Degree Project

Control of Double Inverted Pendulum First Approach

By

DAREINI Ali & DABRETEAU Teerapong

Degree Project in Electrical Engineering

Blekinge Institute of Technology

Karlskrona, Sweden

2015

Thesis Supervisor: Anders Hultgren
Title: Senior Lecturer

Thesis Examiner: Sven Johansson
Title: Senior Lecturer

Abstract

An Inverted double pendulum is a combination of two individual pendulums which represents an example of a nonlinear and unstable dynamic system and it is also a good example of a physical system which can exhibit chaotic behavior. This document contains a first analysis of the model and the control of this system. Also presented is the installation of the electrical materials needed to control the system contain instrumenting the motor, current measurement system, motor shaft angle sensor, vision system and MYRIO which is an embedded hardware device created by National Instruments will be used for data acquisition and control the system.

Keywords: Double inverted pendulum, nonlinear system, unstable dynamic system, dc motor, Maxon device, Hough transform, edge detection, pattern recognition, vision system, NI MYRIO, LabVIEW, MATLAB, PSPICE.

Acknowledgement

We would like to express our sincere gratitude to Anders Hultgren who has been our supervisor and mentor for the past five months.

Moreover, we want to thank Matz Lenells who gave us a lot of useful information regarding our project and Kristian Nilsson who gave us the My Rio data acquisition.

Contents

Introduction **7**
1 The System and equipment **8**
Introduction 8
1.1 The equipment 9
1.2 DC Motor 9
1.3 Voltage Regulator: Shunt Regulator DSR 70/30 10
1.4 ESCON 50/5 Servo Controller 11
1.5 Control the DC Motor with LabVIEW 13
1.6 Hardware connections 16
1.7 The Encoder 18
Introduction 18
1.7.1 Feature of the encoder 18
1.7.2 Getting the angular position of the first limb 19
1.7.3 Implementation of the shaft angle sensor on LABVIEW 19
2 Motor`s Current measurement **21**
2.1 Getting the current motor with the “ESCON Servo Controller” device 21
2.2 Getting the current motor with an electrical scheme 21
2.2.1 General view 21
2.2.2 Shunt Resistor + tension Divider Bridge 22
2.2.3 Amplifier with Galvanic isolation 23
2.2.4 Realization of the amplifier with galvanic isolation 27
2.2.5 Conclusion: Feature of the current measurement electric scheme 36
3 Vision System **37**
Introduction 37
3.1 First method: The Hough transform [18] 37
3.1.1 Hough Transform principle 37
3.1.2 Hough Transform M to 1 37
3.1.3 Hough Transform 1 to M 38
3.1.4 Operations Processing Chain 39
3.1.5 Implementation on MATLAB 40
3.1.6 Implementation of the Hough Transform on LabVIEW 42
3.2 Second method: by detecting some markers 45

3.2.1 Picture format 46
3.2.2 Marker Detection 46
3.3 Third method: Using pattern recognition 51
3.3.1 Introduction 51
3.3.2 MATLAB Function 51
3.3.3 Running an experiment with the vision system 53
3.3.4 Verifying the vision system 53
Conclusion 55
4 Modelling and Control 57
4.1 Moment of Inertia 57
4.2 Gravity of Earth 61
4.3 Double Pendulum without the motor 63
4.4 Double pendulum with DC motor 67
4.5 Friction 70
4.5.1 First Experiment 73
4.5.2 Second Experiment 74
4.6 Simulating the nonlinear model 76
4.7 Linearized model 83
4.7.1 Linearizing around down position 84
4.7.2 Linearizing around up position 87
4.8 Control law 88
5 Interface of LabVIEW 94
5.1 NI MYRIO 94
5.2 Survey 95
5.3 Structure of the code on LabVIEW 95
5.4 LabVIEW Interface 98
Conclusion 101
REFERENCES 102
TABLE OF FIGURES 103
Appendix 1 - Specifications of the camera 107
1- Frame rate 107
Introduction 107
Case 1: Eigenvalue -3 108
Case 2: Eigenvalue -6 109
Case 3: Eigenvalue -9 110
Case 4: Eigenvalue -15 111

Conclusion.....112
2- Shutter speed 113
3- Resolution.....114
References:.....115
Appendix 2 – MATLAB Codes for Hough Transformation 116

Introduction

The inverted double pendulum is known as one of the typical nonlinear systems. Because it has the property of simple structure and unstability, it has been widely used for the research and experiment.

