

Bachelor Thesis



**Czech
Technical
University
in Prague**

F3

**Faculty of Electrical Engineering
Department of Control Engineering**

Development of control algorithms laboratory model

Matěj Kopecký

**Supervisor: doc. Ing. Tomáš Haniš, Ph.D.
Field of study: Cybernetics and Robotics
May 2022**

I. Personal and study details

Student's name: **Kopecký Mat j** Personal ID number: **491559**
Faculty / Institute: **Faculty of Electrical Engineering**
Department / Institute: **Department of Control Engineering**
Study program: **Cybernetics and Robotics**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Development of control algorithms laboratory model

Bachelor's thesis title in Czech:

Vývoj laboratorního modelu pro výuku řídicích algoritm

Guidelines:

The goal of the thesis is to design laboratory model to support control and system theory laboratories. Technical courses heavily rely on hands-on experience gained in laboratories. Developed mobile laboratory model, with carry-home possibilities, will enable students practical experience in remote or hybrid form of course. Thesis will address following points:

- 1) Review of current control theory laboratory models, software, and hardware possibilities.
- 2) Development of new mechatronic laboratory model suited for basic and advance control systems design and demonstration
- 3) Implementation of Matlab & Simulink libraries and demo Simulink mathematical and control model.
- 4) Verification of developed laboratory model and developed control strategy.

Bibliography / sources:

- [1] J. David Powell, Gene F. Franklin, Abbas Emami-Naeini: Feedback Control of Dynamic Systems. Prentice Hall; 5 ed., 2005, ISBN: 0131499300
[2] Armin Steinhauser, Maarten Verbandt, Niels van Duijkeren, Ruben Van Parys, Laurens Jacobs, Jan Swevers, Goele Pipeleers, Low-cost Carry-home Mobile Platforms for Project-based Evaluation of Control Theory, IFAC-PapersOnLine, Volume 50, Issue 1, 2017, Pages 9138-9143, ISSN 2405-8963, <https://doi.org/10.1016/j.ifacol.2017.08.1718>.

Name and workplace of bachelor's thesis supervisor:

doc. Ing. Tomáš Haniš, Ph.D. Department of Control Engineering FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **28.01.2022** Deadline for bachelor thesis submission: **20.05.2022**

Assignment valid until:
by the end of summer semester 2022/2023

doc. Ing. Tomáš Haniš, Ph.D.
Supervisor's signature

prof. Ing. Michael Šebek, DrSc.
Head of department's signature

prof. Mgr. Petr Páta, Ph.D.
Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt

Student's signature

Acknowledgements

I would like to express gratitude to my supervisor Ing. Tomáš Haniš, PhD. for his support, friendly advice and most importantly for his time.

Also, I would like to thank Ing. Denis Efremov for his advice as well.

Tomáš Twardzik helped me with the 3D models, so I would like to thank him as well.

Last but not least I would like to thank Ing. Krištof Pučejdl for providing me with his 3D printers.

Declaration

I hereby declare that this bachelor thesis was developed independently and that I have cited all used sources of information by the methodical instructions for observing the ethical principles in the preparation of a university thesis.

In Prague, May 20, 2022

.....

Abstract

The goal of this thesis is to design a laboratory model to support control and system theory laboratories. The thesis reviews the state-of-the-art available laboratory models and compares their capabilities to new requirements.

A new laboratory model is consequently developed. It is a cost-effective, simple and robust carry-home model. It uses the BeagleBone Blue a Linux-based computer as the main computing unit.

Model hardware consists of Merkur platform, 3D printed components and electrical components. This model can be simply described as a pendulum on a cart.

The laboratory model is physically modelled with Lagrange's equations. Identification experiments were conducted. Simulink demo simulation, as well as a simple feedback control loop, were developed.

Keywords: automatic control, BeagleBone Blue, laboratory model, pendulum on a cart

Supervisor: doc. Ing. Tomáš Haniš, Ph.D.
Czech Technical University in Prague,
Faculty of electrical engineering,
Department of Control Engineering -
K13135,
Karlovo náměstí 13,
121 35 Praha 2

Abstrakt

Cílem této práce je navrhnout laboratorní model pro podporu laboratoří automatického řízení a teorie systémů. Práce reviduje nejmodernější dostupné laboratorní modely a porovnává jejich možnosti s novými požadavky.

Následně je vyvinut nový laboratorní model. Jedná se o levný, jednoduchý a robustní model, který si studenti mohou vzít domů. Jako hlavní výpočetní jednotku používá BeagleBone Blue, počítač založený na Linuxu.

Hardware modelu se skládá z platformy Merkur, 3D tištěných součástek a elektrických součástek. Tento model lze zjednodušeně popsat jako kyvadlo na vozíku.

Laboratorní model je fyzikálně modelován pomocí Lagrangeových rovnic. Byly provedeny identifikační experimenty. Byla vyvinuta demo simulace Simulink a také jednoduchá zpětnovazební regulační smyčka.

Klíčová slova: automatické řízení, BeagleBone Blue, kyvadlo na vozíku, laboratorní model

Překlad názvu: Vývoj laboratorního modelu pro výuku řídicích algoritmů

Contents

1 Introduction	1		
1.1 Goal and requirements	1		
2 Hardware	3		
2.1 Design overview	4		
2.1.1 Chassis	4		
2.1.2 Hood	6		
2.1.3 Pendulum	6		
2.2 Electrical components and sensors	8		
2.2.1 BeagleBone Blue	8		
2.2.2 Merkur DC motors	9		
2.2.3 LiPo batteries	9		
2.2.4 I2C bus sensors	11		
2.2.5 Accessories	11		
2.3 3D printed components	11		
2.3.1 Pendulum frame	12		
2.3.2 I2C multiplexer hub mounting	13		
2.3.3 Magnetic rotary position sensor mounting	13		
2.3.4 BeagleBone Blue holder	14		
3 Software	15		
3.1 BeagleBone Blue setup	15		
3.1.1 Debian Image	15		
3.1.2 Starting BeagleBone Blue	15		
3.1.3 Debian password	16		
3.2 Matlab and Simulink setup	16		
3.2.1 Installing BeagleBone Blue Support Package	17		
3.2.2 I2C Simulink library fix	17		
3.3 Using Simulink to control BeagleBone Blue	18		
3.3.1 BeagleBone Blue IMU calibration	18		
3.3.2 Connecting to BeagleBone Blue with Matlab	18		
3.3.3 Simulink setup for the BeagleBone Blue	18		
3.3.4 Using Simulink blocks to control the model	21		
3.3.5 Getting the relative angle to the start of the simulation	26		
3.3.6 Logging Signals	28		
3.3.7 Checksum mismatch error handling	28		
4 Control	31		
4.1 Modelling dynamic system	31		
4.1.1 Cart energy and dissipation	32		
4.1.2 Pendulum energy and dissipation	33		
4.1.3 Lagrange's equations	33		
4.1.4 Linearisation	34		
4.2 Model identification	35		
4.3 Control	40		
5 Conclusion	47		
Bibliography	49		
A Attachments	51		

Figures

2.1 The model diagram	3	3.9 Angle registers [2]	25
2.2 The model pictures	4	3.10 I2C Read block parameters . . .	26
2.3 The chassis with its components; 1: non-driven axis, 2a, 2b: driven wheels, 3: magnet with magnetic rotary sensor	5	3.11 Absolute to relative angle block sequence	27
2.4 Closeup of the magnet of the lower magnetic rotary sensor	5	3.12 Unwrap block parameters with the π radians tolerance	27
2.5 Top of the chassis with its components; 1: non-driven axis, 2a, 2b: driven wheels, 3: magnet, 4: BeagleBone Blue, 5: battery pack . .	6	3.13 To File block parameters	29
2.6 The hood with its components; 6: lower magnetic rotary sensor, 7: multiplexer hub, 8: pendulum frame, 9: upper magnetic rotary sensor . . .	7	3.14 StopFcn Callback with the getFile() command	30
2.7 The lower magnetic rotary sensor	7	3.15 CloseFcn Callback with the deleteFile() command	30
2.8 The pendulum frame with its components; 10: pendulum, 9: upper magnetic rotary sensor with a magnet in the circle	8	4.1 The model parameters	31
2.9 The upper magnet closeup	9	4.2 Pendulum identification: response to initial condition	36
2.10 The model electrical diagram . .	10	4.3 Cart identification: step response	37
2.11 Pendulum frame comparison . .	12	4.4 Model identification: step response: pendulum	37
2.12 Multiplexer hub mounting comparison	13	4.5 Model identification: step response: cart	38
2.13 Magnetic rotary position sensor mounting comparison	13	4.6 Identification GUI	40
2.14 BeagleBone Blue holder comparison	14	4.7 Feedback control loop	40
3.1 Model properties: Callbacks: InitFcn	19	4.8 Pendulum filter rltool design . . .	41
3.2 Model properties: Callbacks: InitFcn	20	4.9 Pendulum without dampening . .	42
3.3 Button block connected to the LED block	20	4.10 Motor input: pendulum is not damped	42
3.4 DC motor block with maximum input power	21	4.11 Pendulum with dampening	43
3.5 I2C multiplexer hub: command byte definition [1]	22	4.12 Motor input: pendulum is damped	43
3.6 I2C write block with the multiplexer address	22	4.13 Displacement P regulator rltool desing	44
3.7 Priority setting in the Block Properties	24	4.14 Model displacement without dead-zone correction	45
3.8 Simulink blocks to read the angle value from the magnetic sensor . . .	25	4.15 Model displacement with dead-zone correction	45

Chapter 1

Introduction

Hands-on education is arguably the most important education for an electrical engineer there is. Lessons taught in laboratories and with actual hardware are irreplaceable. When the Covid-19 pandemic happened, students and teachers could not meet at the laboratory and therefore missed a lot of experience, they could have got.

Students of the Faculty of Electrical Engineering need this kind of experience, so the goal was to either buy or develop a new laboratory model. The laboratory models available [3], [4], [5] did not meet the requirements. The management of the Department of Control Engineering decided to develop a new laboratory model that would meet the requirements.

1.1 Goal and requirements

This thesis aims to develop a control algorithms laboratory model. The requirements are as follows.

- Laboratory model must be a simple, cost-effective and robust device.
- It has to be a carry-home model.
- It must be suited for basic and advanced control system design and demonstration.
- It must be able to run Matlab Simulink simulations.

Chapter 2

Hardware

This chapter describes the hardware components of the model. The model consists of mechanical components including 3D printed parts and electrical components. Figure 2.1 depicts a diagram of the model. Figure 2.2 shows pictures of the model.

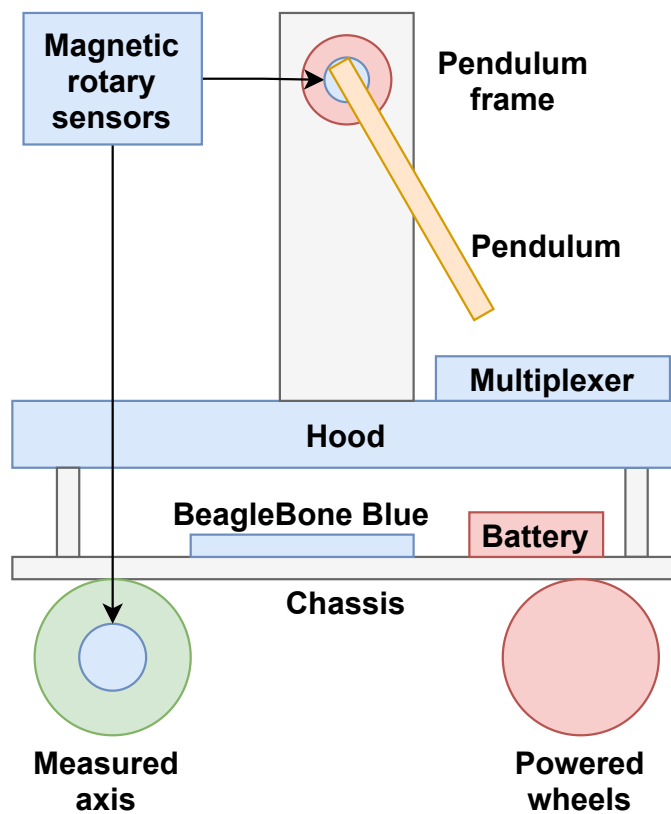


Figure 2.1: The model diagram

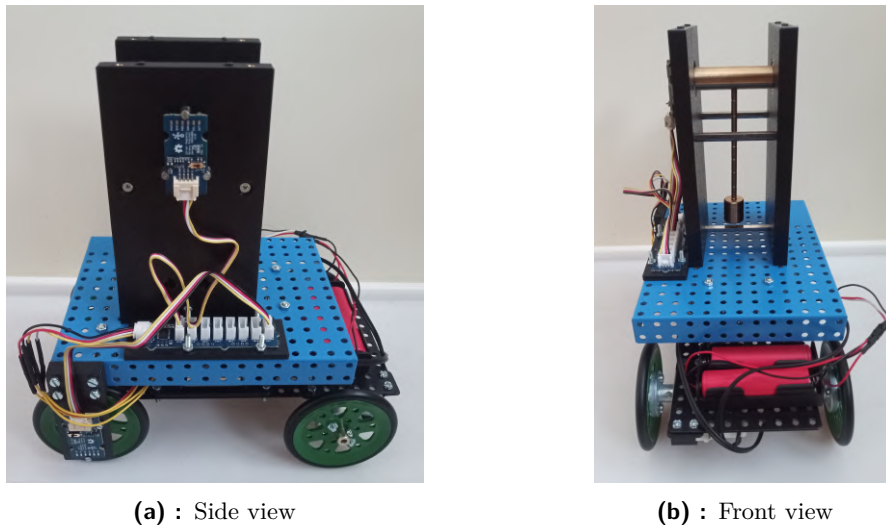


Figure 2.2: The model pictures

2.1 Design overview

This section describes the components of the model. Merkur building platform was widely used with the addition of the 3D printed components [6] and some bought components including electrical parts. The pendulum itself was made with the help of a lathe.

Merkur is a Czech building platform [7]. Merkur components were used for the model to provide a uniform yet modular platform, simple to use in DIY projects. The Merkur platform provides a higher level of precision and rigidity of resulting construction compare to for example LEGO Mindstorm alternatives.

