. This is not the case of our experimental setup, which is in a way more similar to the soft landing in camless engines. On the other hand, compared to the problem in camless engines (as described in [13]), our system has much more linear behaviour, the input saturation does not play an important role and no significant disturbance is at play.

Our problem could be formulated for example as *constrained sensorless contact force control*, but the term soft landing is in my opinion much more elegant and self-explaining. My approach is to model the electromechanical system of the experimental setup using the hybrid system framework and try to apply an optimal control strategy such as model predictive control on it. The peculiarity of the experimental setup resides in an overactuated nature of the dual-stage, which offers several different possibilities of the mutual stage motion for the soft landing task. My goal is to show in a simulation what control strategy can we get by optimization and then to implement a similar idea with the physical model.

In EZconn, the soft landing task is solved very conservatively by spliting the dual-stage motion into sequential procedures and leaving the soft landing part only to the voice coil stage, which starts approximately 100 µm above the wafer with components and using a very low velocity it moves towards it. This motion takes about 4 seconds on the robot which is currently in operation at the production line, therefore there might be some area for improvement.

4.2 Hybrid System Models

Models of hybrid systems consist of a continuous dynamics, described by differential or difference equations, and a discrete dynamics which models the logical rules such as ON/OFF switches or if-else rules. Events, e.g. a state crosses a certain threshold etc., are generated from the continuous dynamics and processed by the discrete logical part, which then switches betweeen the individual continuous modes of the system.

Usually, these systems are controlled by some heuristics given by the particular controlled system, for example by switching between speed and torque/force control in the cascade controller structure and no proper hybrid model is needed, as the control task is tuned intuitively. However, there exist a theoretical as well as implementation framework capable of modeling the hybrid system and generating hybrid controllers, based on *discrete hybrid automata* (DHA), which are a result from connecting the finite state machine with the continuous part of hybrid dynamics – so called switched affine system (SAS), implemented by difference equations. The connection between FSA and SAS is provided by an event generator, which extracts logic signals from the continuous part, and a mode selector, which chooses the continuous system mode based on all available logic variables. Although the word "con-

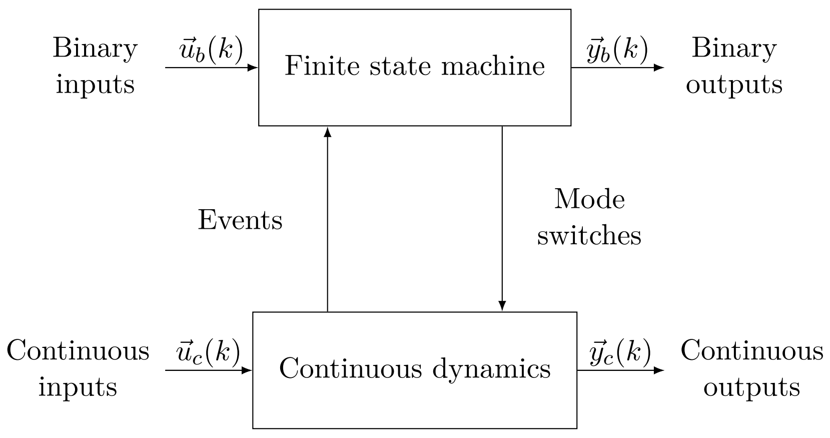


Figure 4.1: Hybrid system as DHA

tinuous" is used here, it is merely for distinction between FSA and system dynamics, since the DHA use a discrete time, mainly for computational reasons. The schematic of the functionality of DHA is depicted on the diagram 4.1. DHA can be implemented in two possible computational model – *mixed logical dynamical model* (MLD) and *piecewise affine* (PWA) systems, which are equivalent as stated in [3], in terms of same input-same output and possibility of model transformation between both forms. PWA systems are SAS where the decision over the system mode is made only by the current location of the state vector in the polyhedral structure representing the individual modes and they form the base for the explicit form of an MPC, whereas an MLD form is used in the optimization computation of the standard "online" MPC.

As for the available frameworks for implementation in Matlab®, there is the Hybrid Toolbox[1] by Alberto Bemporad and the Multi-Parametric Toolbox[12] by M. Herceg, M. Kvasnica, C.N. Jones, and M. Morari. I have implemented the model using the Hybrid Toolbox[1] by Alberto Bemporad. The implementation is written in HYSDEL introduced by F. D. Torrisi and A. Bemporad in [20]. I have chosen the Hybrid Toolbox because its author is one of the leading figures in the field of hybrid systems, hybrid MPC and MPC in general. But it would certainly be interesting to delve into the Multi-Parametric Toolbox as it seems to be a bit more developed, with a newer implementation of HYSDEL which should support vectors and matrices and more model optimization during parsing, but I have not found any direct comparison of the two.

4.2.1 Mixed logical dynamical model

According to [5], the mixed logical dynamical system (MLD) is defined as follows:

$$\vec{x}(k+1) = \mathbf{A}\vec{x}(k) + \mathbf{B}_1\vec{u}(k) + \mathbf{B}_2\vec{\delta}(k) + \mathbf{B}_3\vec{z}(k), \tag{4.1a}$$

$$\vec{y}(k) = \mathbf{C}\vec{x}(k) + \mathbf{D}_1\vec{u}(k) + \mathbf{D}_2\vec{\delta}(k) + \mathbf{D}_3\vec{z}(k), \tag{4.1b}$$

$$\mathbf{E}_2\vec{\delta}(k) + \mathbf{E}_3\vec{z}(k) \leq \mathbf{E}_1\vec{u}(k) + \mathbf{E}_4\vec{x}(k) + \mathbf{E}_5, \tag{4.1c}$$

where the state vector is $\vec{x}(k) = \begin{bmatrix}\vec{x}_c(k)\\ \vec{x}_b(k)\end{bmatrix}$, its continuous part $\vec{x}_c \in \mathbb{R}^{n_c}$ and binary part $\vec{x}_b(k) \in \{0,1\}^{n_b}$, the output vector $\vec{y}(k) = \begin{bmatrix}\vec{y}_c(k)\\ \vec{y}_b(k)\end{bmatrix} \in \mathbb{R}^{p_c} \times \in \{0,1\}^{p_b}$, the input vector $\vec{u}(k) \begin{bmatrix}\vec{u}_c(k)\\ \vec{u}_b(k)\end{bmatrix} \in \mathbb{R}^{m_c} \times \in \{0,1\}^{m_b}$ with $\vec{u}_c$ as the continuous control action and $\vec{u}_b$ as the binary ON/OFF action, and the auxiliary variables, the continuous $\vec{z}(k) \in \mathbb{R}^{r_c}$ and binary $\vec{\delta}(k) \in \{0,1\}$. The matrices $\mathbf{A}, \mathbf{B}_i, \mathbf{C}, \mathbf{D}_i, \mathbf{E}_j$ are constant, real and of appropriate dimensions. The equation 4.1a is a state equation, 4.1b an output equation and 4.1c is a set of linear inequalities, which describe the conditions for switching between the hybrid modes. The key idea of the MLD approach resides in transformation of the boolean logic variables into integers $\{0,1\}$ and expressing the logical relations into mixed-integer linear inequalities, for example that the logical expression $x_1 \vee x_2$ translates into $x_1 \geq x_2$. By doing this, the optimization problem can be numerically solved with *mixed-integer linear programming.* We will thus use this model for defining the hybrid MPC in the next section, which describes the simulation and optimization results for the hybrid model.

4.3 Hybrid System Control

As already mentioned, the usual way to control the system with any kind of dynamics describable as hybrid dynamics is by introducing heuristics and hacks such as switching between controllers or models, or not using a model at all in simple applications and tune the control intuitively to the desired behaviour. This is often a reasonable solution, because the mode switching conditions or instants are not always a priori exactly known or it is necessary to take a conservative approach to rule out some potentially expensive failure, which might occur with only a slight model mismatch. Although this is our case as well – in case of crossing the prescribed impact force constrain, the components on the wafer can get destroyed and this could happen easily with some unwanted position measurement offset – still, there is enough motivation for using the hybrid system framework, even if only for simulation purposes and showing what control strategy comes out as optimal.

Now, let us look at a model predictive control of hybrid systems. First, I state linear constrained MPC problem and then compare it with the extension for hybrid systems. There is a lot of literature concerning the model predictive control, such as [15] or [17], which is available online. As for the hybrid MPC, the dominant source is Alberto Bemporad and articles he cooperated on [5], [4], [6], his lectures [2] and even the user guide for the Hybrid Toolbox [1]. It is only natural to lean towards his publications, since he is one of the leading figures in the field of hybrid system control, and in addition to that I have used his Hybrid Toolbox for hybrid system modeling and controller generation.

In the following sections, I will omit the notation of vectors and matrices held so far, since the distinction is clear and additional arrows would only create a visual smog.