An inverted double pendulum is called an underactuated system because it has greater number of joints than the number of actuators. The control of underactuated systems is currently an active field of research due to their broad applications in robotics, aerospace and marine vehicles. Examples of underactuated systems include flexible-link robots, walking robots, acrobatic robots, helicopters, satellites, space robots, underactuated marine vehicles, the pendubot, spacecrafts etc.

The final purpose of this project is swung up the pendulum from the vertical down stable position to the upward unstable position in a controlled trajectory and then stabilizing the pendulum in upward position.

The control algorithm uses the rotation angle of the pendulums as input measurements provided by different sensors. While we used an encoder for the first pendulum and a purely vision based tracking system for the second pendulum.

In this project we study extensively the previous similar works. In [1] a vision system was used for control an inverted pendulum by using a cascaded filters. In [2] also a vision system was used for control a cart inverted pendulum. The challenge task for Wang, the writer of paper [2] is how to stabilize the pendulum by using a low cost CCD camera. Paper [3] is about control of a rotary inverted pendulum based on LQR mapping. Krishen the writer of paper [3] uses two control laws, one for swing up and another for the stabilization of the pendulum. Paper [4] by using this idea that, humans can control a pendulum without knowing the mass and the length of a pendulum tried to control a card pendulum. Yasunobu uses a fuzzy controller and could confirm the effectiveness of human control strategy.

In the first chapter, the structure of the system and relevant equipment is showed and it talks about how to control the DC motor and implement the encoder. The LabVIEW codes also brought for better understanding.

The second part contains information and study about a current measurement electrical scheme for the DC motor.

The third part contains describes the vision systems we used in order to get the angle position of the second limb.

The fourth's part contains deriving the equations of motion, nonlinear and linearized models and finally the control law of the system.

The last chapter is about the interface of LabVIEW, codes and the way for using the interface.

1 The System and equipment

Introduction

In the topic of control, pendulums are a good example of nonlinear and unstable systems. For having a more challenge task we select a double pendulum. In most of the double pendulums the cables are a problem when the pendulum rotates, specially the cable of the second limb's encoder. As it can be seen in Figure 1, in this project we make the second limb free of cable and instead of using an encoder we will use a vision system that is completely separate from the pendulum. The basic geometry of our double pendulum is shown in Figure 2.