2.1.1 Chassis

The chassis of the model consists of four wheels, a platform and some fasteners. One axis with two wheels is not driven (1). The other two wheels are both driven by a DC motor (2a), (2b). The non-driven axis also holds a magnet for the magnetic rotary sensor to measure the rotation angle of the axis (3). Figure 2.3 shows chassis with its components. Figure 2.4 depicts the magnet sitting on the non-driven axis with the magnetic rotary sensor.

The non-driven axis was cut and tapered at one end according to the `wheel_axis.pdf` drawing, which is in the attachments. The taper was made so that the magnets could fit onto the axis. The magnet has a radial magnetic field for measuring purposes. Magnet dimensions are in the `magnet_drawing.pdf` in the attachments.

On the top of the chassis sits the main computing unit of the model,

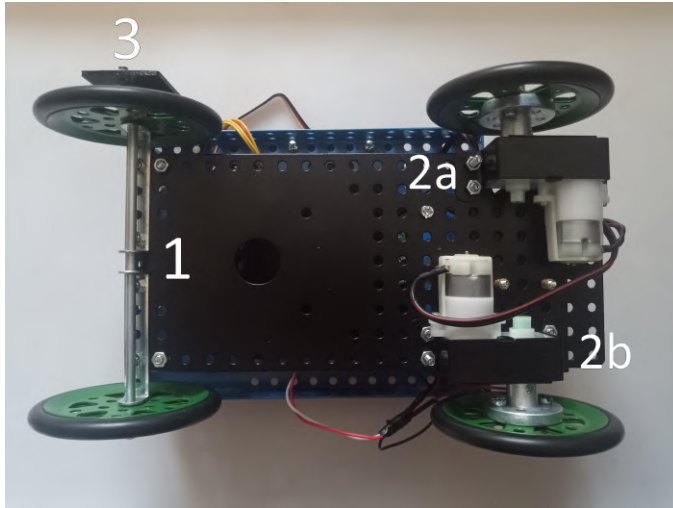


Figure 2.3: The chassis with its components; 1: non-driven axis, 2a, 2b: driven wheels, 3: magnet with magnetic rotary sensor

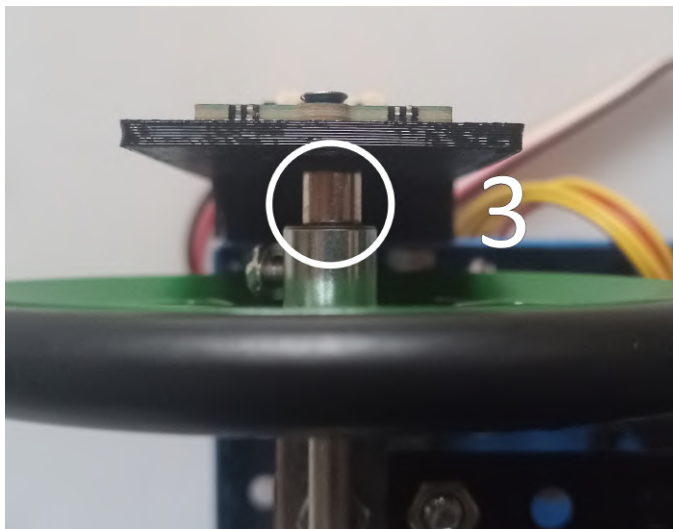


Figure 2.4: Closeup of the magnet of the lower magnetic rotary sensor

BeagleBone Blue (4). BeagleBone Blue [8] is mounted to the chassis using a 3D printed component. Next to the BeagleBone Blue are two 16850 LiPo batteries (5). Figure 2.5 shows mentioned components.

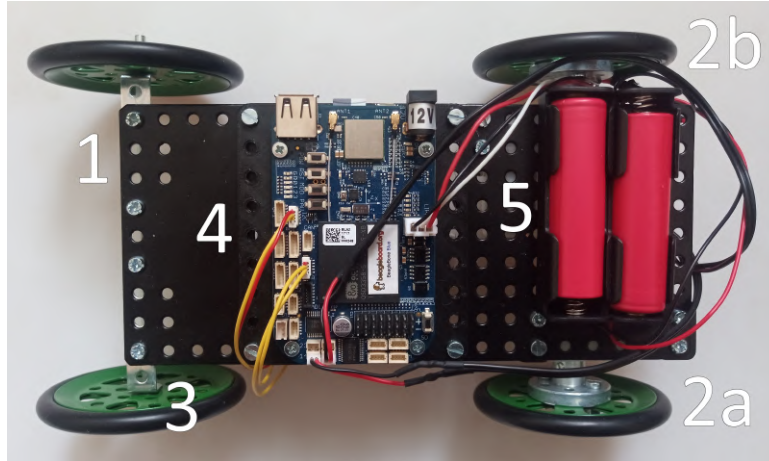


Figure 2.5: Top of the chassis with its components; 1: non-driven axis, 2a, 2b: driven wheels, 3: magnet, 4: BeagleBone Blue, 5: battery pack

2.1.2 Hood

The hood of the model sits on the top of the chassis. It is covering both the BeagleBone Blue and the battery pack. Multiple components are attached to the hood.

The lower magnetic rotary sensor (6) is attached using a 3D printed part. This magnetic rotary sensor measures the non-driven axis rotation. I2C multiplexer hub (7) is also attached to the hood with another 3D printed part. The 3D printed pendulum frame is attached to the hood (8).

The upper magnetic rotary sensor (9) is attached to the pendulum frame. This sensor measures the pendulum's rotation angle. Figure 2.6 shows a picture of the hood with its components. A closeup of the lower magnetic rotary sensor is in the Figure 2.7.

2.1.3 Pendulum

The pendulum was designed by me and manufactured by our faculty. A lathe and drilling machine were used to manufacture the pendulum. The pendulum itself is made out of brass. Brass is an alloy of copper and zinc. The pendulum axis drawing, rod drawing and weight drawing are in the `pendulum_drawing.pdf` in the attachments.

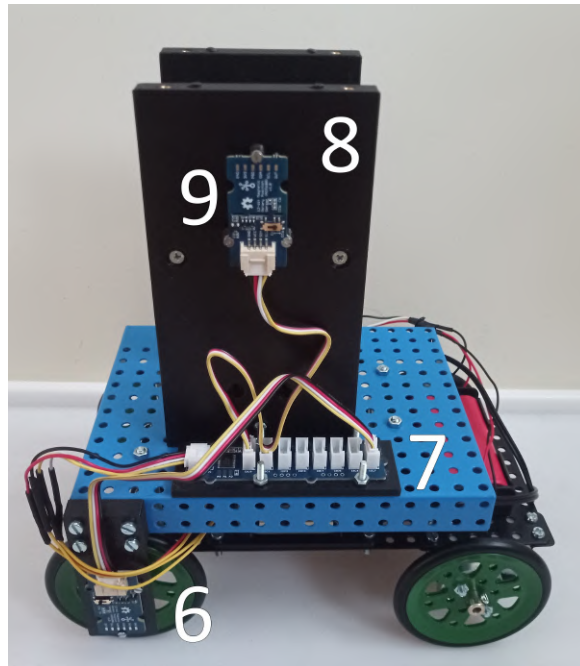


Figure 2.6: The hood with its components; 6: lower magnetic rotary sensor, 7: multiplexer hub, 8: pendulum frame, 9: upper magnetic rotary sensor

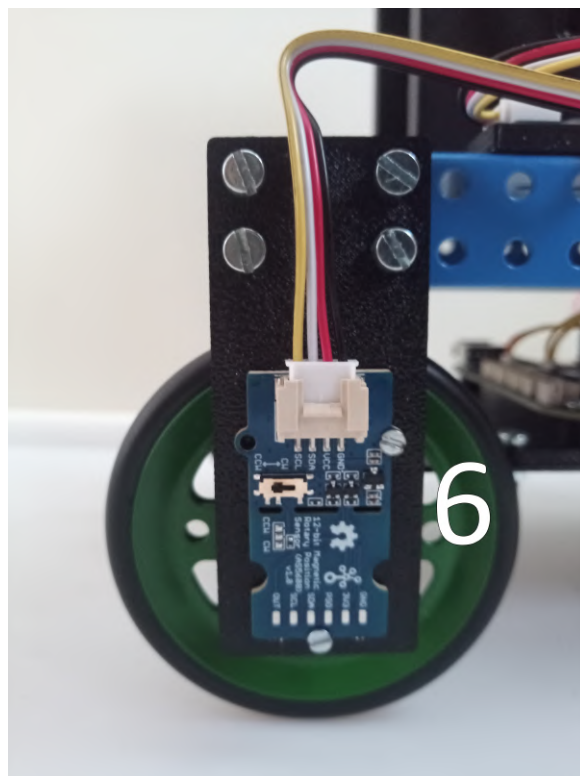


Figure 2.7: The lower magnetic rotary sensor

Figure 2.8 shows the pendulum (10) inserted between the two pendulum frames. Inside each pendulum frame is a roller skate bearing. The bearing dimension type is 608 of ABEC 9 or ABEC 7 quality. The pendulum sits in these bearings.

Figure 2.9 presents a closeup of the magnet sitting on the pendulum next to the upper magnetic rotary sensor.

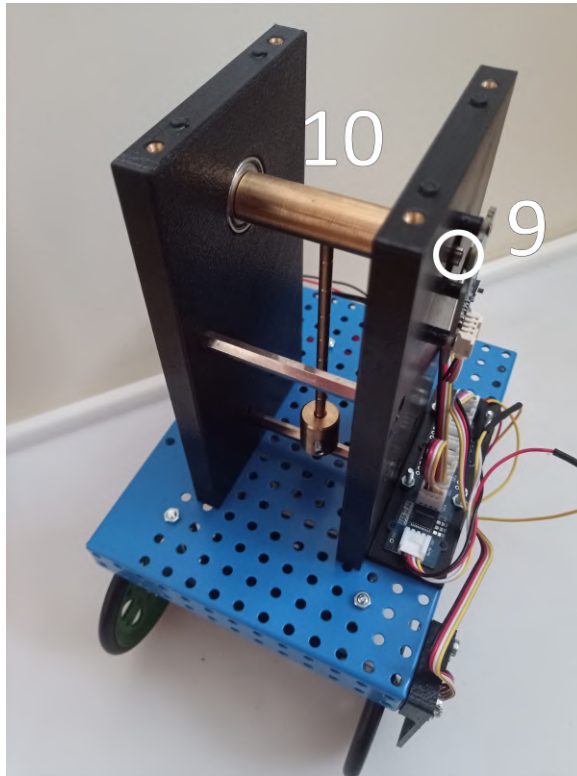


Figure 2.8: The pendulum frame with its components; 10: pendulum, 9: upper magnetic rotary sensor with a magnet in the circle

■ 2.2 Electrical components and sensors

This section describes the electrical components of the model. The electrical components were connected according to the electrical diagram depicted in Figure 2.10.

■ 2.2.1 BeagleBone Blue

The main computing unit of this model is BeagleBone Blue [8], which is a Linux-based computer suitable for embedded robotics and control-oriented applications. It uses Octavo OSD3358 microprocessor. We used WiFi, H-Bridge drivers, connectors for DC motors, power regulation, state-of-charge



Figure 2.9: The upper magnet closeup

LEDs for 2-cell LiPo and I2C bus from all its peripherals. BeagleBone Blue is supported by Matlab Simulink. Matlab Simulink support made this whole project suitable for students as a friendly laboratory model.

■ 2.2.2 Merkur DC motors

The model uses two Merkur DC motors as a source of motion. They come with fasteners and wheel carriers. Figure 2.3 shows these two motors as (2a) and (2b).

■ 2.2.3 LiPo batteries

Two 16850 LiPo batteries were used as a power source for the whole model. LiPo batteries are widely used in industry and academics. These batteries carry 3.7 volts and were used in series to power BeagleBone Blue.

Two single battery boxes were used to hold two 16850 LiPo batteries. Single battery boxes have two cables coming out of them. Put together in series they have four cables coming out of them. The two middle cables were connected and used as a balance connector. The balance connector is used by the BeagleBone Blue when charging to recharge both LiPo cells evenly [9].

DEMONSTRATOR ELECTRICAL DIAGRAM

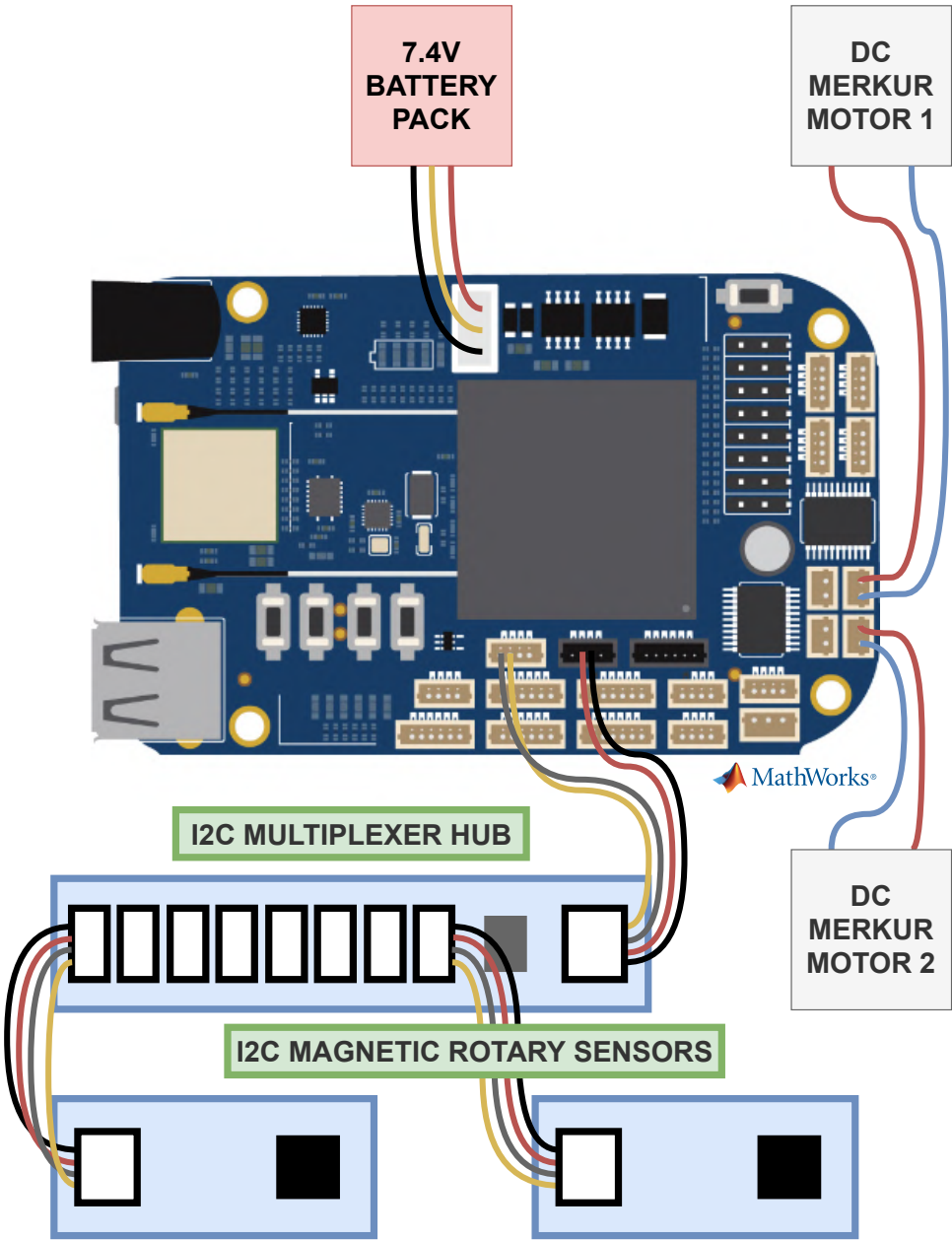


Figure 2.10: The model electrical diagram

■ 2.2.4 I2C bus sensors

Two Grove 12-bit Magnetic Rotary Position Sensor / Encoder (AS5600) [10] were used. These sensors operate on the I2C bus. The first sensor was used to measure the angle of rotation of the pendulum. The second sensor was used to measure the angle of rotation of the wheel axis.