4.3.1 Linear MPC

The linear constrained model predictive regulation is based on the following optimization criterion [15]:

$$\begin{aligned}
\min_{u_t^{N-1},\, x_{t+1}^{t+N}} \quad & \frac{1}{2}x(t+N|t)^T S x(t+N|t) + \frac{1}{2}\sum_{k=0}^{N-1} x(t+k|t)^T Q x(t+k|t) + \\
& + u(t+k)^T R u(t+k) \\
\text{s.t.} \quad & x(0|t) = x(t), \\
& x(t+k+1|t) = Ax(t+k|t) + Bu(t+k), \\
& y(t+k|t) = Cx(t+k|t) + Du(t+k), \\
& u_{\min} \le u(t+k) \le u_{\max}, \quad k = 0,1,\dots,N-1, \\
& y_{\min} \le y(t+k) \le y_{\max}, \quad k = 0,1,\dots,N-1
\end{aligned} \tag{4.2}$$

where $x(t+k|t)$ denotes the state vector predicted at time $t+k$ obtained by applying the input sequence $u(t),\dots,u(t+k-1)$ to the state space model (3.13b) with initial state $x(t)$, equivalently with $y(t+k|t)$. The MPC is called linear, because it makes predictions on the linear model of the system dynamics., given in the state space form by matrices A, B, C, D.

The control is based on so called *receding horizon* strategy, which resides in computation of an optimal control problem over a finite horizon every sampling step. The length of the finite horizon is called the prediction horizon N. In case of linear MPC, the optimal control problem is solved by quadratic programming. After the problem is solved, the first step inputs are applied and the rest is thrown away, i.e. $u(t) = u_t^*(0)$, where $u_t^*(0)$ is the first input of the optimal solution for optimization procedure at time t. By doing this at every sampling step, the open-loop control problem is transformed into a closed-loop feedback control. Very important and practical characteristic of an MPC is that it inherently respects the input and output constraints, which most controllers do not.

4.3.2 Hybrid MPC

As stated in [1], the extension of tracking MPC optimization problem for hybrid models using the MLD model from equations 4.1 can be formulated as follows:

$$
\begin{aligned}
\min_{\{u,\delta,z\}} \quad & J(\{u,\delta,z\}_0^{N-1}, x(t)) = ||Q_{xN}(x(N|t)-x_r)||_p + \sum_{k=1}^{N-1} ||Q_x(x(k)-x_r)||_p + \\
& + \sum_{k=0}^{N-1} ||Q_u(u(k)-u_r)||_p + ||Q_z(z(k|t)-z_r)||_p + ||Q_y(y(k|t)-y_r)||_p \\
\text{s.t.} \quad & x((k+1)|t) = Ax(k|t) + B_1u(k) + B_2\delta(k|t) + B_3z(k|t), \\
& y(k|t) = Cx(k|t) + D_1u(k) + D_2\delta(k|t) + D_3z(k|t), \\
& E_2\delta(k|t) + E_3z(k|t) \leq E_1u(k) + E_4x(k|t) + E_5, \\
& u_{\min} \leq u(t+k) \leq u_{\max}, \quad k = 0,1,\ldots,N-1, \\
& x_{\min} \leq x(t+k|t) \leq x_{\max}, \quad k = 0,1,\ldots,N, \\
& y_{\min} \leq y(t+k) \leq y_{\max}, \quad k = 0,1,\ldots,N-1, \\
& S_x x(N|t) \leq T_x
\end{aligned}
\tag{4.3}
$$

where u is the input vector,x the state vector, y the output vector and u_r is the input vector,x_r the state vector, y_r their respective references. δ and z are the auxiliary variables of the MLD system (4.1). Q_x, Q_u, Q_y, Q_z are the weights. The last condition is a final state constraint with weight S_x saying that the terminal state set $X_f = \{x : S_x x \leq T_x\}$ should be reached after N steps.

The main difference from 4.2 lies in the fact that the optimization is computed over MLD system instead of a linear one. This results in a different kind of optimization – mixed integer linear programming as opposed to the quadratic programming in linear MPC. Thus, a different solver is used.

4.4 Hybrid Model of Dual-stage Dynamics

In this section, the mathematical description of the dual-stage system dynamics is derived. But first, let us concern a bit with the way how the actual impact is modeled.

4.4.1 Impact Model

The material characteristics play a dominant role in the impact dynamics. Most solid materials are generally viscoelastic, i.e. they have both elastic and viscous characteristic. The basic linear models describing the viscoelastic material consist of various interconnections of springs and dampers. The most basic model is the Hook's law – just a stiff mechanical string. Other simple options are either the Voigt-Kelvin model or the Maxwell model, which are parallel and series spring-damper connections, respectively. More complex models often consist of several Voigt-Kelvin and Maxwell models together, as discussed for example in [18].

For our purposes, no complex model is required, therefore I am going use the Voigt-Kelvin model, as it is used in the element "translational hard stop" in the Simscape library[11] or in [7] as well. However, metals such as steel, which is present at the force sensor, exhibit notable viscous behaviour mostly at high temperatures, otherwise it is dominantly elastic. For this reason, I used just a spring representation of impact dynamics in the simulation, which simplified the hybrid model a bit. Nevertheless, the forthcoming section about modeling and the resulting model includes the damper, I just set the damping parameter to zero for simulation. But in case the model would be used for a situation where the impact material is for example wood, it is not a problem to set the damping parameter appropriately.

The Young modulus of steel is around[12] 190−210 GPa. The spring stiffness k_3 representing the impact can be computed from the modulus, which is given by following expression:

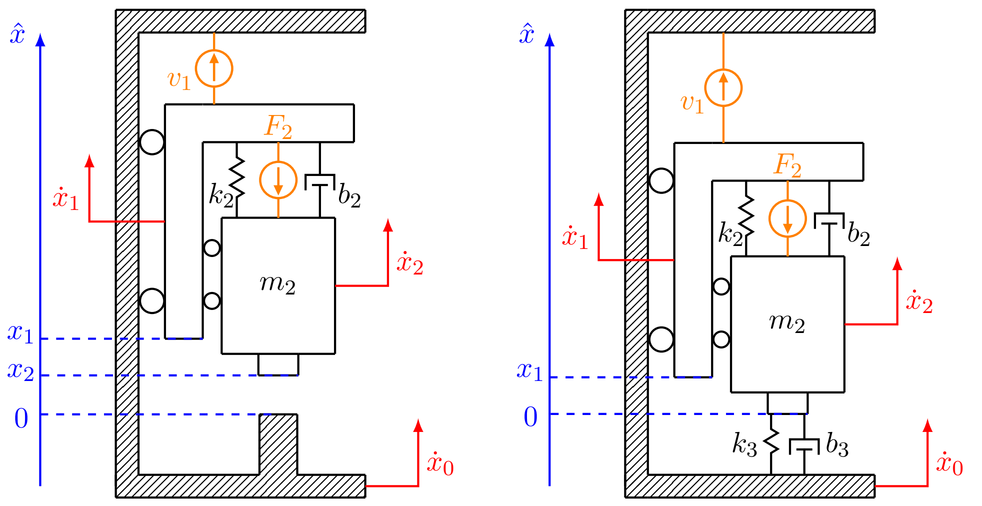
$$E = \frac{\sigma}{\epsilon} = \frac{F l_0}{A \Delta l}, \tag{4.4}$$

where σ is the stress in one axis, which is given by the acting force F on the surface A: $\sigma = \frac{F}{A}$, ϵ is a strain or proportional deformation given by the change in length to the original length ratio $\epsilon = \frac{\Delta l}{l_0}$. From 4.5, we can express the force:

$$F = \left(\frac{EA}{l_0}\right) \Delta l = k \Delta l, \tag{4.5}$$

[11]`https://www.mathworks.com/help/physmod/simscape/ref/translationalhardstop.html`

[12]`https://www.efunda.com/materials/alloys/alloy_home/steels_properties.cfm`



(a) Schematic of the mechanical system at mode M_1

(b) Schematic of the mechanical system at impact mode M_2

Figure 4.2: Two modes of the hybrid system model

which is the form of the original Hook's law $F = kx$. Considering that the area of impact is the end-effector needle with end diameter approximately $A = 1\,\text{mm}$ and the force sensor metal height is $l = 2\,\text{mm}$, this gives us:

$$k_3 = \frac{EA}{l_0} = \frac{200 \times 10^9 \pi \times 0.0005^2}{0.002}\,\text{Nm}^{-1} = 7.8540 \times 10^7\,\text{Nm}^{-1}. \tag{4.6}$$

In the end, it is just a large number representing that the spring is very stiff, the exact value is not too important for the simulation.

4.4.2 Hybrid Model

We will split the the dual-stage motion model into two modes, as depicted on the schematic 4.2. The first mode (M_1) models the behaviour of the system without any obstacles in its way, whereas the second mode (M_2) simulates the situation of impact with the target. Bond graphs of these two modes can be seen on the figure 4.3, with a description of the parameters in the table 4.1.