These magnetic sensors have a non-changeable I2C address. This means that the user cannot read data from both sensors at the same time. So to use both sensors simultaneously, an I2C multiplexer hub was necessary.

Radial field magnets were used with the magnetic sensor to measure the angle of rotations as mentioned above. The magnets were glued and put onto the tapered parts of the axis and pendulum. Magnet dimensions are in the `magnet_drawing.pdf` file in the attachments.

Grove 8 Channel I2C Multiplexer/I2C Hub (TCA9548A) [11] was used to collect data from both magnetic sensors. It operates on the I2C bus. Two of the maximum eight channels are occupied by the sensors.

Six remaining channels are free for future extensions and modifications with new peripherals. Multiplexer was used to alternate between the two magnetic sensors to safely read the pendulum's rotation angle and wheel axis rotation angle.

■ 2.2.5 Accessories

LiPo batteries, that power BeagleBone Blue, need to be occasionally recharged. Therefore 12-volt power adapter with a barrel connector is used.

SD card is needed for functioning BeagleBone Blue. Section 3.1.1 describes how is the SD card used with BeagleBone Blue. The SD card needs at least 4GB of space.

A standard micro USB cable is needed for communication between the BeagleBone Blue and a computer. This cable needs two male connectors at each end. One male connector should be USB-A type for the computer at one end. The other male connector should be USB Micro-B for the BeagleBone Blue at the other end.

■ 2.3 3D printed components

Hardware connections between Merkur parts and electrical parts were done using 3D printed components [6]. All 3D printed components are listed below.

2.3.1 Pendulum frame

Two 3D printed frames are used to hold the pendulum and bearings in place. Both blocks have holes for bearings and the pendulum. Blocks are held together with three 50 mm M3 spacers.

The blocks are not identical although very similar. One block has three cylindrical protrusions with holes for M2 screws. With the help of these protrusions, one of the two magnetic sensors is held in place. The other block does not have any protrusions.

The sensor is screwed by its mounting holes with M2 screws inside the holes in the protrusions. With the sensor in place and magnet on the pendulum, the angle of the rotation of the pendulum can be measured.

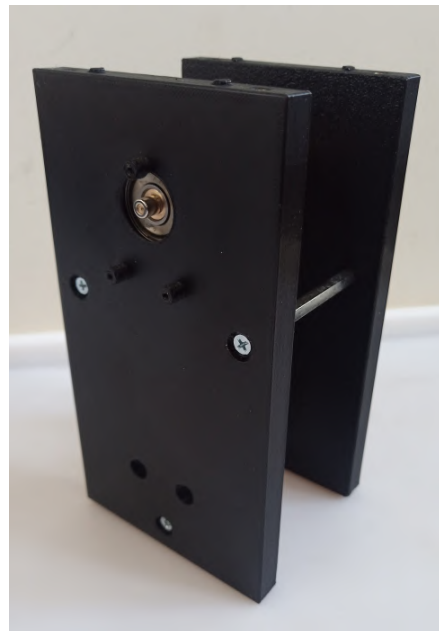
Both blocks have holes made for M3 inserts. Holes are made inside the side with the smallest area. With these inserts, blocks are screwed together with the hood.

It is also possible to rotate both blocks and the pendulum around the pendulum axis to create an inverted pendulum. The inverted pendulum can be used for the same education purposes as the non-inverted pendulum.

Figure 2.11 depicts both the 3D model from Fusion 360 software and the actual 3D print.



(a) : Pendulum frame in Fusion 360



(b) : Pendulum frame 3D print

Figure 2.11: Pendulum frame comparison

2.3.2 I2C multiplexer hub mounting

I2C multiplexer hub is screwed to the hood with the help of another 3D printed component. This component is to fit the multiplexer hub onto the hood and hold it in place. Merkur M3.5 screws and bolts are used in this case.

Figure 2.12 shows both the 3D model from Fusion 360 software and the actual 3D print.



(a) : Multiplexer hub mounting in Fusion 360



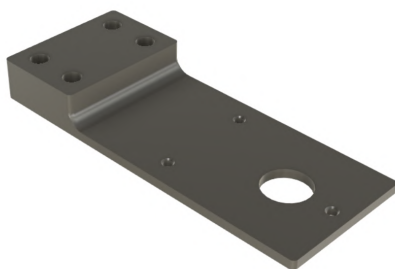
(b) : Multiplexer hub mounting 3D print

Figure 2.12: Multiplexer hub mounting comparison

2.3.3 Magnetic rotary position sensor mounting

The magnetic rotary position sensor mounting is used to hold the magnetic sensor in place at the side of the hood. The holder is placed directly side by side with the wheel axis, where one of the two magnets is placed. The sensor then measures the angle of the rotation of the wheel axis.

Figure 2.13 depicts both the 3D model from Fusion 360 software and the actual 3D print.



(a) : Magnetic rotary position sensor mounting in Fusion 360



(b) : Magnetic rotary position sensor mounting 3D print

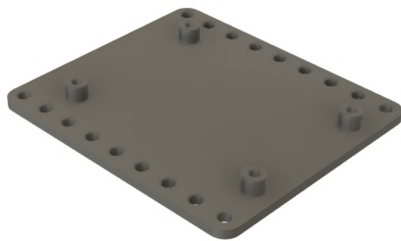
Figure 2.13: Magnetic rotary position sensor mounting comparison

2.3.4 BeagleBone Blue holder

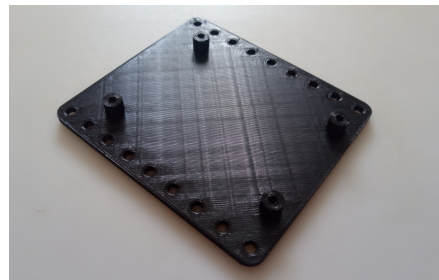
The BeagleBone Blue holder is used to hold BeagleBone Blue in place on the top of the chassis. The holder is designed specifically for BeagleBone Blue and Merkur components. By using this holder BeagleBone Blue is safely placed between the chassis and hood.

All its important ports are accessible and it will not move, when the user needs to pull some of the cables out of BeagleBone Blue.

Figure 2.14 shows both the 3D model from Fusion 360 software and the actual 3D print.



(a) : BeagleBone Blue holder in Fusion 360



(b) : BeagleBone Blue holder 3D print

Figure 2.14: BeagleBone Blue holder comparison

Chapter 3

Software

3.1 BeagleBone Blue setup

This section will cover all there is to do before the first Simulink code can be run on the BeagleBone Blue. This procedure is taken from the official Beagleboard Getting Started website [12] with the addition of my solutions.

3.1.1 Debian Image

BeagleBone Blue is a Linux-based computer with Debian operating system. To control it BeagleBone Blue with Matlab and Simulink, the right Debian image on the BeagleBone Blue must be used.

Matlab supports only the following version of Debian image: Debian 9.5 2018-10-07 4GB SD LXQT. Debian image for Beagle boards can be downloaded from Beagleboard official website [13].

The next step is writing the downloaded image to the SD card, which will be used with BeagleBone Blue. SD card programming utility like balenaEtcher [14] is needed for this task.

Connect the chosen SD card to the computer. All data from this SD card will be erased, so make a backup beforehand. Use the balenaEtcher application to write the downloaded Debian image to the SD card. After a couple of minutes, the write should be done.

The SD card can be now inserted into the powered-down BeagleBone Blue and then the power can be applied by connecting a USB cable or power adapter to the board.

3.1.2 Starting BeagleBone Blue

Before connecting to BeagleBone Blue, power needs to be provided either by a USB cable or a 12-volt adapter. There are two ways to connect to the BeagleBone Blue. Either by a USB cable or wirelessly by WiFi.

- MATLAB Coder
- Simulink Coder
- DSP System Toolbox
- Simulink Coder Support Package for BeagleBone Blue Hardware [16]

■ 3.2.1 Installing BeagleBone Blue Support Package

When installing the Simulink Coder Support Package for BeagleBone Blue Hardware Matlab installation window will appear.

Instruction on how to install this package and configure BeagleBone Blue for the use with Simulink will be on the screen. The installation can connect the BeagleBone Blue to a chosen WiFi network, which is practical.

The BeagleBone Blue can be always connected to the WiFi network later. This article describes connecting to the BeagleBone Blue remotely via WLAN [17].

■ 3.2.2 I2C Simulink library fix

In this project, the I2C bus is used. There is a defect in the Simulink source codes for I2C blocks. Simulink with an I2C block from BeagleBone Blue library will not run unless this issue is fixed.

To fix this defect the user needs to follow a few steps that are described below. Run the following command in the Matlab command line.

```
edit(fullfile(codertarget.bbblue.internal.getSpPkgRootDir,
'src','MW_I2C.c'))
```

A file named MW_I2C.c should open. Scroll down to see lines 77 and 78. It should look like this.

```
76 #include <fcntl.h>
77 #include <linux/i2c-dev.h>
78 #include <sys/ioctl.h>
79 #include <sys/types.h>
```

Now insert the following lines between lines 77 and 78.

```
1 #ifndef I2C_M_RD
2     #include <linux/i2c.h>
3 #endif
```

Now the lines from 77 to 81 should look like this.

the user has installed all the Matlab products described above 3.2.

The best way to ensure the `beagleboneblue()` command is active all the time is to put it to the Initialization function (InitFcn) from Model Properties Callbacks. Use the right mouse button and then click Model Properties.

A small window should open. In this window find Callbacks. In the Model, callbacks select InitFcn and insert it in the Model initialization function.

The IP address will be different for the USB connection and for the WiFi connection as mentioned above. This setting will run the command every time a simulation is run.

Figure 3.1 shows the InitFcn window in Simulink with the `beagleboneblue()` command for connection via WiFi.

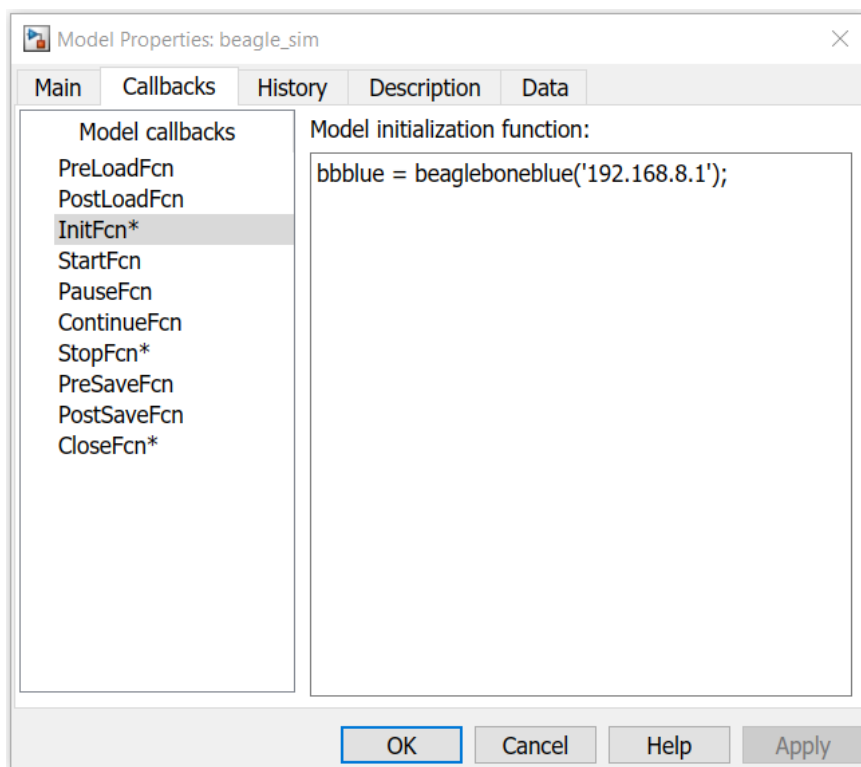


Figure 3.1: Model properties: Callbacks: InitFcn

Now go back to the blank Simulink model and click the right mouse button again. This time select Model Configuration Parameters. A small window should open.

In the window select Hardware Implementation and open Target hardware resources. From groups choose Board Parameters. The Device Address should be either 192.168.7.2 for the USB connection or 192.168.8.1 for the WiFi

connection. Username should be `debian`. Password should be `temppwd`.

Figure 3.2 shows the Target hardware resources setting for the WiFi connection.

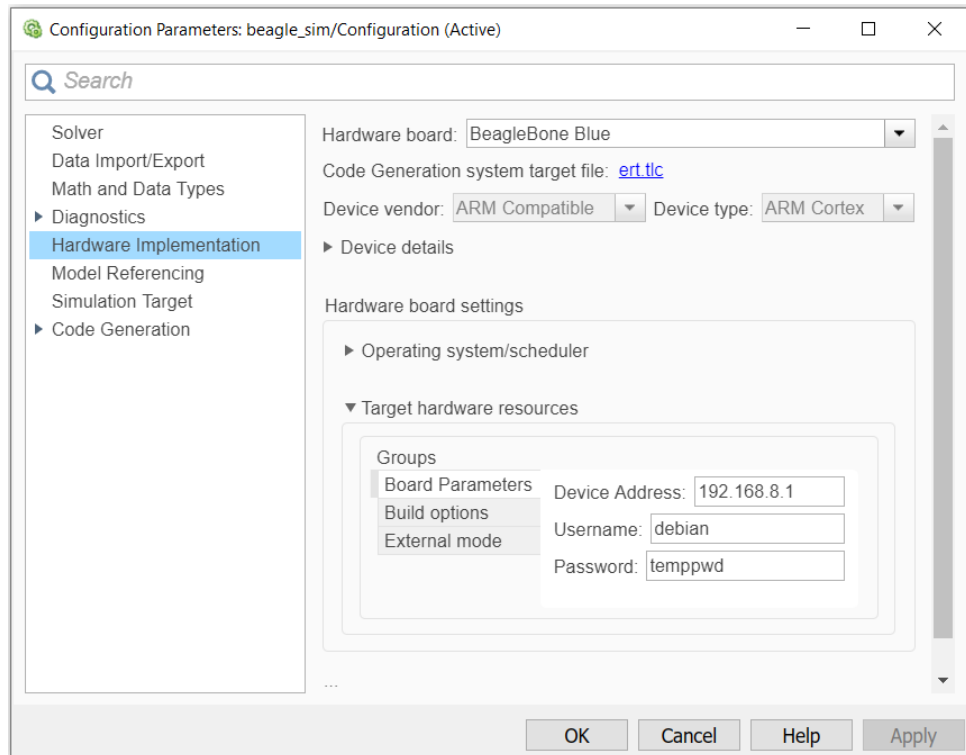


Figure 3.2: Model properties: Callbacks: InitFcn

If all these steps are implemented the Simulink model should run. Try putting a LED block and a Button block in the Basic section from the Support Package for BeagleBone Blue into the simulation.

Figure 3.3 shows these two blocks connected.

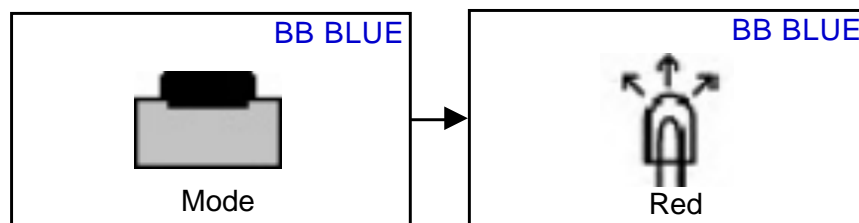


Figure 3.3: Button block connected to the LED block

Now press Monitor & Tune in the HARDWARE tab. The model should start shortly on the BeagleBone Blue. Pressing the MOD button should light up the red LED on the BeagleBone Blue while the Simulation is running.

■ 3.3.4 Using Simulink blocks to control the model

This section describes what blocks are used to control our model.

■ Motor control

Use the DC motor block from the Actuators section in the Simulink Coder Support Package for BeagleBone Blue Hardware to control any DC motors connected to the BeagleBone Blue. The BeagleBone Blue has four channels for DC motors so choose the right Motor index in the DC motor block parameters.

This block takes only `uint8` integer as input so be sure to use the Data Type Conversion block before the DC motor block. Input ranges from 100 to -100 power, 100 meaning maximum power in one direction, 0 meaning no power and -100 meaning maximum power in the other direction.

Figure 3.4 shows usage of the DC motor block.

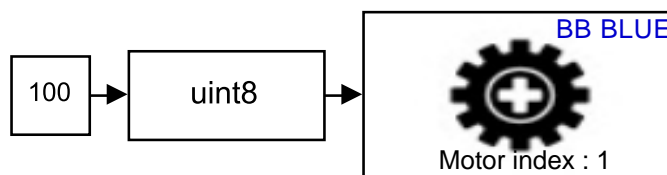


Figure 3.4: DC motor block with maximum input power

Motors need sufficient power for them to rotate, so the battery pack or the 12-volt adapter needs to be connected.

■ Reading data from the I2C bus

Two I2C blocks from the Communication section in the Simulink Coder Support Package for BeagleBone Blue Hardware are used to read data from the I2C bus and write data into the I2C bus.

This model uses the I2C multiplexer hub to communicate with both magnetic sensors, the one that measures pendulum rotation and the other one that measures wheel rotation.

I2C multiplexer hub has its I2C address. This address is hardware selectable. The default address is 0x70. The Grove magnetic sensors unfortunately do not have a selectable address. Their address is 0x36.

That is why the multiplexer is needed to communicate with both magnetic sensors. Without it, there would always be a collision between these two sensors with the same address.

CONTROL REGISTER BITS								COMMAND
B7	B6	B5	B4	B3	B2	B1	B0	
X	X	X	X	X	X	X	0	Channel 0 disabled
							1	Channel 0 enabled
X	X	X	X	X	X	X	0	Channel 1 disabled
							1	Channel 1 enabled
X	X	X	X	X	X	X	0	Channel 2 disabled
							1	Channel 2 enabled
X	X	X	X	X	X	X	0	Channel 3 disabled
							1	Channel 3 enabled
X	X	X	X	X	X	X	0	Channel 4 disabled
							1	Channel 4 enabled
X	X	X	X	X	X	X	0	Channel 5 disabled
							1	Channel 5 enabled
X	X	X	X	X	X	X	0	Channel 6 disabled
							1	Channel 6 enabled
0	X	X	X	X	X	X	0	Channel 7 disabled
							1	Channel 7 enabled
0	0	0	0	0	0	0	0	No channel selected, power-up/reset default state

Figure 3.5: I2C multiplexer hub: command byte definition [1]

■ Using I2C multiplexer hub to read both magnetic sensors

The I2C multiplexer hub [1] has eight channels. Two channels are occupied by the magnetic sensors.

To switch between the two magnetic sensors the user needs to write one byte (8 bits) to the multiplexer address. In our case, the address is 0x70.

The byte written to the multiplexer address affects which channels of the multiplexer are enabled and which are disabled.

For example, if we would write 0x01 to the multiplexer, only channel 0 would be enabled. All other channels would be disabled.

Figure 3.5 shows how the byte sent to the multiplexer address affects which channel is enabled or disabled.

Figure 3.6 shows I2C write block with the right configuration, to write 0x01 data to the 0x70 address of the multiplexer.

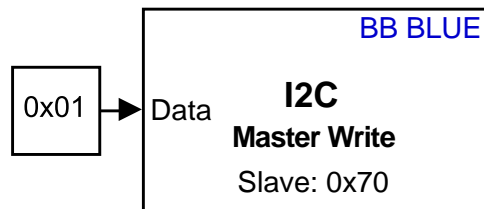


Figure 3.6: I2C write block with the multiplexer address

■ Sequence of using the I2C blocks to read both magnetic sensors

Each magnetic sensor is connected to one multiplexer channel. The user needs to enable one channel and read the magnetic sensor connected to the enabled channel.

Then disable the first channel and enable the other channel and read the magnetic sensor that is connected to the new enabled channel.

The order of this writing and reading from the I2C bus is crucial. If the user would switch only one read or write it would not work.

Let us say that one magnetic sensor is connected to the channel 0 and the second magnetic sensor is connected to the channel 7.

The sequence of writing and reading goes as follows.

1. Write 0x01 data to the 0x70 address. This tells the multiplexer to enable only channel 0.
2. Read data from the 0x36 address. This data represents the angle from the magnetic sensor and it is described in detail in the next sections.
3. Write 0x80 data to the 0x70 address. This tells the multiplexer to enable only channel 7.
4. Read again data from the 0x36 address. This data represents the angle from the other magnetic sensor.

■ Priority and execution order of the Simulink blocks

Simulink blocks have their priority. This priority can be set. Use right-click on the block and choose Properties. Priority is in the lower half of the opened window.

The execution order of all the blocks in the simulation is affected by the block priority. This execution order is what enables us to use the blocks in the right sequence.

A lower priority number means that this block is going to be executed earlier. The I2C write block to control enabled multiplexer channels needs to have a lower priority number than the I2C read block that reads the angle data value.

Two I2C read blocks are used to read the angle value data from the magnetic sensor. This is described in the next section, but the priority must be set like this.

1. first I2C write block to enable the first channel - priority 1

2. first I2C read block - priority 2
3. second I2C read block - priority 3
4. second I2C write block to enable the second channel - priority 4
5. third I2C read block - priority 5
6. fourth I2C read block - priority 6

Figure 3.7 shows the Priority setting. In this example, the priority is set to 1.

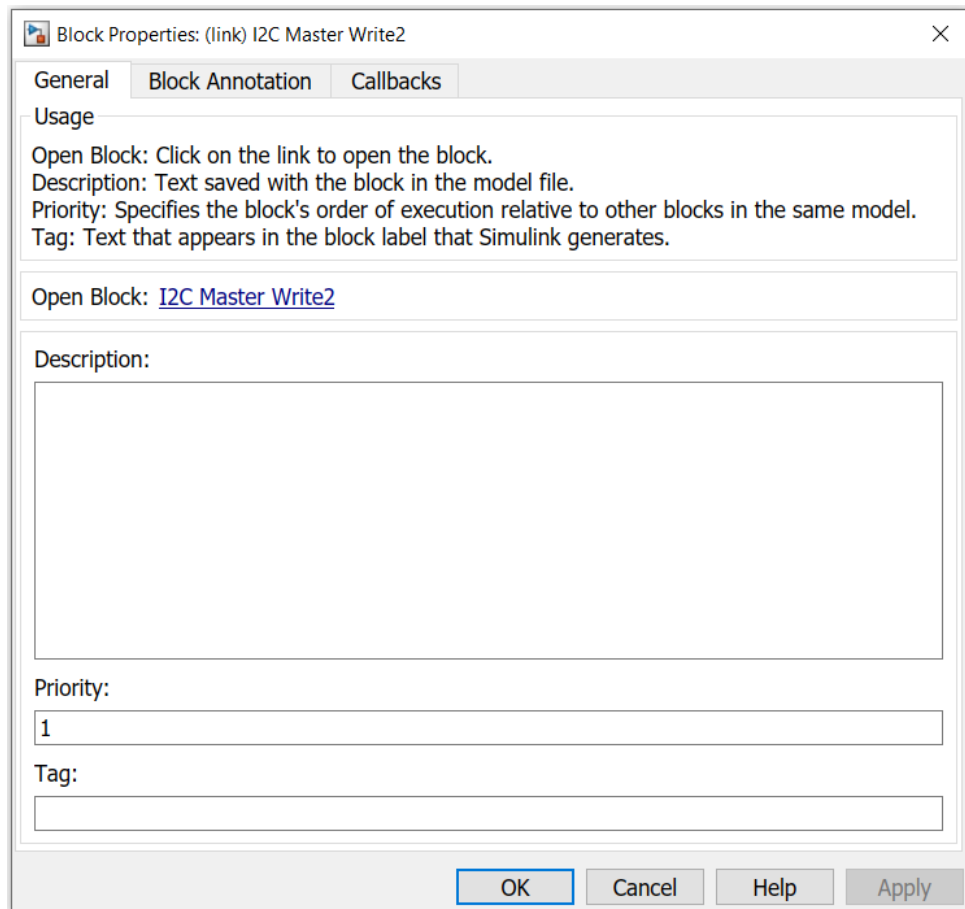


Figure 3.7: Priority setting in the Block Properties

■ Reading from one magnetic sensor

Reading data from magnetic sensors requires a few steps. The steps are taken from the magnetic sensor datasheet [2].

Firstly the right multiplexer channel must be enabled as described above. Then the user must read the data from the right address and the right register

two times.

The user can read one or more bytes but the magnetic sensor that we used measures a 12-bit value representing the angle. So the user needs to read one byte two times and then combine these two bytes to get the actual angle. Figure 3.8 shows the exact blocks from the Simulink model for this reading procedure.

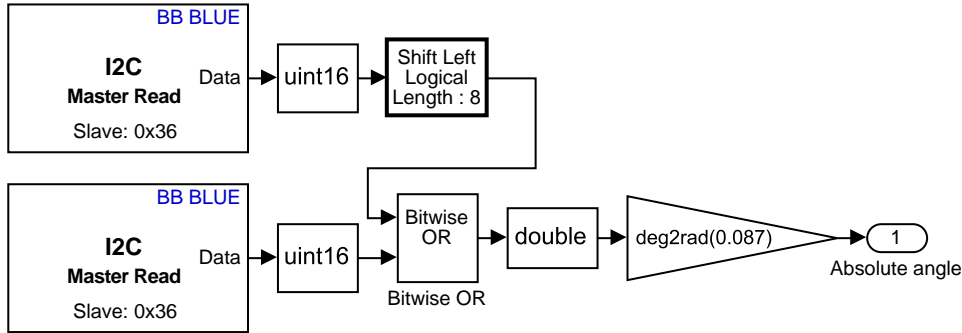


Figure 3.8: Simulink blocks to read the angle value from the magnetic sensor

First, the user reads one byte from the magnetic sensor at address 0x36 and the 0x0C register. This register holds the four upper bits of the angle value.

Figure 3.9 shows the two registers that we use to read angle value.

Output Registers							
0x0C	RAW ANGLE	R					RAW ANGLE(11:8)
0x0D		R	RAW ANGLE(7:0)				

Figure 3.9: Angle registers [2]

Figure 3.10 shows I2C Read block parameters with the corresponding register address.

Another byte with another I2C read block must be read but from a different register. Register 0x0D holds the lower eight bits of the angle value. Same parameters as in figure 3.10 except the Slave register address is changed to 0x0D.

Both values are converted to uint16 type. The upper value is logically shifted left by 8. Next, the Bitwise OR block is used to combine both values. The output is the 12-bit angle value.