Let us now introduce variables for the state description: position of the ESR stage x_1, momentum of the VC stage p_2, spring displacement q_2 and impact spring displacement q_3. From the bond graph 4.3a, state-equations with states $\vec{x} = (x_1, p_2, q_2)$ can be derived:

Table 4.1: Table of hybrid model parameters and inputs

Parameter	Unit	Value	Meaning
m_2	Kg	0.426	VC moving mass, $m_2 = m_{\text{vc}} + m_{\text{load}}$
b_2	$\mathrm{N\,s\,m^{-1}}$	5.5	VC damping, static friction
k_2	$\mathrm{N\,m^{-1}}$	218.7	VC spring stiffness, both springs together
b_3	$\mathrm{N\,s\,m^{-1}}$	0	Impact damping
k_3	$\mathrm{N\,m^{-1}}$	7.8540×10^7	Impact stiffness
g	$\mathrm{m\,s^{-2}}$	9.81	Gravitational acceleration
v_1	$\mathrm{m\,s^{-1}}$	input	PMSM stage velocity
F_2	N	input	VC actuating force, $F_2 = k_f i$

$$\frac{dx_1}{dt} = v_1, \tag{4.7}$$

$$\frac{dp_2}{dt} = -m_2 g - b_2\left(\frac{p_2}{m_2} - v_1\right) - q_2 k_2 - F_2, \tag{4.8}$$

$$\frac{dq_2}{dt} = \frac{p_2}{m_2} - v_1. \tag{4.9}$$

This can be written in a matrix form:

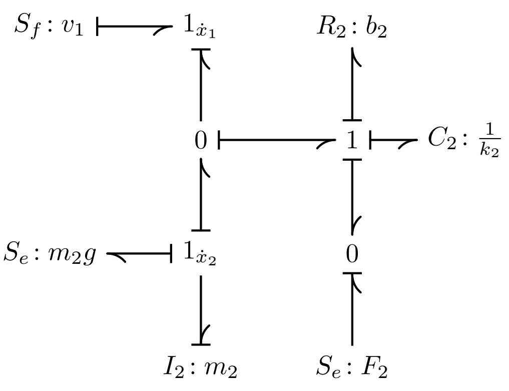
$$\begin{bmatrix} \dot{x}_1 \\ \dot{p}_2 \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\frac{b}{m_2} & -k_2 \\ 0 & \frac{1}{m_2} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ p_2 \\ q_2 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ b_2 & -k_f \\ -1 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ i \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & -m_2 g \\ 0 & 0 \end{bmatrix} \tag{4.10}$$

The last matrix is a offset, which we would like to get rid off, in order to get the equation into a state-space form 3.13b. One way to do this is through a linearization. We start by setting the state derivatives equal to zero:

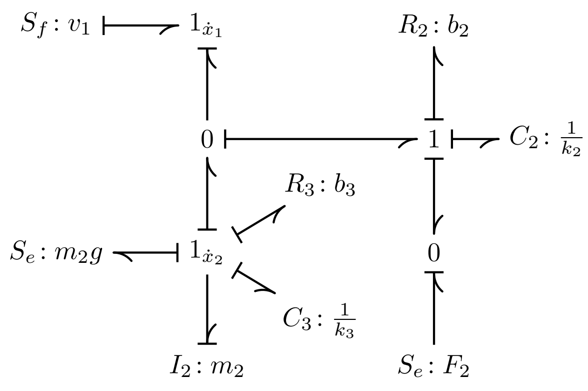
$$\begin{bmatrix} \dot{x}_1 \\ \dot{p}_2 \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\frac{b}{m_2} & -k_2 \\ 0 & \frac{1}{m_2} & 0 \end{bmatrix} \begin{bmatrix} x_1^{ss} \\ p_2^{ss} \\ q_2^{ss} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & -m_2 g \\ 0 & 0 \end{bmatrix}. \tag{4.11}$$

Next, we set the $x_1^{ss} = 0$ and $p_2^{ss} = 0$, which yields us $q_2^{ss} = -\frac{m_2 g}{k_2}$, representing the steady state displacement in an equilibrium of the spring-mass system C_2,I_2. Let us now introduce a new variable $\Delta q_2 = q_2 - q_2^{ss}$ with a meaning of a deviation from the steady state q_2^{ss}. The state vector of the first mode will be from now on $\vec{x}_1 = (x_1, p_2, \Delta q_2)$. Now we can write the system a state-space form $\frac{d\vec{x}_1}{dt} = \mathbf{A}_1 \vec{x}_1 + \mathbf{B}_1 \vec{u}$:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{p}_2 \\ \Delta\dot{q}_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\frac{b}{m_2} & -k_2 \\ 0 & \frac{1}{m_2} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ p_2 \\ \Delta q_2 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ b_2 & -k_f \\ -1 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ i \end{bmatrix}. \tag{4.12}$$



(a) Normal mode



(b) Impact mode

Figure 4.3: Hybrid system model

The output equation will be mentioned later, since its choice depends heavily on the purpose for which we want to use the model.

From the bond graph for the impact mode 4.3b, the state-equations are following:

$$\frac{dx_1}{dt} = v_1, \tag{4.13}$$

$$\frac{dp_2}{dt} = -m_2 g - b_2 \left(\frac{p_2}{m_2} - v_1 \right) - q_2 k_2 - q_3 k_3 - F_2, \tag{4.14}$$

$$\frac{dq_2}{dt} = \frac{p_2}{m_2} - v_1, \tag{4.15}$$

$$\frac{dq_3}{dt} = \frac{p_2}{m_2}. \tag{4.16}$$

Compared to the first mode, one additional state q_3 has been added. It symbolizes an impact deformation of the material, modeled as a displacement of the very stiff spring C_3. The state vector for the second mode is $\vec{x}_2 =$

$(x_1, p_2, \Delta q_2, q_3)$, and the state-space form $\frac{d\vec{x_2}}{dt} = \mathbf{A}_2\vec{x_2} + \mathbf{B}_2\vec{u}$:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{p}_2 \\ \Delta\dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -\frac{b_2+b_3}{m_2} & -k_2 & -k_3 \\ 0 & \frac{1}{m_2} & 0 & 0 \\ 0 & \frac{1}{m_2} & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ p_2 \\ \Delta q_2 \\ q_3 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ b_2 & -k_f \\ -1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ i \end{bmatrix}. \tag{4.17}$$

4.4.3 Model Implementation

For the purposes of implementation, I have extended the state-space model of the mode M_1 by the state q_3, so that the state vectors of both modes are the same ($\vec{x} = \vec{x}_1 = \vec{x}_2$), which simplifies the code. By adding one line and a column of zeros, the state-space matrices $\mathbf{A}_1$ and $\mathbf{B}_1$ have now the same dimensions as $\mathbf{A}_2$ and $\mathbf{B}_2$ respectively:

$$\mathbf{A}_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -\frac{b_2}{m_2} & -k_2 & 0 \\ 0 & \frac{1}{m_2} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{B}_1 = \mathbf{B}_2. \tag{4.18}$$

Since we are interested in the position of the end effector, let us introduce new output variable $x_2 = x_1 + \Delta q_2 + x_{\text{offset}}$, where x_{offset} is given by the mutual position of the VC stage in equilibrium and the PMSM stage and is of significance only for the control of the real system in case of precise end effector positioning, whereas for simulation purposes it can be set to zero without any loss. In the schematic on the figure 4.2b, the offset is visible as $x_{\text{offset}} = x_2 - x_1$ given that $\Delta q_2 = 0$.

To represent the transition between the modes M_1 and M_2 we introduce the event variable $\delta \in \{0, 1\}$:

$$[\delta = 1] \leftrightarrow [x_2 \leq \epsilon], \tag{4.19}$$

where ϵ is some small distance from the position of impact, i.e. $x_{\text{impact}} = 0$.

Now we introduce new continuous variables:

$$g_i = \begin{cases} x_i, & \text{if } \delta = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{4.20}$$

$$h_i = \begin{cases} x_i, & \text{if } \delta = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{4.21}$$

$$l_j = \begin{cases} u_j, & \text{if } \delta = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{4.22}$$

$$m_j = \begin{cases} u_j, & \text{if } \delta = 1, \\ 0 & \text{otherwise,} \end{cases} \tag{4.23}$$

where $x_i \in \vec{x}$ are the system states, $u_j \in \vec{u}$ are the inputs, $i \in \{1,2,3,4\}$ and $j \in \{1,2\}$.

$$\frac{d\vec{x}(t)}{dt} = \mathbf{A}_1\vec{g}(t) + \mathbf{B}_1\vec{l}(t) + \mathbf{A}_2\vec{h}(t) + \mathbf{B}_2\vec{m}(t). \tag{4.24}$$

And after discretisation with sampling period T_s:

$$\vec{x}(k+1) = \bar{\mathbf{A}}_1\vec{g}(k) + \bar{\mathbf{B}}_1\vec{l}(k) + \bar{\mathbf{A}}_2\vec{h}(k) + \bar{\mathbf{B}}_2\vec{m}(k), \tag{4.25}$$

$$\bar{\mathbf{A}}_i = e^{\mathbf{A}_i T_s}, \bar{\mathbf{B}}_i = \left(\int_0^{T_s} e^{\mathbf{A}_i \tau} d\tau\right)\mathbf{B}_i,\ i \in \{1,2\}. \tag{4.26}$$

The equations 4.19,4.20 to 4.23 and 4.25 form together a Discrete Hybrid Automaton, which can be now implemented in a textual form using HYSDEL and then compiled into MLD or PWA system frameworks. In these forms, it is possible to simulate the model in Matlab®/Simulink, verify them through reachability analysis and design model predictive controllers (MPCs) with receding horizon control based on online optimization – solving mixed integer linear (MILP)/quadratic (MIQP) problems. Another option is to solve the optimization offline and generate an explicit MPC. All these options are available in Hybrid Toolbox.

4.5 Simulation

I have used the Hybrid Toolbox to generate a hybrid MPC, which uses the hybrid model implementation from the previous section. This version of MPC from the Hybrid Toolbox library uses state measurements, therefore it knows the state q_3, which is proportional over k_3 to the impact force. This is as if there had been a force sensor present on the end effector, which is not the case with the physical model. For the simulation purposes, this is not a problem for us, since we wish to use an optimization to get an intuition for the control strategy for the physical model. For computational reasons, I had to lower the value of k_3 a bit, because with the original value, the computing had been very slow.

The outputs of the hybrid system were set as $\vec{y} = (x_2, p_2, -k_3 q_3)$. The last imput is the force acting on the wafer. I set the constraints in a following way, based on the realistic behaviour of the physical model:

- Inputs: $v_1 \in [-0.0835, 0.0835]\,\mathrm{ms}^{-1}$, $i \in [-1.2, 1.2]\,\mathrm{A}$
- States: $x_1 \in [0.1, 0.001]\,\mathrm{m}$, $\Delta q_2 \in [-0.005, 0.005]\,\mathrm{m}$, other states a large range enough to consider them unconstrained.
- Outputs: $F_{\mathrm{imp}} \in [0, 0.03]\,\mathrm{N}$

The initial conditions are set as $\vec{x}_0 = (x_1, p_2, \Delta q_2, q_3)_{|0} = (0.015\,\mathrm{m}, 0, 0, 0)$, which means that the PMSM stage is 15 mm above the impact point, the VC stage is in equilibrium position, the system is still and no force is acting on the wafer. At the start of the simulation, reference on the end effector coordinate x_2 is set to zero. Also, the references on control action inputs are set to zero, to minimize the inputs in the steady state. The sampling period for the simulation has been $T_s = 0.001\,\mathrm{s}$. The result of the simulation is depicted by the series of graphs on the figure 4.4. The control strategy picked by the optimization is clear. At first, both inputs are set to maximum. The VC stage moves to the marginal position in the direction to the impact position where it breaks by setting the current to the opposite maximum value and then stays steadily (with respect to the PMSM stage) in the marginal position, while the PMSM stage moves down with maximum velocity. When the prediction comes to the point where the impact is going to take place, the MPC lowers the end effector momentum p_2 enough to assure that the impact force constraint is not exceeded. After the impact, the end effector holds a steady position (x_2), while the PMSM stage moves to its limit value in simultaneous motion with VC stage, which moves in the opposite direction to keep the end effector position still.

When I tried to mismatch the model for the MPC with respect to the simulation plant by changing a the impact position in which the modes switch, it resulted for one direction either in a weaker and delayed impact for a low change ($\leq 0.1\,\mathrm{mm}$) or for the impact not taking place at all for a greater change ($\geq 0.1\,\mathrm{mm}$). In the other direction, the MPC threw an error that the problem is not feasible, which means that the constraint on the force could not be satisfied. This implies that knowing the position of impact is crucial for the successful soft landing task.

4.5.1 Discussion

Since the dual stage system is overactuated (there are two actuators for the linear vertical motion), it is perhaps not immediately intuitively clear what might be the optimal positioning strategy in case we are interested in the end effector coordinate x_2 only. Often in such cases, optimization tools can be used for getting an insight of how it would be beneficial to design the control algorithm, even when the controller is not for some reason applicable for the real system.

Our problem is an example of this kind of situation, since the Hybrid Toolbox does not support the code generation of the hybrid MPC in the standard online-form, only in the explicit form. The problem is, that the generation of the explicit form can get very complex and unfeasible very quickly – the dependence is exponential ([1]), with the growing number of constrains and longer horizons. When I tried to generate the explicit version of the controller I used for the simulation, and with the necessary constraints it was success-

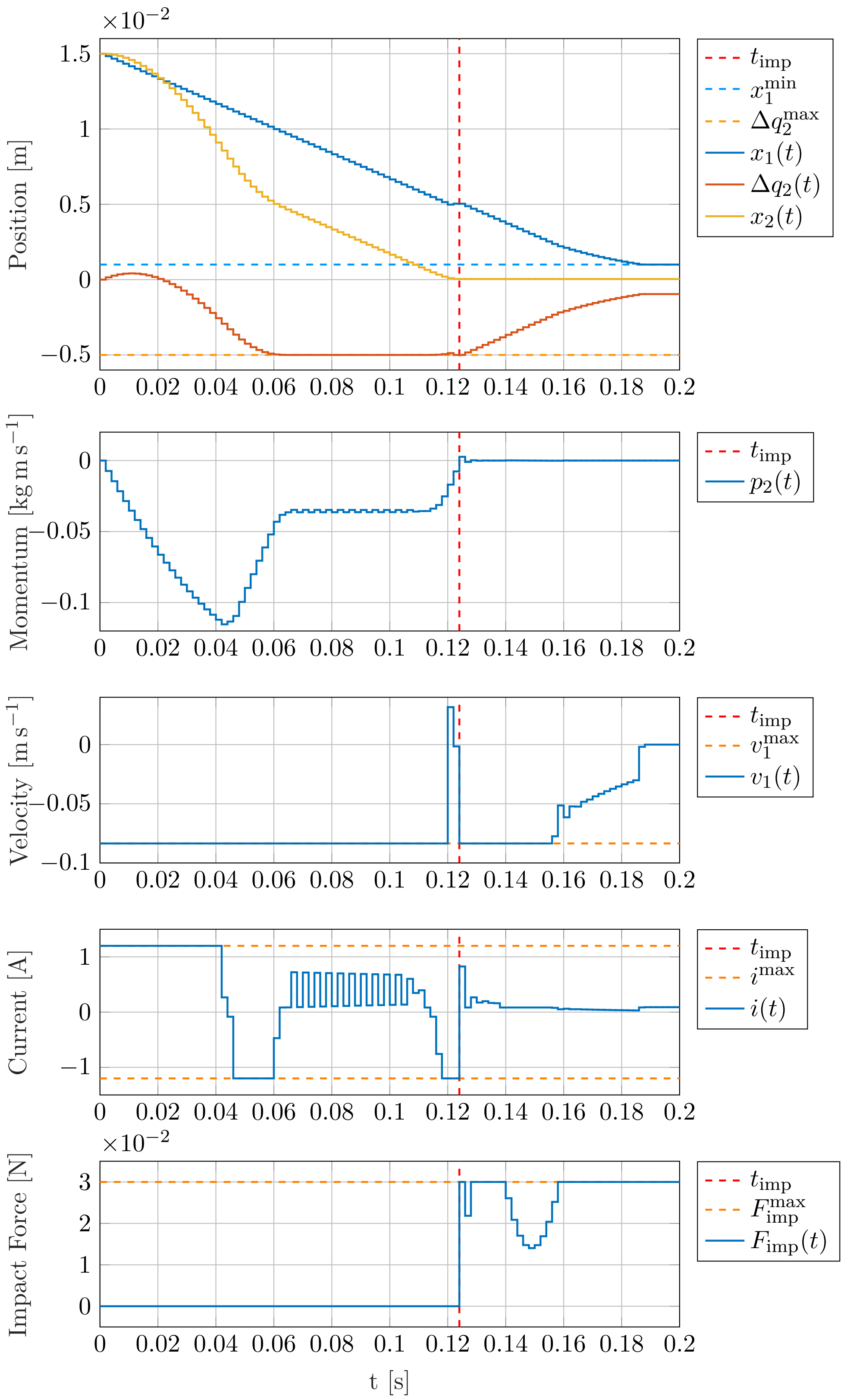


Figure 4.4: Simulation of the hybrid system with hybrid MPC

fully generated only when the prediction horizon had been set to $N = 2$. This would be useless for our problem of soft landing, since the horizon is in this case also sort of how far the controller "sees".

Imagine you were riding a car in a fog and somewhere in front of you was an obstacle. The prediction horizon would be how far you are able to see in the fog. Now, the car – the controlled system, and you – the controller, need some minimum braking distance in order not to crush destructively into the obstacle. It is only logical that the denser the fog, the slower you have to drive to be able to brake in time. When the fog is heavy, it would be unwise to deliberately burst into the thick mist, but you still need to hurry a bit, so driving slowly the whole time is also not a good option.

Apart from that, even the online form of the hybrid MPC controller is very slow in simulation and would probably not perform well real-time. This might be given by the nature of the problem, since the impact mode is an example of a stiff system. From the various examples from the Hybrid toolbox, it seems that a the hybrid MPC performs much better when the system is hybrid in a sense of piecewise linear approximation of nonlinear system dynamics, as opposed to our system, where the two modes have very different dynamics. Also, there is still the fact that the force measurement would not be available, which could be overcome by setting the threshold for the mode switch a bit more conservatively and let the q_3 state measurement be zero all the time, which would be an introduction of a heuristic. Also, the elasticity parameters of the materials participating in impact would have to be well identified in order to match the model with the reality, which is not always possible to do very conveniently, for example because of non-homogeneous material composition or uneven surface.

In the end, the result of the simulation confirmed what is important for successful soft landing:

- Precise knowledge of the impact position
- Lowering the momentum and thus the velocity to sufficient value before the impact takes place

In addition to that, it showed how would the algorithm solve the dual-stage positioning – it would place the smaller stage to the marginal position. In next section, I describe the control algorithm for the physical model inspired by the simulation.