To convert the 12-bit value to radians, two conversions are used. First is multiplying the 12-bit value by a constant as shown in equation 3.1 where

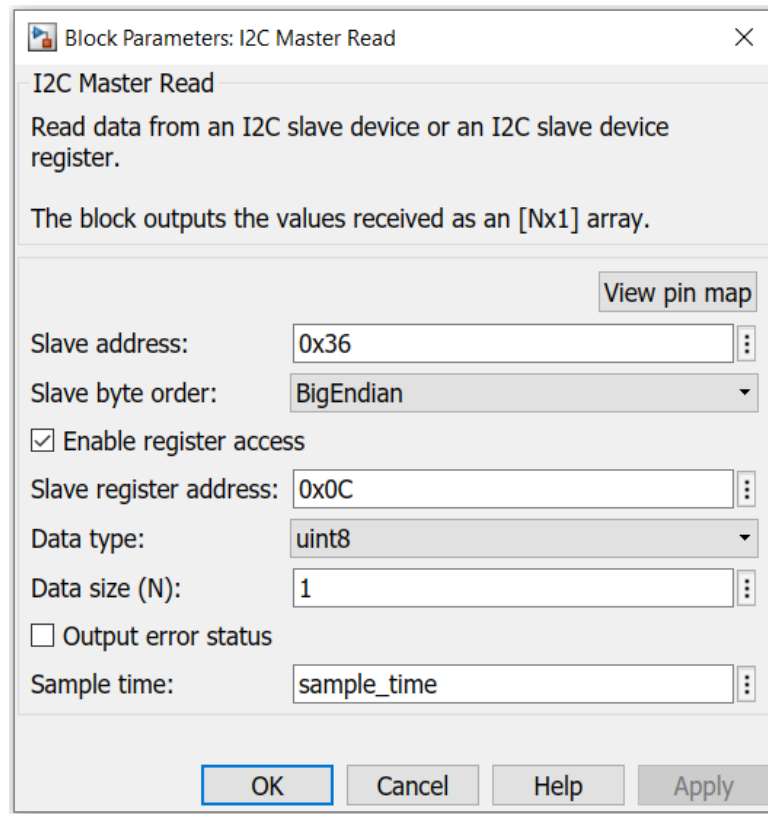


Figure 3.10: I2C Read block parameters

A_{BIT} is the 12-bit angle value and A_D is the angle in degrees.

$$A_D = A_{BIT} \cdot \frac{360}{2^{12}} = A_{BIT} \cdot 0.087 \quad (3.1)$$

The last adjustment is converting the degrees to radians. The last two conversions of the 12-bit value are done in the same block as shown in Figure 3.8.

3.3.5 Getting the relative angle to the start of the simulation

The computed angle in radians is just an absolute angle value. A relative angle to the start of the simulation is needed to control the model. This is achieved by using another few blocks and a Matlab function.

Figure 3.11 shows the Simulink blocks for converting between the absolute and relative angle.

The first block after Absolute angle input is the Unwrap block from the DPS System Toolbox. This block is crucial as it adds 2π radians to the input value of the block when the difference between the two last input values is bigger than chosen tolerance.

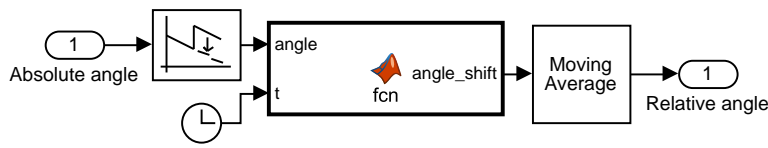


Figure 3.11: Absolute to relative angle block sequence

Figure 3.12 shows the Unwrap block parameters with the chosen tolerance as π radians.

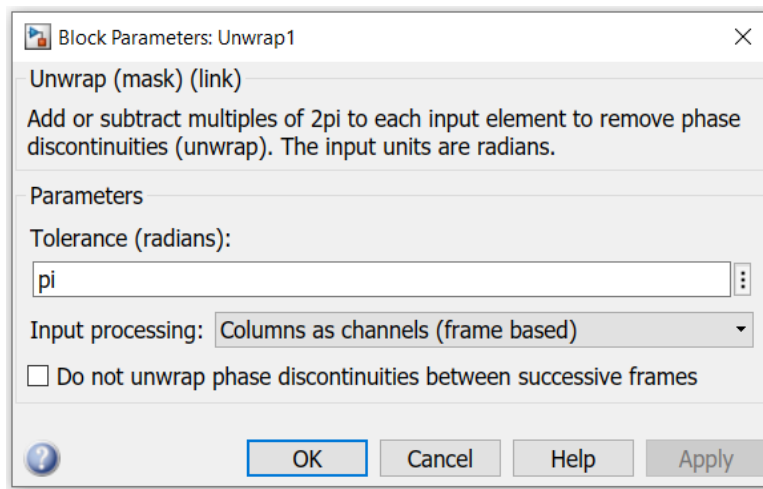


Figure 3.12: Unwrap block parameters with the π radians tolerance

The next block is a Matlab function. It has two inputs. One input is the unwrapped angle, second input is the Simulation time from the Clock block.

This Matlab function stores the input value at the start of the Simulation and subtracts it from the input value throughout the simulation.

This way the output value is always relative to the start of the simulation. Listing 3.1 shows this function.

Listing 3.1: Matlab function for subtracting initial value of the input

```

1 function angle_shift = fcn(angle, t)
2     persistent init_angle
3     if isempty(init_angle)
4         init_angle = 0;
5     end
6     if t == 0
7         init_angle = angle;
8     end
9     angle_shift = angle - init_angle;

```

Moving Average block is the last block used in the sequence. It is also from

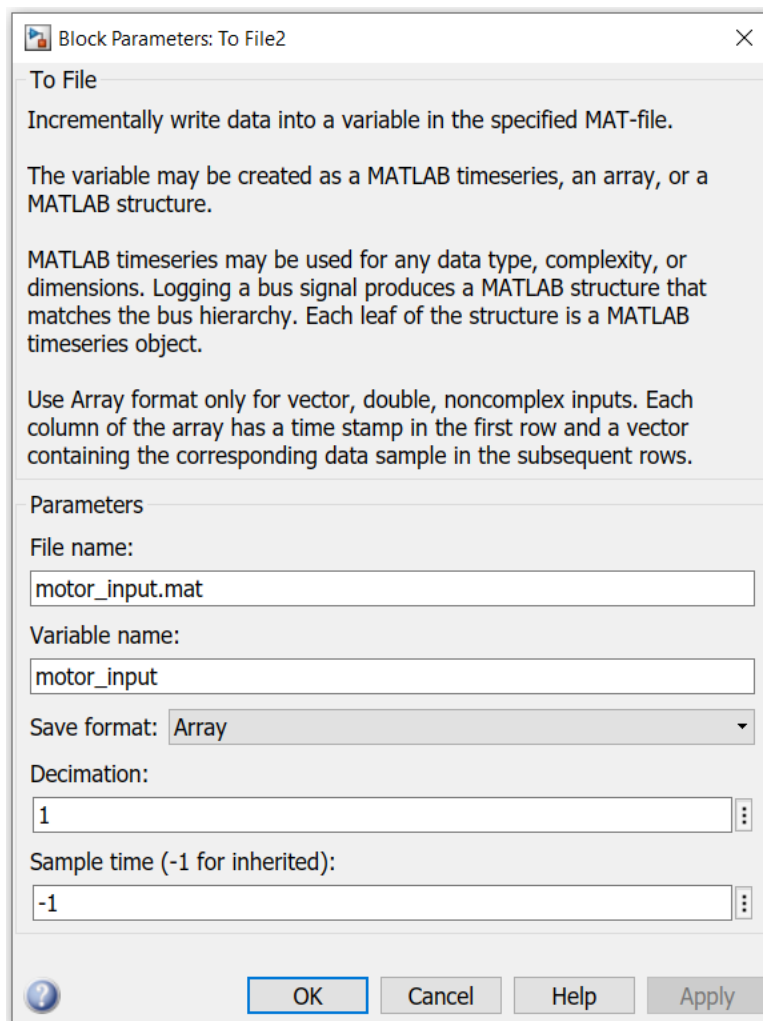


Figure 3.13: To File block parameters

preventing the Checksum error from happening at the start of the next session.

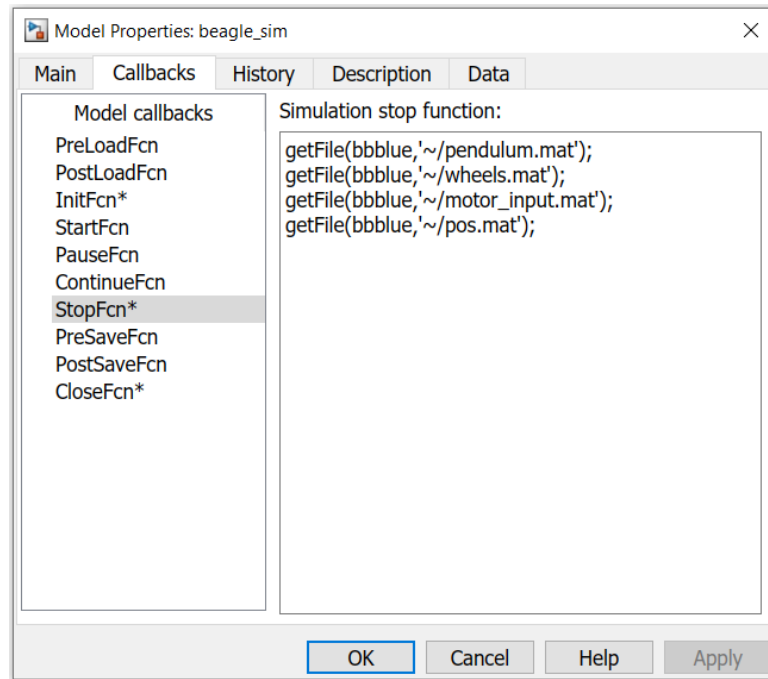


Figure 3.14: StopFcn Callback with the `getFile()` command

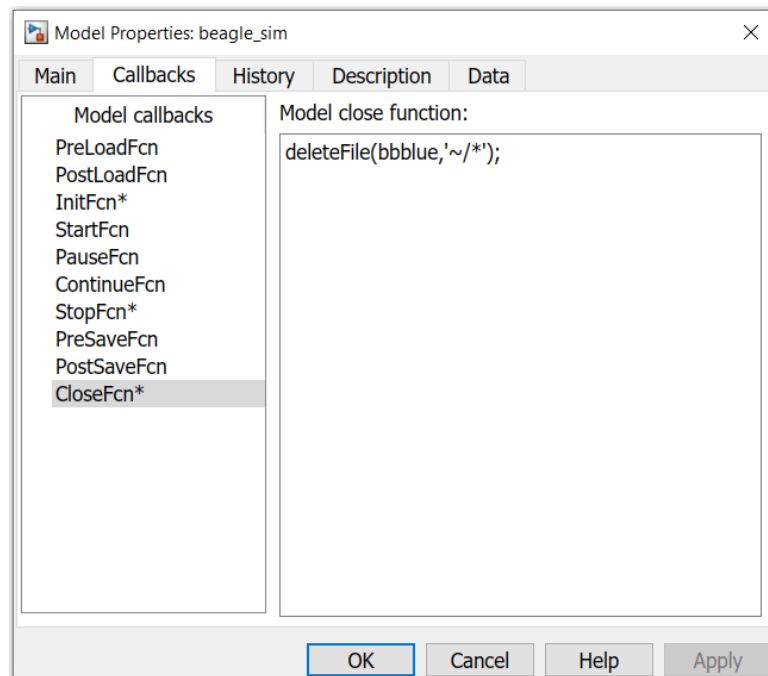


Figure 3.15: CloseFcn Callback with the `deleteFile()` command

Chapter 4

Control

4.1 Modelling dynamic system

This section describes the physical modelling of the laboratory model using Lagrange's equations [20].

Figure 4.1 shows the necessary parameters to describe the laboratory model as a dynamic system.

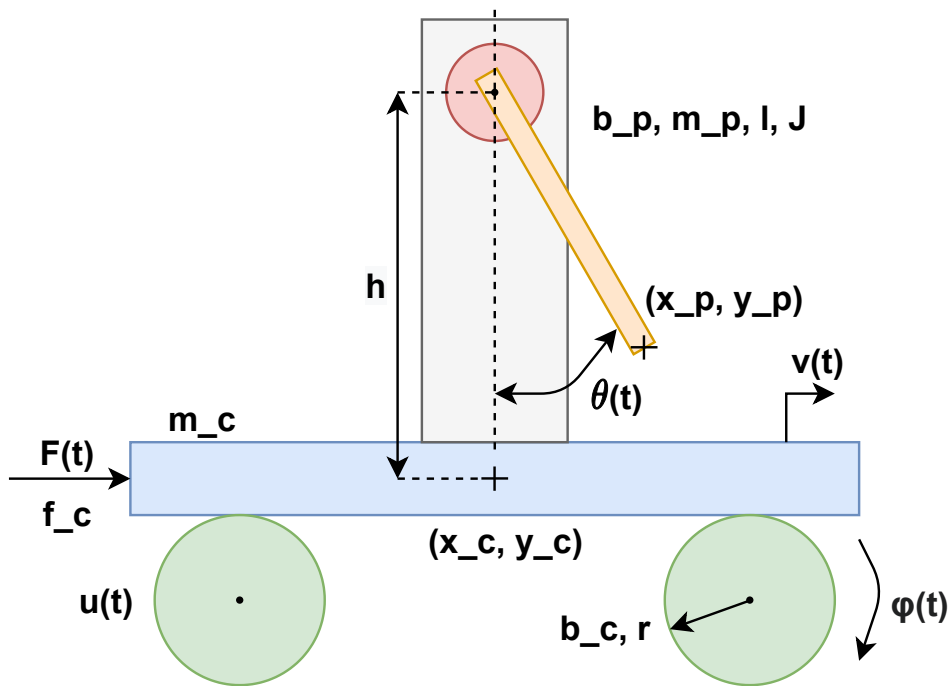


Figure 4.1: The model parameters

Two coordinates $\begin{bmatrix} x(t) \\ \theta(t) \end{bmatrix}$ are used to describe the full state of the model, where $x(t)$ is displacement of the whole model and $\theta(t)$ is pendulum's angle.

The model is divided into two components. The first is the cart and the second is the pendulum.

Parameters belonging to the cart component of the model are listed below:

- b_c viscous friction of the cart;
- m_c mass of the cart;
- r radius of the cart wheels;
- f_c gain from motors to generated force.

The force $F(t)$ generated by the motors input voltage $u(t)$ is described by the equation below.

$$F(t) = f_c \cdot u(t) \quad (4.1)$$

The rotation angle of the wheels is $\varphi(t)$. Using the radius of the wheels we get displacement of the model as follows

$$\varphi(t) \cdot r = x(t). \quad (4.2)$$

Wheel slip is neglected.

Derive the displacement by time we get velocity of the model

$$\frac{dx(t)}{dt} = v(t). \quad (4.3)$$

Parameters belonging to the pendulum component of the model are listed below:

- b_p viscous friction of the pendulum;
- m_p mass of the pendulum;
- l length of the pendulum;
- J moment of inertia of the pendulum;
- h height of the pendulum.