4.6 Physical Model Control

For the reasons provided in the previous section, the hybrid MPC is not very applicable the physical model. Instead of that, I decided to use the MPC from MPC Toolbox for Matlab® because it is much more optimized for code generation and performs quite well with the physical model. It does not use the hybrid model but the linear model (4.12) instead, therefore some heuristic strategy has to be chosen in order to implement the soft landing task. The output matrix is chosen as:

$$C = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.27}$$

to output the variables $\vec{y} = (x_2, p_2, \Delta q_2)$.

A classical industrial approach would be to separate the movement of the stages – move with the PMSM stage to a designated position and then stop. Afterwards (or during, it makes not much of a difference), start moving the VC stage in velocity mode with a low constant velocity and compute the expected position. When the expected position starts to differ from the real position by some threshold, the contact has been made. Now, switch to the force mode and set some constant force greater than zero and lower than the force constraint.

What I do is to combine the approach from the simulation with the industrial approach. There is a clear necessity to split the control strategy into two modes, where in the first mode the dual-stage can move as fast as possible in order to make the whole procedure fast and in the second mode the momentum has to be limited to make sure that the force constraint is not crossed. My solution is the MPC with switched weights – a different values of the weight matrices in the individual modes, with a switch based on a position threshold for the coordinate x_2. Using this strategy, it is not necessary to switch between controllers such as MPC→PID cascade or MPC→MPC with different models, but the MPC with linear model is enough.

The weights we are going to switch are the weights on outputs Q_x and on the inputs Q_u. In the first mode, they are set as $Q_y^1 = (20, 5, 10)$, $Q_u^1 = (1, 0.01)$, which gives preference to the x_2 tracking and prevents the VC stage to move too much. The input In the second mode, the weights change to $Q_y^2 = (50, 5, 0)$ and $Q_u^2 = (1 \times 10^3, 1 \times 10^{-8})$, which basically prohibits the PMSM stage to move further through the input limitation, while the low input weight on the current is encourages the movement of the VC stage together with the weight on x_2.

The procedure is initiated by setting the reference on x_2 to zero at time $t_{\text{init}} = 0.37\,\text{s}$. The constraints are set according to the real possibilities of the dual-stage:

- Inputs: $v_1 \in [-0.0835, 0.0835]\,\text{ms}^{-1}$, $i \in [-1.2, 0.1]\,\text{A}$
- Outputs: $x_2 \in [0, 0.08]\,\text{m}$, $\Delta q_2 \in [-0.001, 0.005]\,\text{m}$

The most important constraint for the task of soft landing is the constraint on the current $i_{\text{lim}} = 0.1\,\text{A}$, which assures that after the impact, the current rises to this value and no further. The impact force then stays within the constrain of $F_{\text{imp}}^{\max} = 0.3\,\text{N}$, as seen on the figure 4.5 on the bottom graph. In addition to the constraints on inputs and outputs, I have set constraints on the rate of change of both imputs, i.e. the maximum difference between input $u(k)$ and $u(k+1)$, in a following way:

- $|v_1(k+1) - v_1(k)| \leq 0.5\,\text{ms}^{-1}$
- $|i(k+1) - i(k)| \leq 0.1\,\text{A}$

This makes the behaviour much more smooth and in case of the PMSM stage, it supplements the model of a transience in the reference tracking, which has been omitted. In another words, the velocity v_1 cannot change immediately, which is not included in the model, therefore limiting the rate of change of the computed system inputs helps the MPC to be more realistic in prediction.

4.6.1 Discussion

From the graphs 4.5 and 4.6, we can see that the resulting behavior satisfies the set constraints and results in very low impact force (the impulse at time t_{imp}), and its subsequent rise to a steady value around $F = 0.2\,\text{N}$, since in production it is necessary to maintain some contact force in order for a successful component pick up on one source stand and soldering on the target stand.

In EZconn, the setting is such that the two stages do not move simultaneously. The starting position is either 60 mm above the wafer, as is the case of the conducted experiment on the physical model, or 11 mm. It moves with speed of approximately $5\,\text{mms}^{-1}$. The subsequent motion of the VC stage starts at 100 µm and it takes about 4 s to perform the soft landing procedure. So in the first case, it takes about 16 s to finish the the total motion in z axis together with soft landing and in the second case 6.2 s. My algorithm with the physical model took about 2.6 s to make the impact or 3.5 s to reach the steady force from the position of 60 mm. This might seem as a wonderful improvement, but we have to keep in mind that it is only a model and thus a simplification of the real situation, which is of course much more complex, since the robot has twelve different actuators. However, most of them are programmed to move conservatively in a sequential manner, therefore there might be enough space for optimization of a certain part of the robot procedure without affecting the other parts. My result is therefore valuable as a certain direction indicator of what could be done on the real robot in the future to speed up the production process.

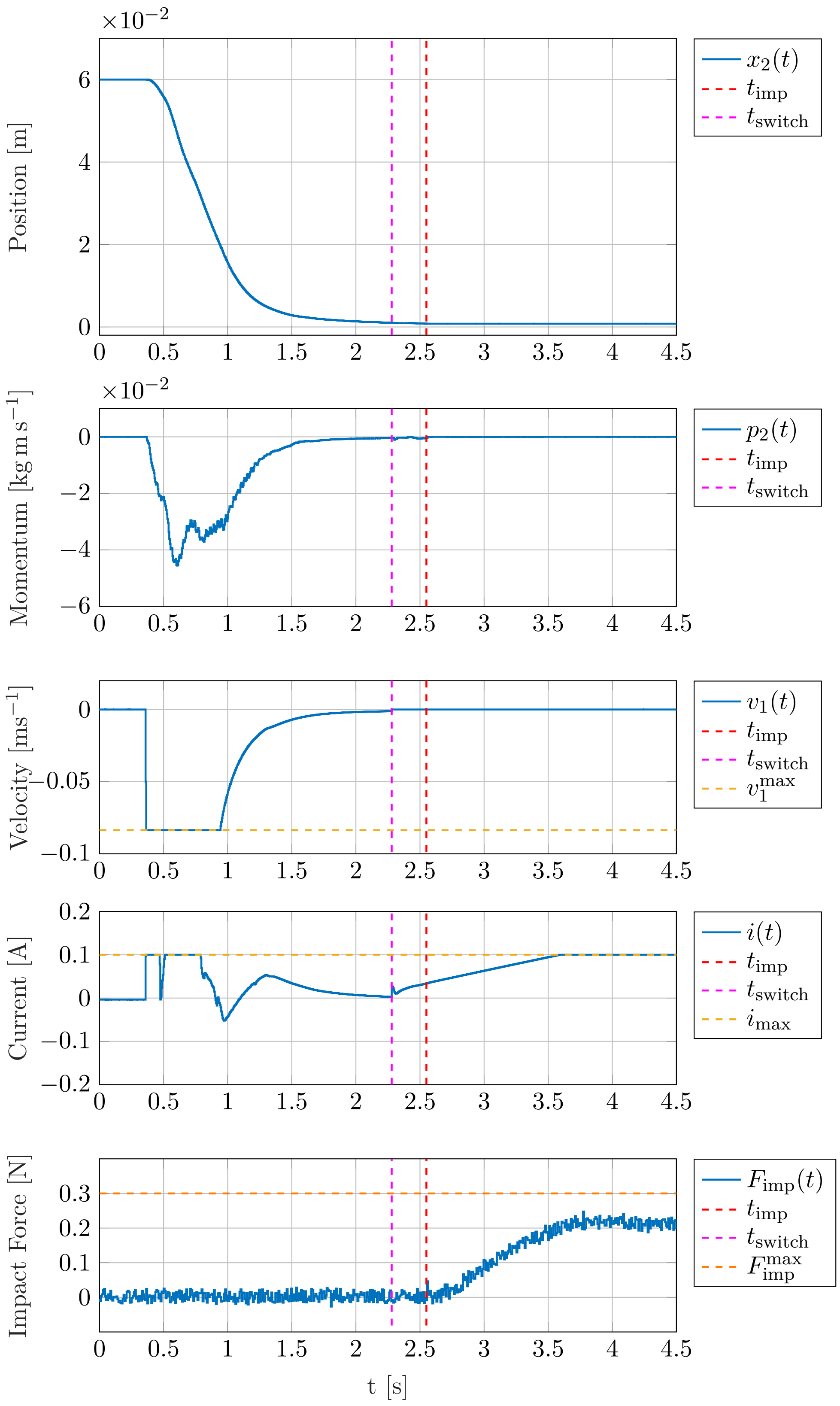


Figure 4.5: Soft landing on physical model

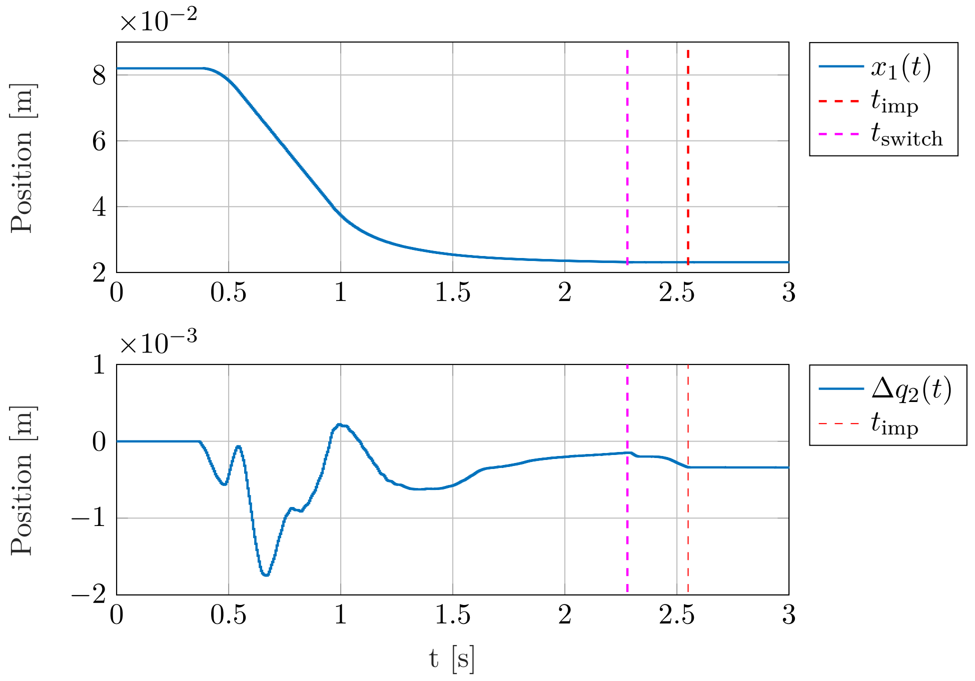


Figure 4.6: Position of the two stages, x_1 is shifted by $x_{\text{offset}} = 2.2\,\text{cm}$ given by the mutual stage position

Conclusion

During the work on this thesis, I have successfully designed a driver for the voice coil motor and produced it in two variants of PCBs – one for the experimental setup and connection to the dSpace platform and one as a BoosterPack for TI Launchpad platform. Then I have finalized the dual-stage in terms of the necessary hardware changes for the application as a vertical motion dual-stage experimental setup, together with communication interface between the PMSM driver and dSpace MicroLabBox. Next, I have modeled the dual-stage as a hybrid system using the Hybrid Toolbox by A. Bemporad and applied the hybrid model predictive control to it in a simulation. Based on the results of the simulation, I have designed a suboptimal control strategy for the experimental setup using a linear MPC with weight switching method, which assured a successful soft landing of the end effector in a relatively short time compared to the conservative industrial control of the assembly robot in EZ-conn. Thus the goals of this thesis have been met, with a minor deflection from the original intention for the research direction.

Contrary to the assignment, which has been formulated before the complete finishing of the hardware part of this thesis as a possible experimental direction, a different has been chosen. Some minor experimentation with iterative learning control has been tried on the dual-stage, but the results were not very promising, partly because the hardware part of this thesis took a bit longer than previously expected. With my supervisor, we have evaluated that the experimentation with the hybrid system framework seems much more promising, which proved to be right.

In summary, the main contributions of this thesis are:

- Voice coil driver electronics and control design and production of two driver versions for two different platforms
- Application of the hybrid system modeling framework and hybrid MPC on the dual stage soft landing problem in simulation

- Design of a practical suboptimal control algorithm for the soft landing with the physical model

Future work

Due to the problems with noise in the voice coil motor version in EZconn, I am going to redesign the driver to include a decoder for the quadrature encoder. The decoder will receive differential signals, in order to make the position measurement immune to the noise in the particular voice coil motor version. After resolving this problem, the driver can be tested as a part of a control system of the assembly robot, where it could potentially form a base for the replacement of the inconvenient commercial driver in the future.

I would also like to continue in the experimentation with the hybrid system framework, notably by exploring the possibilities of the Multi-Parametric Toolbox as an alternative to the Hybrid Toolbox.

Bibliography

[1] A. Bemporad. Hybrid Toolbox - User's Guide, 2004. http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox.

[2] A. Bemporad. Model predictive control phd course, 2019. http://cse.lab.imtlucca.it/~bemporad/mpc_course.html.

[3] A. Bemporad, G. Ferrari-Trecate, and M. Morari. Observability and controllability of piecewise affine and hybrid systems. *IEEE Transactions on Automatic Control*, 45(10):1864–1876, Oct 2000.

[4] A. Bemporad, W.P.M.H. Heemels, and B. Schutter, de. On hybrid systems and closed-loop mpc systems. In *Proceedings of the 40th IEEE Conference on Decision and Control, December 2001, Orlando, vol. 2*, pages 1645–1650, United States, 2001. Institute of Electrical and Electronics Engineers (IEEE).

[5] Alberto Bemporad and Manfred Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, March 1999.

[6] F. Borrelli, A. Bemporad, M. Fodor, and D. Hrovat. An mpc/hybrid system approach to traction control. *IEEE Transactions on Control Systems Technology*, 14(3):541–552, May 2006.

[7] R. Carloni, R. G. Sanfelice, A. R. Teel, and C. Melchiorri. A hybrid control strategy for robust contact detection and force regulation. In *2007 American Control Conference*, pages 1461–1466, July 2007.

[8] John Chiasson. *Modeling and High-Performance Control of Electric Machines.* John Wiley & Sons, Inc., 2005.

[9] John J. Craig. *Introduction to Robotics: Mechanics and Control.* Pearson, 330 Hudson Street, NY 10013, 4th edition, 2018.

[10] Marco Crescentini, Marco Marchesi, Aldo Romani, Marco Tartagni, and Pier Andrea Traverso. Bandwidth limits in hall effect-based current sensors. *ACTA IMEKO*, 6:17, 12 2017.

[11] Gene F. Franklin, J. Da Powell, and Abbas Emami-Naeini. *Feedback Control of Dynamic Systems.* Prentice Hall Press, Upper Saddle River, NJ, USA, 7th edition, 2014.

[12] M. Herceg, M. Kvasnica, C.N. Jones, and M. Morari. Multi-Parametric Toolbox 3.0. In *Proc. of the European Control Conference*, pages 502–510, Zürich, Switzerland, July 17–19 2013. `http://control.ee.ethz.ch/~mpt`.

[13] W. Hoffmann, K. Peterson, and A. G. Stefanopoulou. Iterative learning control for soft landing of electromechanical valve actuator in camless engines. *IEEE Transactions on Control Systems Technology*, 11(2):174–184, March 2003.

[14] Sang-Hoon Kim. *Electric Motor Control, 1st Edition.* Elsevier Science, 05 2017. `https://www.elsevier.com/books-and-journals/book-companion/9780128121382`.

[15] J.M. Maciejowski. *Predictive Control with Constraints.* Prentice Hall, 2002.

[16] K. Peterson, A. Stefanopoulou, T. Megli, and M. Haghgooie. Output observer based feedback for soft landing of electromechanical camless valvetrain actuator. In *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*, volume 2, pages 1413–1418 vol.2, May 2002.

[17] J Rawlings, D.Q. Mayne, and Moritz Diehl. *Model Predictive Control: Theory, Computation, and Design,2nd Edition.* 02 2019. `https://sites.engineering.ucsb.edu/~jbraw/mpc/MPC-book-2nd-edition-2nd-printing.pdf`.

[18] D. Roylance. Engineering Viscoelasticity. 24 2001. http://web.mit.edu/course/3/3.11/www/modules/visco.pdf.

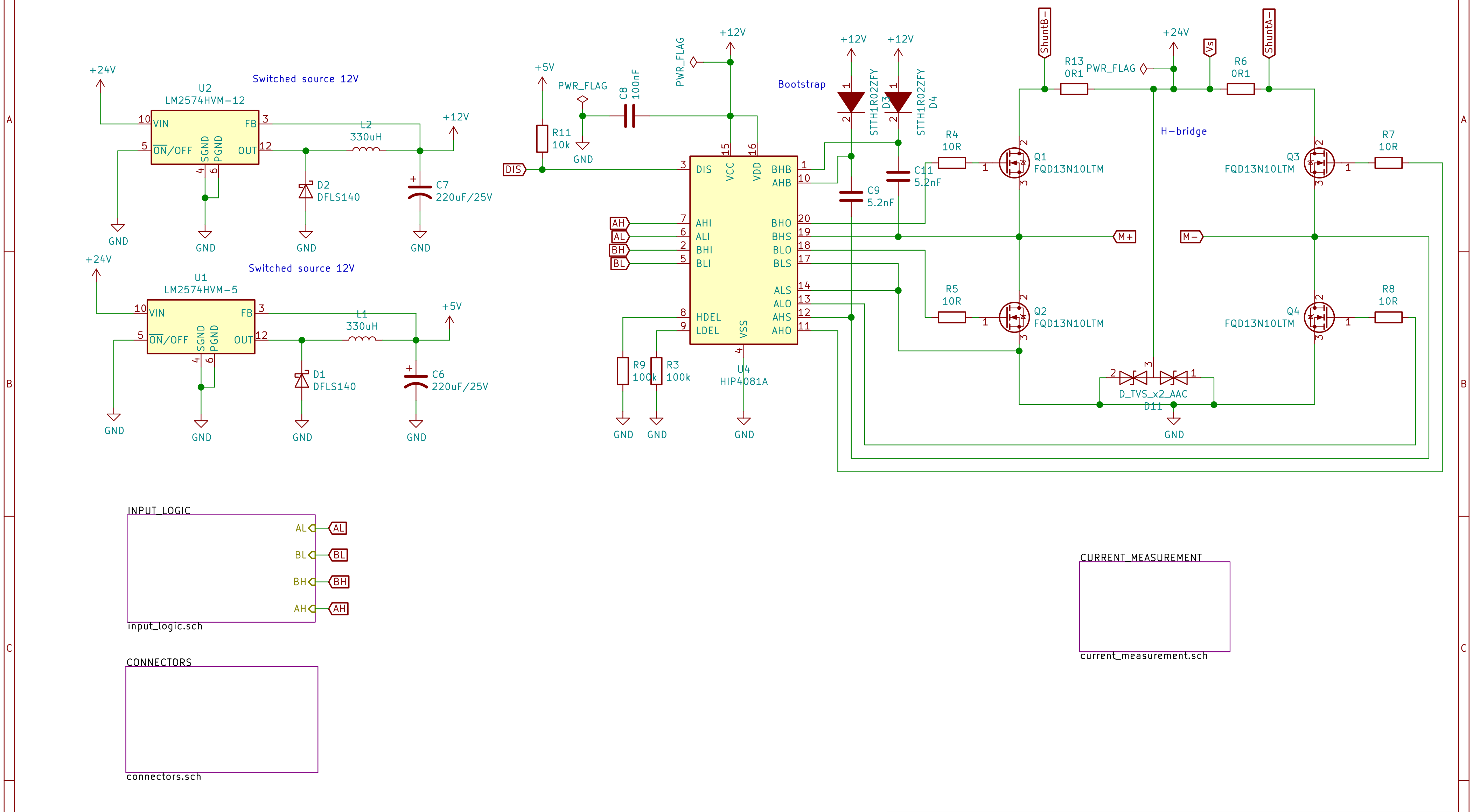
[19] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: Modelling, Planning and Control.* Springer-Verlag London, 2010.