As mentioned above, the pendulum angle is $\theta(t)$.

■ 4.1.1 Cart energy and dissipation

Coordinates of the cart are

$$x_c = x, \quad (4.4)$$

$$y_c = 0. \quad (4.5)$$

Velocity is time derivation of the coordinates as follows

$$\dot{x}_c = \dot{x} = v, \quad (4.6)$$

$$\dot{y}_c = 0. \quad (4.7)$$

Translational energy of the cart is

$$T_{tc}^* = \frac{1}{2}(m_c + m_p)(\dot{x}_c^2 + \dot{y}_c^2). \quad (4.8)$$

Dissipation acting on the cart is

$$D_c = \frac{1}{2}b_c\dot{x}_c^2. \quad (4.9)$$

4.1.2 Pendulum energy and dissipation

Coordinates of the pendulum are

$$x_p = x_c + l \sin(\theta), \quad (4.10)$$

$$y_p = y_c + h - l \cos(\theta). \quad (4.11)$$

Velocity is time derivation of the coordinates as follows

$$\dot{x}_p = \dot{x} + l \cos(\theta)\dot{\theta}, \quad (4.12)$$

$$\dot{y}_p = l \sin(\theta)\dot{\theta}. \quad (4.13)$$

Potential energy of the pendulum where g is gravitational acceleration is

$$V_p = m_p g y_p. \quad (4.14)$$

Translational energy of the pendulum is

$$T_{tp}^* = \frac{1}{2}m_p(\dot{x}_p^2 + \dot{y}_p^2). \quad (4.15)$$

Rotational energy of the pendulum is

$$T_{rp}^* = \frac{1}{2}J\dot{\theta}^2. \quad (4.16)$$

Dissipation acting on the pendulum is

$$D_p = \frac{1}{2}b_p\dot{\theta}^2. \quad (4.17)$$

4.1.3 Lagrange's equations

The Lagrange's equations will look like this.

$$L = T_{tc}^* + T_{tp}^* + T_{rp}^* - V_p \quad (4.18)$$

$$D = D_c + D_p \quad (4.19)$$

Derivation by time and by coordinates is in place. The two final equation will look like this.

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x} = - \frac{\partial D}{\partial \dot{x}} + f_c F \quad (4.20)$$

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = - \frac{\partial D}{\partial \dot{\theta}} \quad (4.21)$$

After all the derivation the equations look like this.

$$m_p \ddot{x} + \ddot{\theta} l m_p \cos(\theta) - l m_p \sin(\theta) \dot{\theta}^2 + \ddot{x}(m_c + m_p) = -b_c \dot{x} + F f_c \quad (4.22)$$

$$J \ddot{\theta} + \ddot{\theta} l^2 m_p + \ddot{x} l m_p \cos(\theta) + g l m_p \sin(\theta) = -b_p \dot{\theta} \quad (4.23)$$

Substitution is made for simplicity. These are the four states of the model.

$$x_1 = x \quad (4.24)$$

$$x_2 = \theta \quad (4.25)$$

$$x_3 = \dot{x} \quad (4.26)$$

$$x_4 = \dot{\theta} \quad (4.27)$$

Input $u(t)$ is substituted for u .

$$u = u(t) \quad (4.28)$$

Equations 4.22 and 4.23 are second-order differential equations. First-order equations are needed for linearisation purposes.

So the equations are rewritten like this.

$$\dot{x}_1 = x_3$$

$$\dot{x}_2 = x_4$$

$$\begin{aligned} \dot{x}_3 = & [u J f_c - J b_c x_3 + u f_c l^2 m_p - b_c l^2 m_p x_3 \\ & + l^3 m_p^2 x_4^2 \sin(x_2) + g l^2 m_p^2 \cos(x_2) \sin(x_2) \\ & + J l m_p x_4^2 \sin(x_2) + b_p l m_p x_4 \cos(x_2)] / [J m_c \\ & + 2 J m_p + 2 l^2 m_p^2 + l^2 m_c m_p - l^2 m_p^2 \cos(x_2)^2] \end{aligned} \quad (4.29)$$

$$\begin{aligned} \dot{x}_4 = & -[b_p m_c x_4 + 2 b_p m_p x_4 + 2 g l m_p^2 \sin(x_2) \\ & + l^2 m_p^2 x_4^2 \cos(x_2) \sin(x_2) + u f_c l m_p \cos(x_2) \\ & - b_c l m_p x_3 \cos(x_2) + g l m_c m_p \sin(x_2)] / [J m_c \\ & + 2 J m_p + 2 l^2 m_p^2 + l^2 m_c m_p - l^2 m_p^2 \cos(x_2)^2] \end{aligned}$$

With these four first-order equations, a linear model of the laboratory model is made.

4.1.4 Linearisation

The model is linearised in an operating point where all the states and input are zero.

$$x_1 = 0, \quad x_2 = 0, \quad x_3 = 0, \quad x_4 = 0, \quad u = 0 \quad (4.30)$$

General matrices A and B are

$$A = \begin{bmatrix} \frac{\partial \dot{x}_1}{\partial x_1} & \frac{\partial \dot{x}_1}{\partial x_2} & \frac{\partial \dot{x}_1}{\partial x_3} & \frac{\partial \dot{x}_1}{\partial x_4} \\ \frac{\partial \dot{x}_2}{\partial x_1} & \cdots & \cdots & \frac{\partial \dot{x}_2}{\partial x_4} \\ \vdots & \ddots & \ddots & \vdots \\ \frac{\partial \dot{x}_4}{\partial x_1} & \cdots & \cdots & \frac{\partial \dot{x}_4}{\partial x_4} \end{bmatrix}, B = \begin{bmatrix} \frac{\partial \dot{x}_1}{\partial u} \\ \vdots \\ \frac{\partial \dot{x}_4}{\partial u} \end{bmatrix}. \quad (4.31)$$

As output of the linearised model displacement of the cart x_1 and pendulum rotation x_2 is choosed. Matrices C and D are as follows

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \quad (4.32)$$

4.2 Model identification

Three identification experiments were conducted. All the figures below show a comparison between the linear model and a real hardware model.

First is pendulum identification. Figure 4.2 shows the pendulum response to initial angle 0.0964 radians. This experiment helped to identify the pendulum parameters as mentioned above.

The real hardware pendulum, which is rendered in blue colour, stops its movement sooner. This is because the friction of a real pendulum is not linear therefore the linear model can not perfectly describe the real pendulum and continues to oscillate longer.

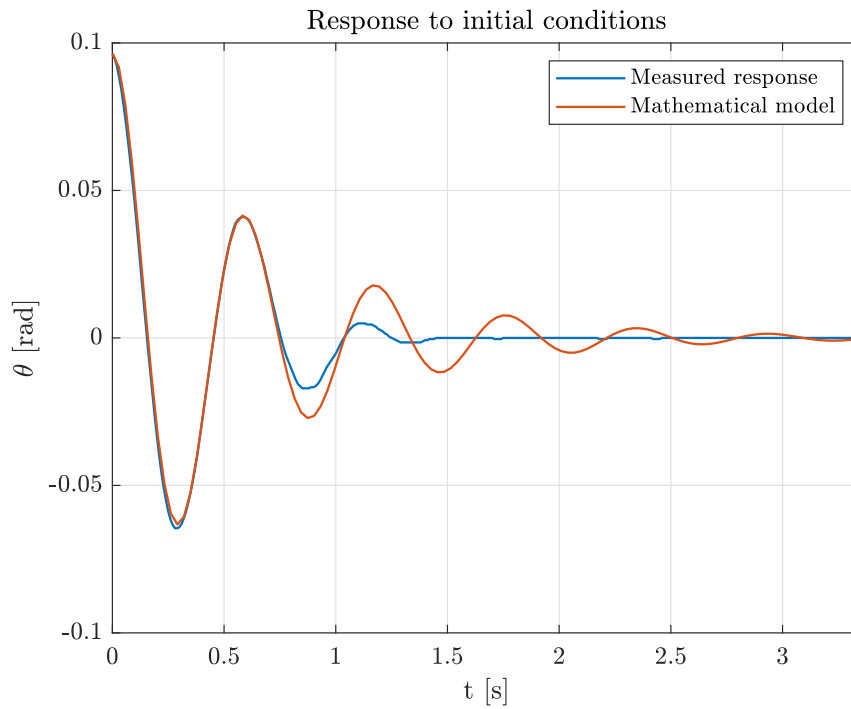


Figure 4.2: Pendulum identification: response to initial condition

The second experiment was a step response to maximum power 100 of the cart with a locked pendulum. Figure 4.3 shows this experiment.

This experiment helped to identify the cart parameters mentioned above. The response is almost linear, but in the beginning, we can see an exponential part of the response.

The last experiment was a step response of the model to the maximum power of 100 with a free pendulum. Figures 4.4 and 4.5 show the pendulum angle and cart displacement.

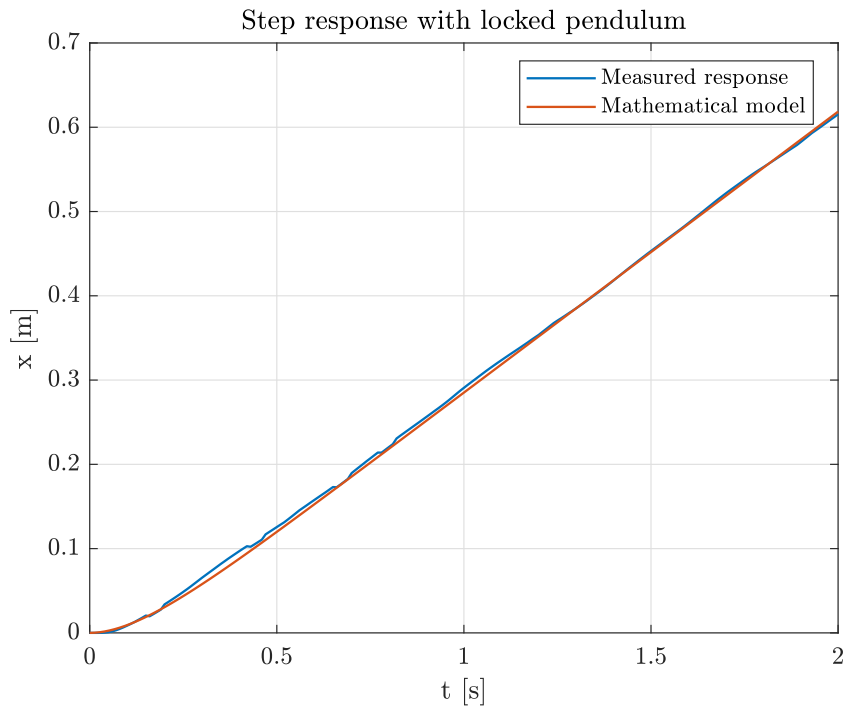


Figure 4.3: Cart identification: step response

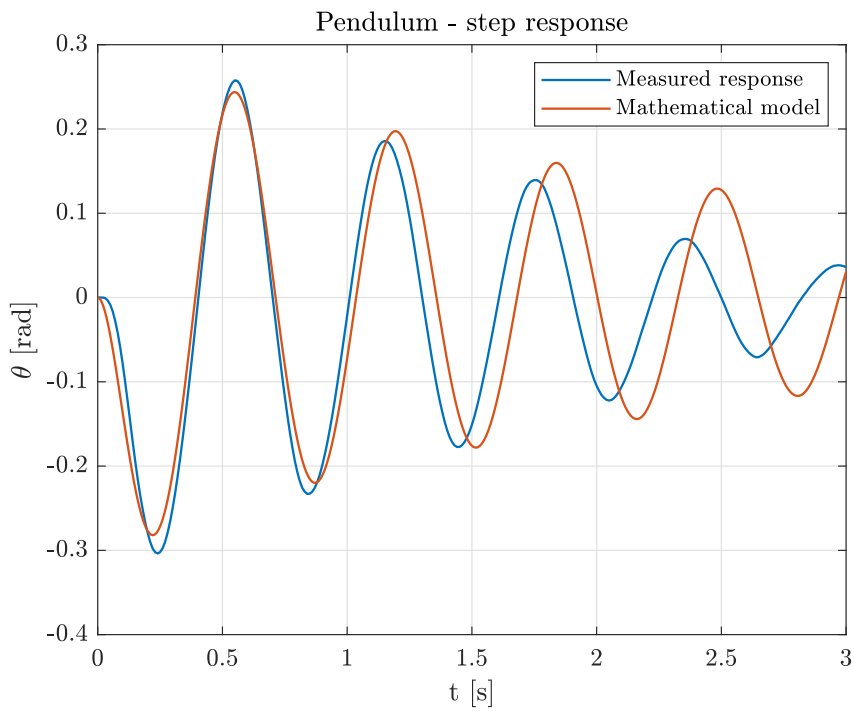


Figure 4.4: Model identification: step response: pendulum

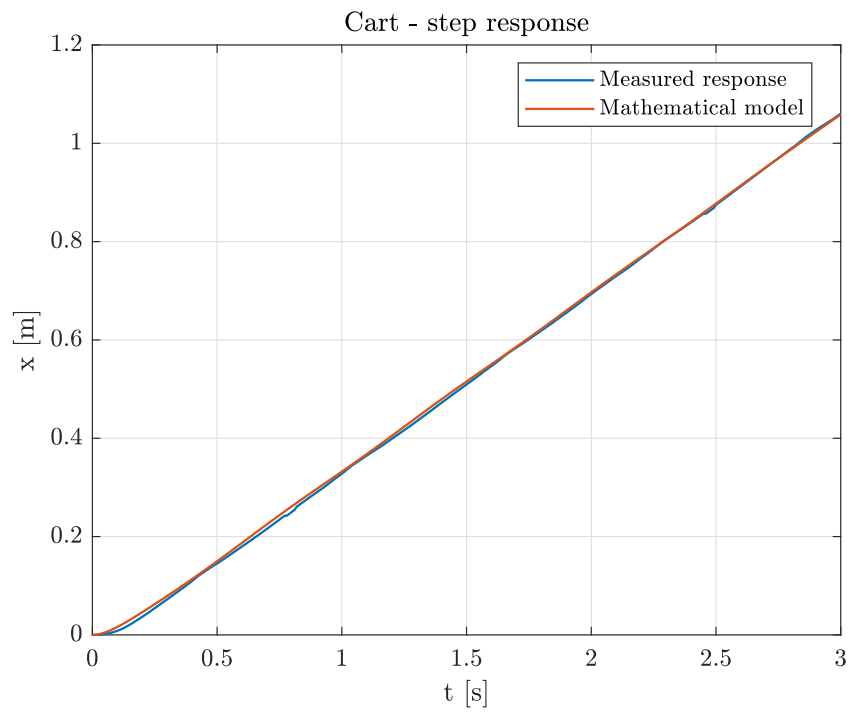


Figure 4.5: Model identification: step response: cart

This was the main experiment that helped identify the dynamic model parameters. Pendulum height was measured with a standard meter. Gravitational acceleration is well known.