[20] F. D. Torrisi and A. Bemporad. Hysdel-a tool for generating computational hybrid models for analysis and synthesis problems. *IEEE Transactions on Control Systems Technology*, 12(2):235–249, March 2004.

Appendix A

Apendix

On following pages, the driver schematic together with front and back PCB layout of the driver boards, first the version for the experimental setup, then the BoosterPack for TI Launchpad.



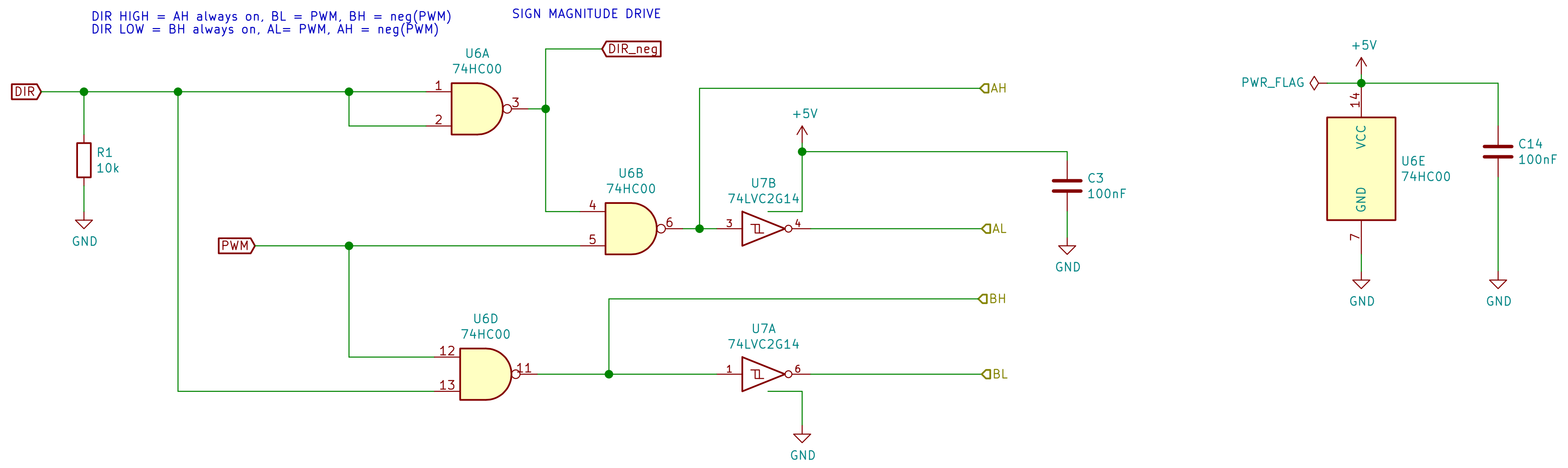
Author: Adam Polák

FEE CTU Prague, aa4cc

Sheet: /
File: VC2_kicad.sch

Title: Voice Coil Driver Boosterpack

Size: A4 | Date: | **Rev:**
KiCad E.D.A. kicad (5.0.0) | Id: 1/4



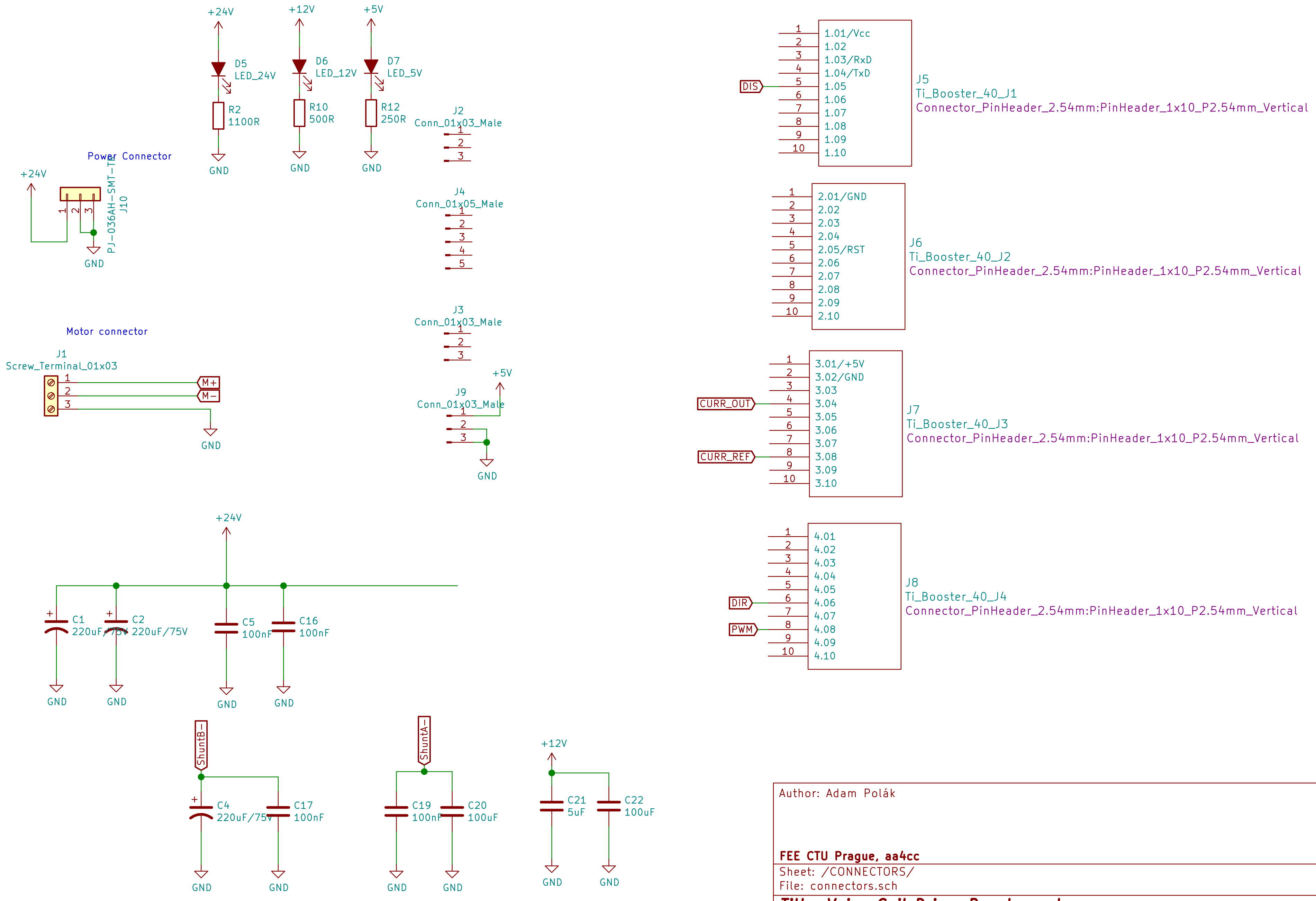
Author: Adam Polák

FEE CTU Prague, aa4cc

Sheet: /INPUT_LOGIC/
File: input_logic.sch

Title: Voice Coil Driver Boosterpack

Size: A4	Date:	**Rev:**
KiCad E.D.A. kicad (5.0.0)		Id: 2/4



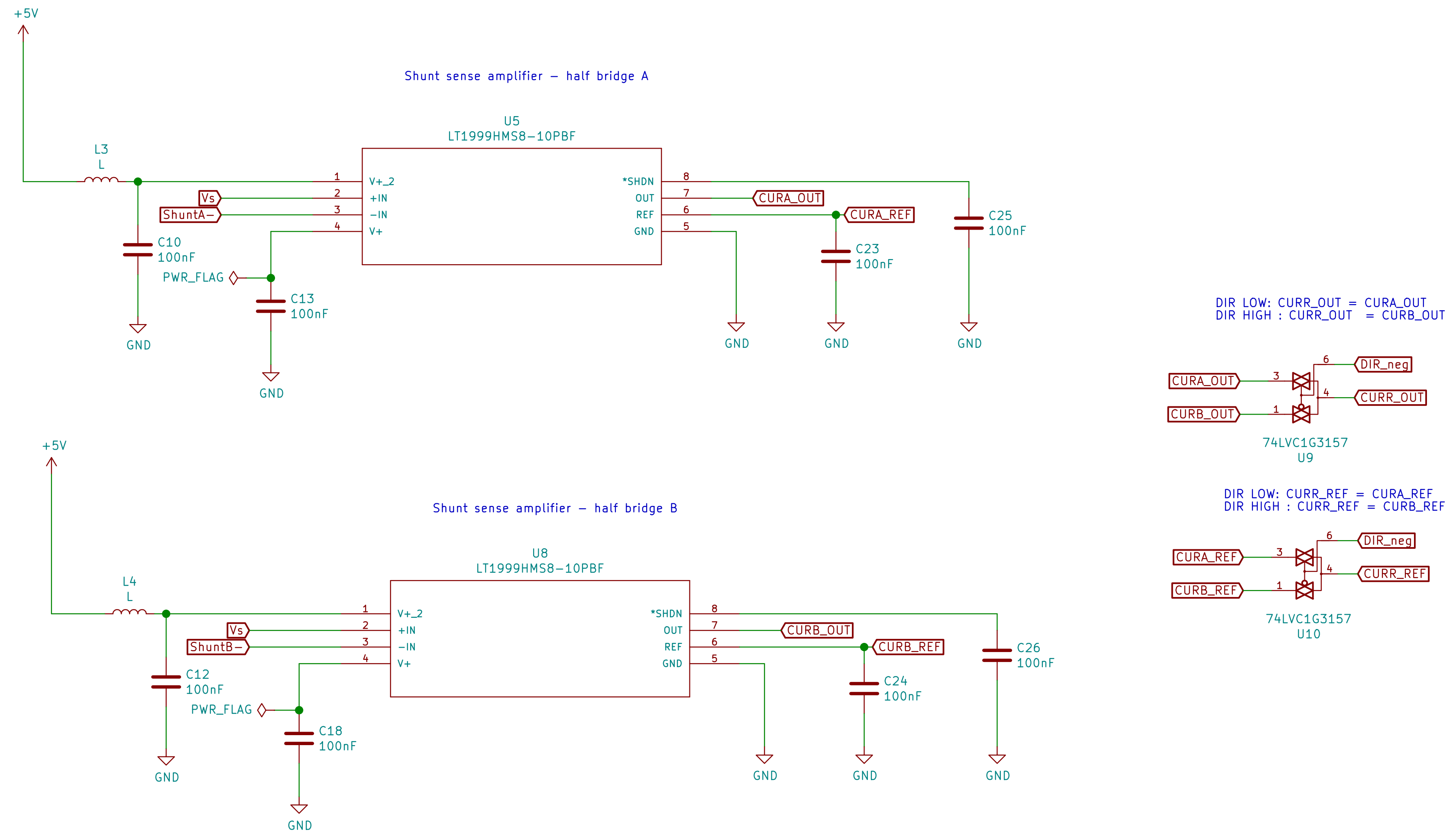
Author: Adam Polák

FEE CTU Prague, aa4cc

Sheet: /CONNECTORS/
File: connectors.sch

Title: Voice Coil Driver Boosterpack

Size: A4	Date:	**Rev:**