- $b_c = 14.068 \text{ kg m}^2 \text{ s}^{-1}$
- $b_p = 1.822 \cdot 10^{-4} \text{ kg m}^2 \text{ s}^{-1}$
- $f_c = 0.051 \text{ A s m}^{-1}$
- $m_c = 0.926 \text{ kg}$
- $m_p = 0.125 \text{ kg}$
- $l = 0.101 \text{ m}$
- $J = 7.466 \cdot 10^{-5} \text{ kg m s}^{-2}$
- $h = 0.12 \text{ m}$
- $g = 9.8 \text{ m s}^{-2}$

Identification was done with the help of a Matlab graphical user interface designed by me. Figure 4.6 shows this GUI.

The so-called UI-sliders were the main reason for designing this GUI. The UI-sliders change model parameters and redraw the step response.

Initial parameter estimates were used to complete the state-space model and to find the step responses.

Then the parameters were adjusted with the help of the UI-sliders so the step responses of the laboratory model and mathematical model overlap in the best possible way.

The UI-sliders allowed me to identify the model parameters in a reasonably robust and flexible way.

With the model parameters identified we substitute these values to the A and B matrix.

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1.0961 & -13.2950 & 0.0016 \\ 0 & -101.6004 & 123.9695 & -0.1488 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 0.0483 \\ -0.4506 \end{bmatrix} \quad (4.33)$$

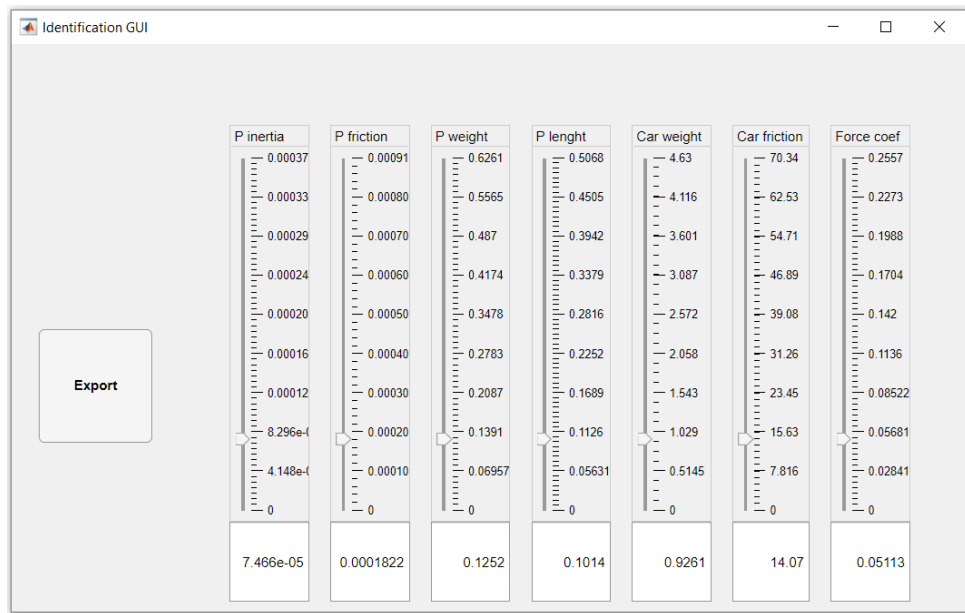


Figure 4.6: Identification GUI

4.3 Control

To demonstrate the functionality of the laboratory model a control strategy was developed. The strategy was supposed to be simple, fast and without unnecessary overshoots.

Two regulators were designed to control the laboratory model. First regulator F is dampening the pendulum and the regulator R is regulating the model displacement [21].

In addition, a dead-zone correction was used with the laboratory model. Figure 4.7 shows the feedback control loop architecture used.

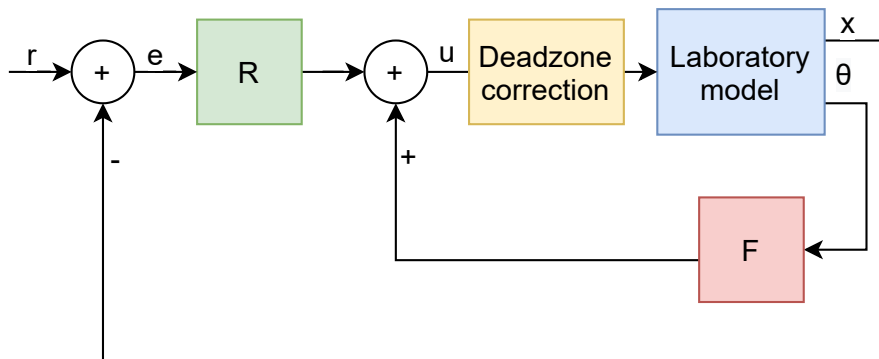


Figure 4.7: Feedback control loop

The regulator F is given by a transfer function shown in a equation 4.34. It consists of a real zero $z = 0$ and complex poles $p = -17.6550 \pm 15.0599i$.

$$F = \frac{9104.4s}{s^2 + 35.31s + 538.5} \quad (4.34)$$

The regulator was designed with the help of the `rltool`. Figure 4.8 shows the step response, root locus and Bode plot of the model system with the regulator in the feedback loop as in Figure 4.7.

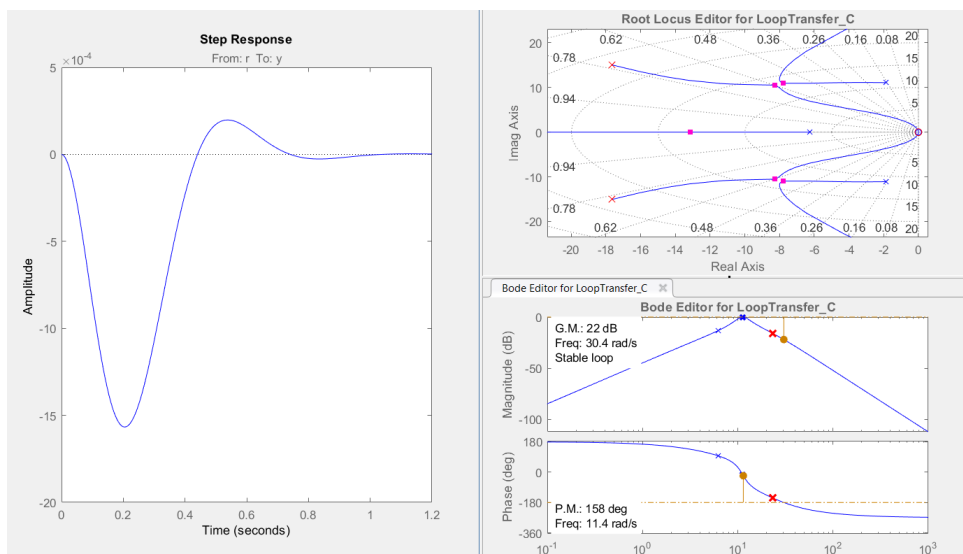


Figure 4.8: Pendulum filter rltool design

Figure 4.9 shows pendulum response to a changing reference without dampening. Figure 4.10 shows motor input that oscillates the pendulum.

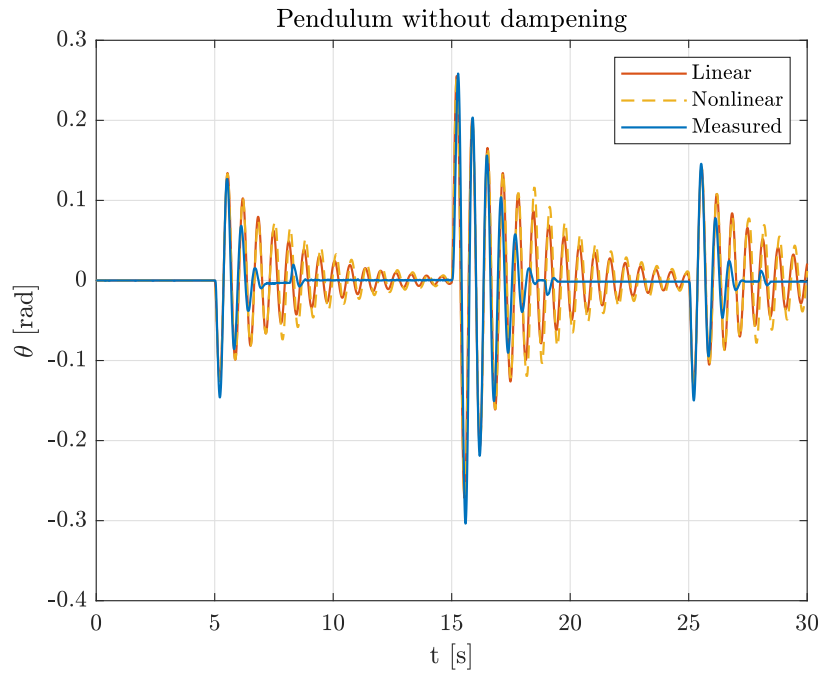


Figure 4.9: Pendulum without dampening

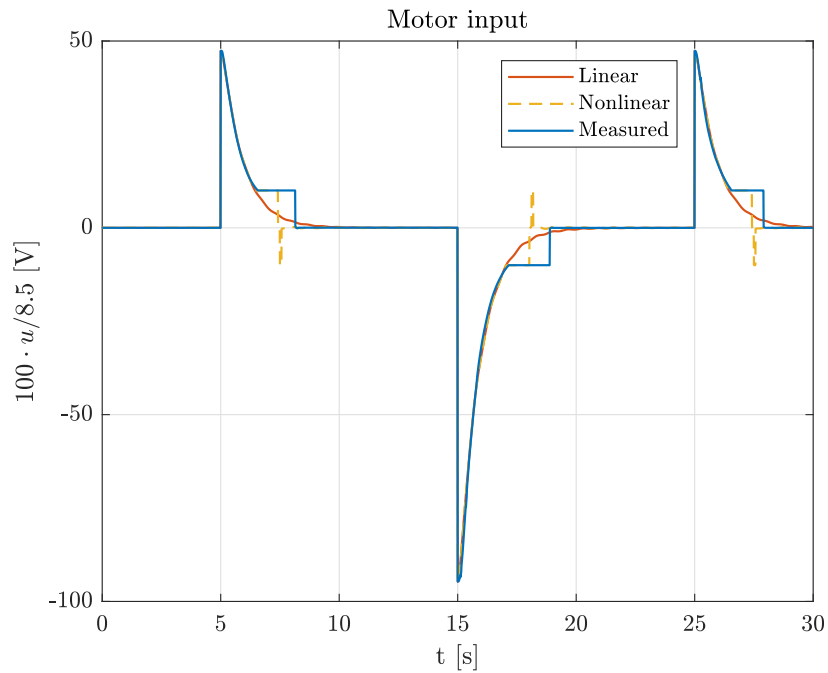
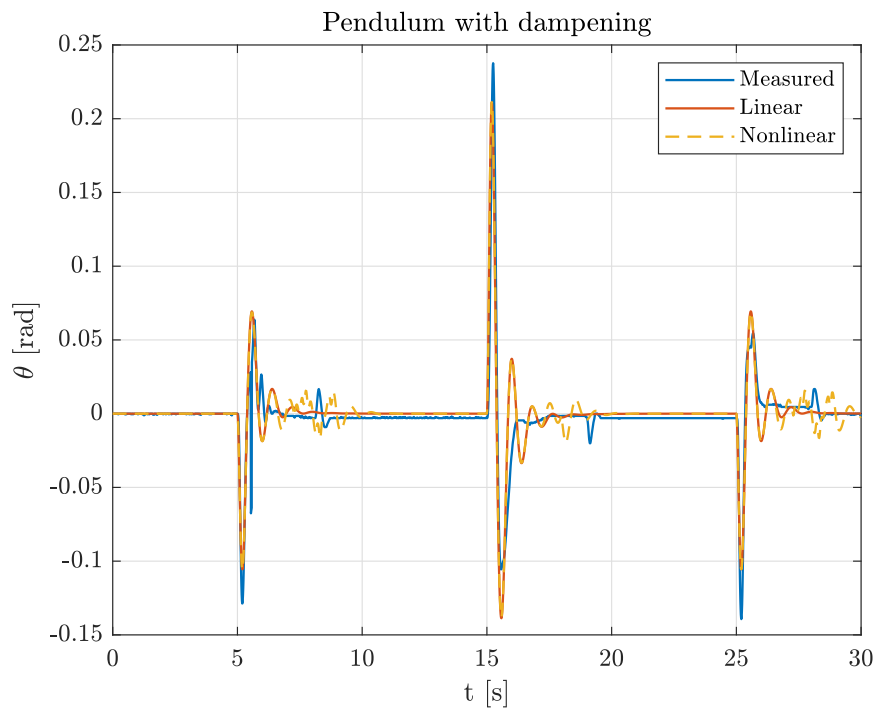
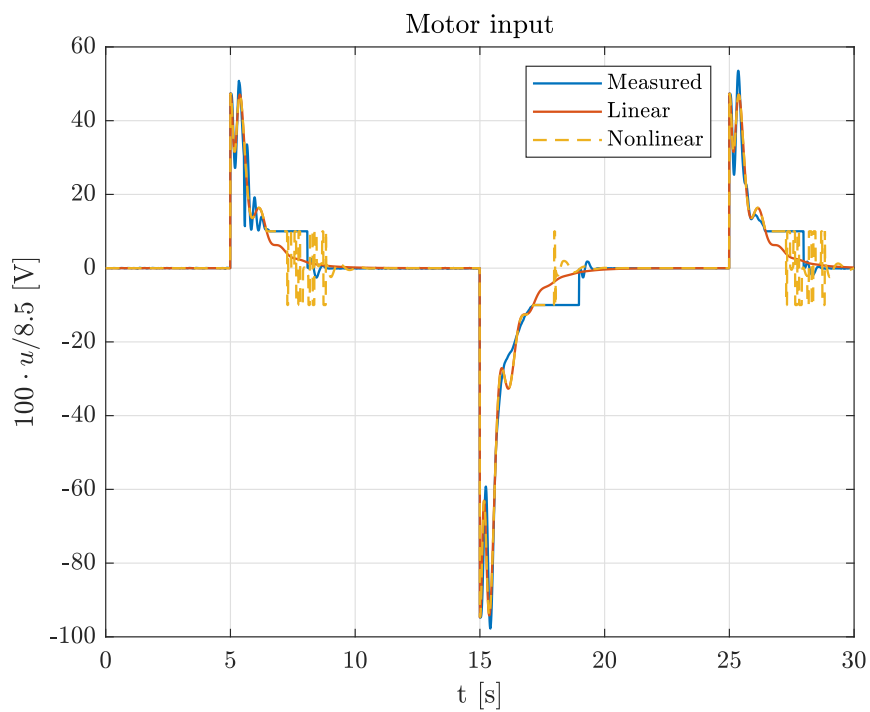


Figure 4.10: Motor input: pendulum is not damped

Figure 4.11 shows pendulum response to a changing reference with dampening. Figure 4.12 shows motor input that oscillates the pendulum.

**Figure 4.11:** Pendulum with dampening**Figure 4.12:** Motor input: pendulum is damped

For regulation of the laboratory model displacement a P regulator R was designed 4.35.

$$R = 284.15 \quad (4.35)$$

Figure 4.13 shows the step response, root locus and Bode plot of the model system with the filter and P regulator.

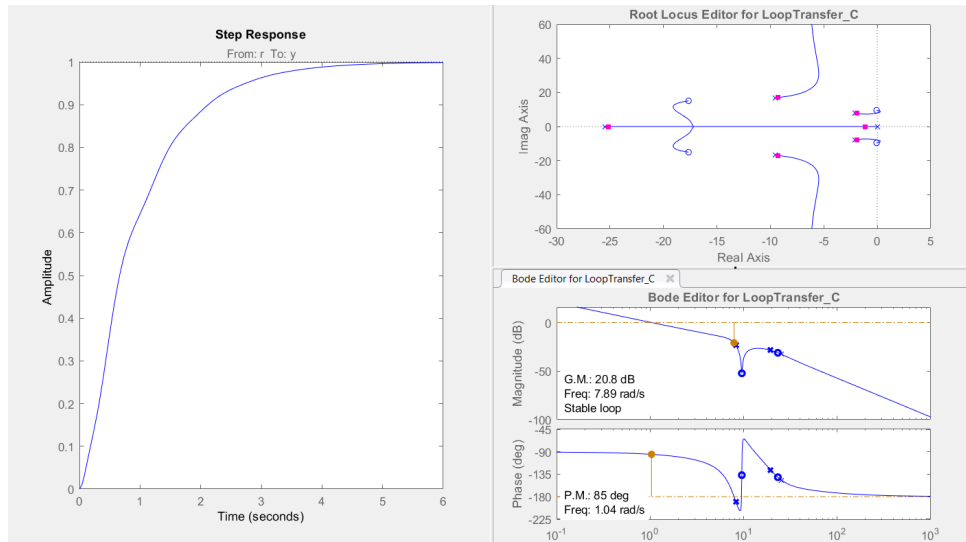


Figure 4.13: Displacement P regulator rltool desing

Figure 4.14 shows displacement reference tracking with the P regulator. There is a noticeable error in the displacement of the laboratory model. That is caused by a dead-zone $[-7, +7]$ of the motor input power.

A dead-zone correction 4.1 was designed to regulate the motor input when the P regulator action is inside the dead-zone 4.1.

Listing 4.1: Matlab function for controlling the model inside the motor dead-zone

```

1 function out = deadzone(in, error)
2     lower = 0;
3     upper = 10;
4     error_val = 0.001;
5     deadfix = 10;
6
7     if in > lower && in < upper && abs(error) > error_val
8         out = + deadfix;
9     elseif in < lower && in > -upper && abs(error) > error_val
10        out = - deadfix;
11    else
12        out = in;
13    end
14 end

```

Figure 4.15 shows reference tracking with the dead-zone correction.

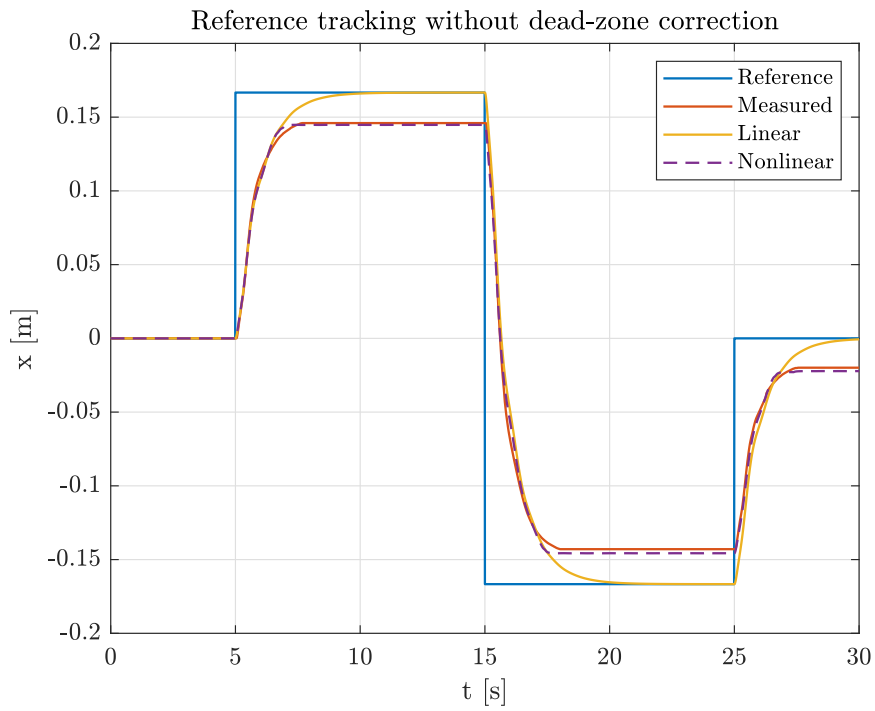


Figure 4.14: Model displacement without dead-zone correction

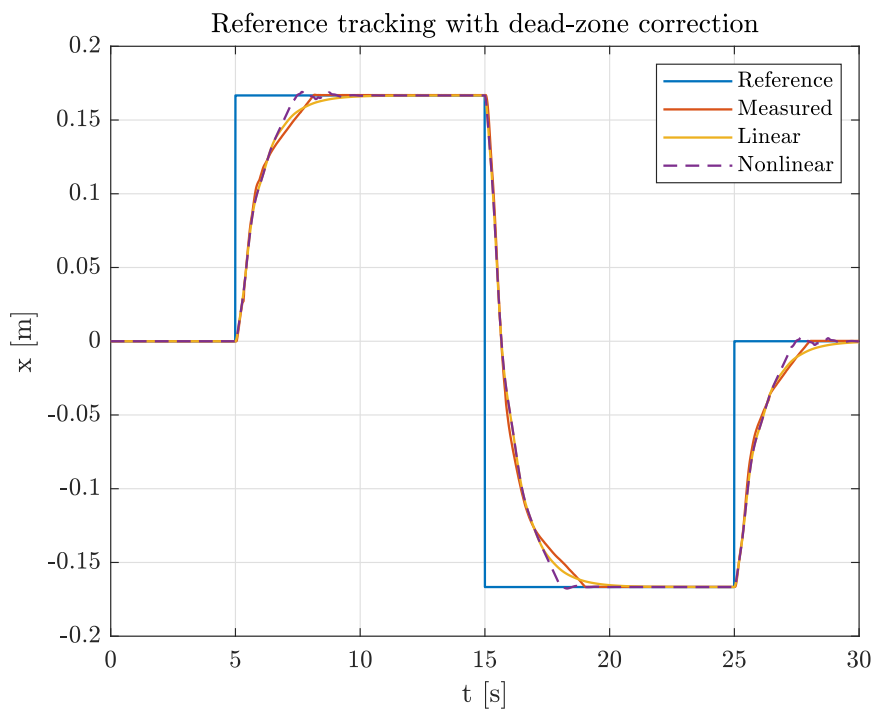


Figure 4.15: Model displacement with dead-zone correction



Chapter 5

Conclusion

In this thesis, a new laboratory model was developed. Existing laboratory models available were reviewed and compared to our requirements. None of the available models was sufficient so new model was developed.

The model hardware consists of a Merkur platform, 3D printed parts and electrical parts. It can be simply described as a pendulum on a cart. The main computing unit of the model is BeagleBone Blue a Linux-based computer.

Seed Studio I2C magnetic rotary sensors and an I2C multiplexor hub were used to measure the pendulum angle and wheel rotation. The model is battery powered and can be controlled via a wireless WiFi connection which is very convenient.

Simulink demo simulation was developed to control the model. Identification experiments were conducted and a linear and nonlinear model was designed. A simple feedback control loop was implemented and verified on the actual hardware.

The model is a simple, cost-effective and robust carry-home platform capable of demonstrating basic and advanced control system designs. It is designed like a plug-and-play device, so students can easily work with this model without a complicated explanation of how the model works.


It has been successfully added to the Automatic Control course taught at the Faculty of Electrical Engineering at the Czech Technical University. Students used the model to complete their term paper. The results were excellent as expected.



Bibliography

- [1] Texas Instruments. Tca9548a low-voltage 8-channel i2c switch with reset. https://www.ti.com/lit/ds/symlink/tca9548a.pdf?ts=1652606187880&ref_url=https%253A%252F%252Fwww.google.com%252F, 2019.
- [2] AMS. As5600 12-bit programmable contactless potentiometer. <https://files.seeedstudio.com/wiki/Grove-12-bit-Magnetic-Rotary-Position-Sensor-AS5600/res/Magnetic%20Rotary%20Position%20Sensor%20AS5600%20Datasheet.pdf>, 2018.
- [3] Armin Steinhauser, Maarten Verbandt, Niels van Duijkeren, Ruben Van Parys, Laurens Jacobs, Jan Swevers, and Goele Pipeleers. Low-cost carry-home mobile platforms for project-based evaluation of control theory. *IFAC-PapersOnLine*, 50(1):9138–9143, 2017.
- [4] Rebecca M Reck and Ramavarapu S Sreenivas. Developing a new affordable dc motor laboratory kit for an existing undergraduate controls course. In *American Control Conference (ACC)*, pages 2801–2806. IEEE, 2015.
- [5] Ryan Krauss. Combining raspberry pi and arduino to form a low-cost, real-time autonomous vehicle platform. In *American Control Conference (ACC)*, pages 6628–6633. IEEE, 2016.
- [6] Matěj Kopecký. Collection of the 3d printed components with sketchfab. <https://skfb.ly/o9y8Y>, 2022.
- [7] Jaroslav Vancl. Merkur platform. <https://merkurtoys.cz/>, 2020.
- [8] Christine Long. Beaglebone blue. <https://beagleboard.org/blue>, 2019.
- [9] Chris TJinGuy. All about lipo balance connectors. <http://www.tjinguys.com/charging-how-tos/balance-connectors>, 2010.
- [10] Seed Studio. Grove 12-bit magnetic rotary position sensor(as5600). <https://wiki.seeedstudio.com/Grove-12-bit-Magnetic-Rotary-Position-Sensor-AS5600/>, 2018.

- [11] Seed Studio. Grove 8 channel i2c multiplexer/i2c hub (tca9548a). <https://wiki.seeedstudio.com/Grove-8-Channel-I2C-Multiplexer-I2C-Hub-TCA9548A/>, 2018.
- [12] Christine Long. Getting started. <https://beagleboard.org/getting-started>, 2019.
- [13] Beagleboard. Latest firmware images. <https://beagleboard.org/latest-images>, 2020.
- [14] Balena. balena etcher. <https://www.balena.io/etcher/>, 2022.
- [15] Tim Jones. Enable sudo without password in ubuntu/debian. <https://phpraxis.wordpress.com/2016/09/27/enable-sudo-without-password-in-ubuntudebian/>, 2016.
- [16] MathWorks. Add support for beaglebone blue hardware. <https://www.mathworks.com/help/supportpkg/beagleboneblue/ug/add-support-for-beaglebone-blue-hardware.html>, 2019.
- [17] Grimmett Richard. Accessing the beaglebone blue remotely via wlan. publish webside, 2017.
- [18] MathWorks. Simulink coder support package for beaglebone blue hardware. <https://www.mathworks.com/help/supportpkg/beagleboneblue/index.html>, 2017.
- [19] MathWorks. Setup and configuration. <https://www.mathworks.com/help/supportpkg/beagleboneblue/setup-and-configuration.html>, 2019.
- [20] Forbes T Brown. *Engineering system dynamics: a unified graph-centered approach*, volume 8. CRC press, 2006.
- [21] Gene F Franklin, J David Powell, Abbas Emami-Naeini, and J David Powell. *Feedback control of dynamic systems*, volume 4. Prentice hall Upper Saddle River, 2002.



Appendix A

Attachments

The following files are attached to the thesis.

drawings\pendulum_drawing.pdf the pendulum drawing
drawings\magnet_drawing.pdf the magnet drawing
drawings\wheel_axis.pdf the wheel axis drawing
datasheets\as5600.pdf the I2C magnetic rotary sensor datasheet
datasheets\tca9548a.pdf the I2C multiplexer hub datasheet
datasheets\beaglebone_blue_datasheet.pdf the BeagleBone Blue datasheet
3d_models\beagle_holder.stl the BeagleBone Blue holder
3d_models\hub_mounting.stl the multiplexer hub mounting
3d_models\pendulum_frame_with_sensor_mount.stl the pendulum frame
with sensor mount
3d_models\pendulum_frame_without_sensor_mount.stl the pendulum frame
without sensor mount
3d_models\sensor_mounting.stl the sensor mouting
matlab_files\all_values.mat MAT file with the identified model parame-
ters for GUI
matlab_files\beagle_model.m Matlab script with the linear model and the
regulators
matlab_files\iden_GUI_full.m the identification graphical user interface
matlab_files\MW_I2C.c the I2C Matlab Simulink modified library source
code
matlab_files\pendulum.mat the measured pendulum response data for GUI
matlab_files\pos.mat the measured cart displacement response data for
GUI
pendulum_on_cart_mercur.zip Automatic Control course term paper as-
signment by Ing. Denis Efremov