KiCad E.D.A. kicad (5.0.0)		Id: 3/4



Author: Adam Polák

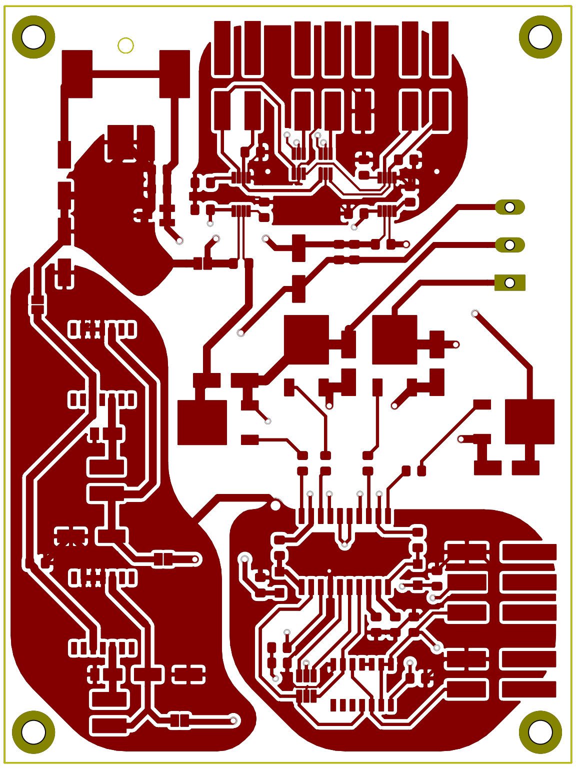
FEE CTU Prague, aa4cc

Sheet: /CURRENT_MEASUREMENT/

File: current_measurement.sch

Title: Voice Coil Driver Boosterpack

Size: A4	Date:	**Rev:**
KiCad E.D.A. kicad (5.0.0)		Id: 4/4



Author: Adam Polák

ČVUT FEL, aa4cc

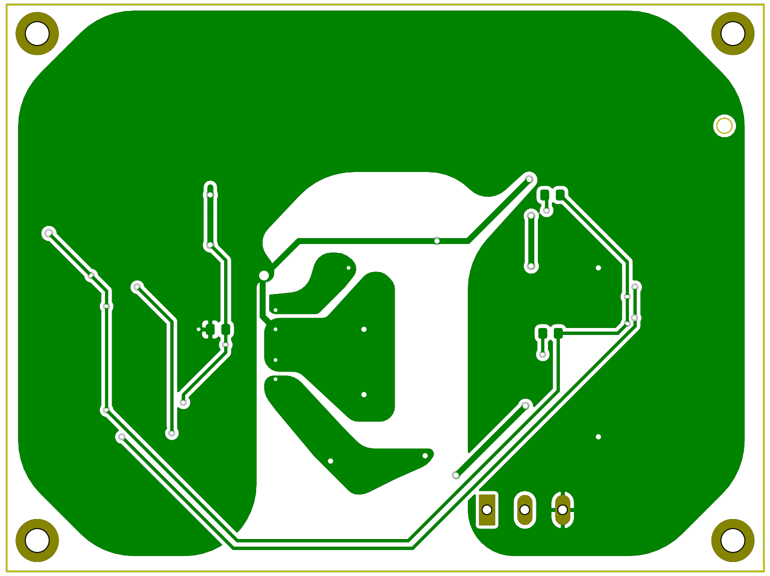
Sheet:

File: VC2_kicad.kicad_pcb

Title: Voice coil driver v1.c

Size: A4 Date: Rev:

KiCad E.D.A. kicad (5.0.0) Id: 1/2

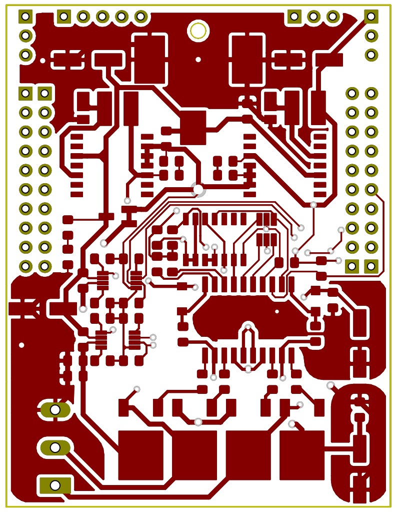


Author: Adam Polák
ČVUT FEL, aa4cc

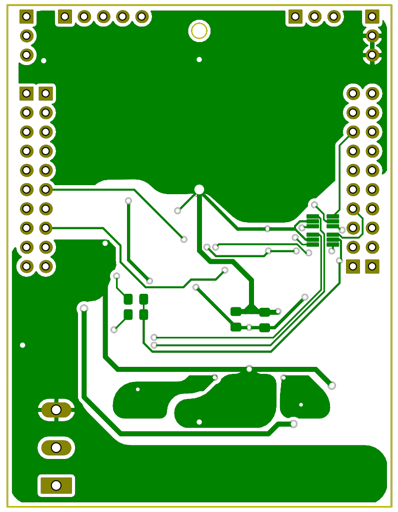
Sheet:
File: VC2_kicad.kicad_pcb

Title: Voice coil driver v1.c

Size: A4	Date:	**Rev:**
KiCad E.D.A. kicad (5.0.0)		Id: 2/2



Author: Adam Polák
ČVUT FEL, aa4cc
Sheet:
File: VC2_kicad.kicad_pcb
Title: Voice coil driver BoosterPack v1.0
Size: A4 Date: **Rev:**
KiCad E.D.A. kicad (5.0.0) Id: 1/1



Author: Adam Polák **ČVUT FEL, aa4cc**		
Sheet: File: VC2_kicad.kicad_pcb		
Title: Voice coil driver BoosterPack v1.0		
Size: A4	Date:	**Rev:**
KiCad E.D.A. kicad (5.0.0)		Id: 1/1