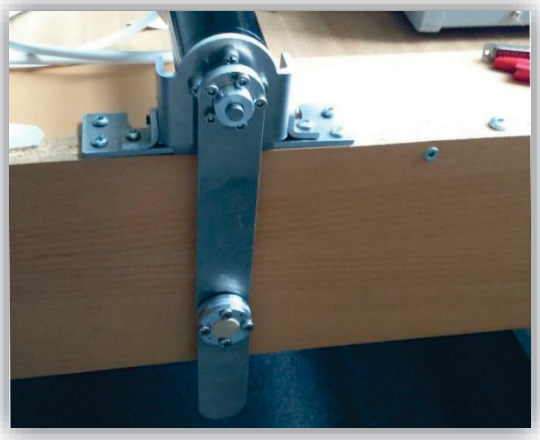


Figure 1: picture of the double inverted pendulum

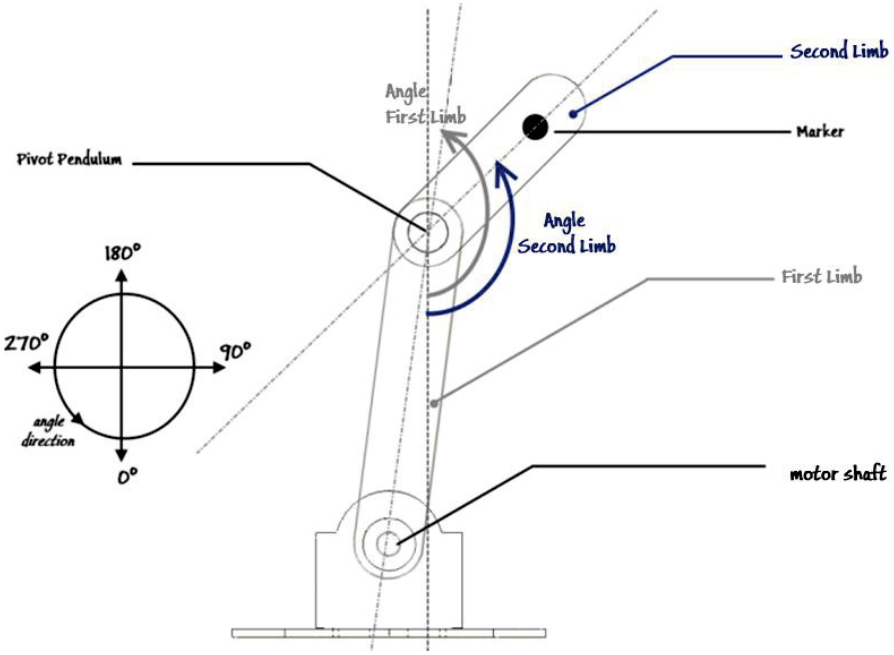


Figure 2 : Our real double pendulum connected to a DC motor shaft

We use a DC motor to drive the pendulums. The First limb, big pendulum is connected to the shaft of the motor and the second limb, small pendulum is connected to the first limb by a joint contain bearing.

1.1 The equipment

The actuator of the double inverted pendulum is a DC motor from the MAXON Company. This is the only actuator of the system to control the displacement of the 2 limbs that constituted the double inverted pendulum.

Maxon offers us 2 devices, Figure 3 and Figure 4, in order to control the power part and the control part for the DC motor.

Figure 3 : the shunt regulator DSR 70/30, Taken from a web page

The shunt regulator DSR 70/30 (see Figure 3) is responsible to regulate the voltage supply value in order to ensure a stable voltage supply for the Servo controller (see Figure 4).

Figure 4: ESCON 50/5 Servo Controller, taken from a web page

The ESCON 50/5 Servo Controller device (see Figure 4) is responsible for generating pwm (Pulse-width modulation) a signal to control the speed of the DC.

The DC Motor can generate a high current variation when he is controlled and supplied by a high source (in our case the supply voltage value can reach 50V). The power section of the motor can easily disturb the state of the signal coming from the control section (signals for sensors or coming from them like the encoder signal). It is important to separate both section.

1.2 DC Motor

The DC motor used for the project is Maxon 370356. A study about its feature was already made by Matz Lenells: « Lab 3: Modeling and simulation of a brushed permanent magnet DC-motor » (see Table 1).

Table 1 : Data Feature of the DC motor Maxon 370356 from Matz Lenells report

Motor	48 V	24 V
Motor length	108.0 mm	
Type power	200.0 W	
Outer diameter	50.0 mm	
Bearer Type	Wälzlager	
Number of autoclave cycles	0.0	
Commutation	GB	
Type	RE	
Values at nominal voltage		
Nominal Voltage	48.0 V	24.0 V
No load speed	4900.0 rpm	5950.0 rpm
No load current	88.4 mA	236.0 mA
Nominal speed	4620.0 rpm	5680.0 rpm
Nominal torque (max. continuous torque)	420.0 mNm	405.0 mNm
Nominal current (max. continuous current)	4.58 A	10.8 A
Stall torque	7370.0 mNm	8920.0 mNm

1.3 Voltage Regulator: Shunt Regulator DSR 70/30

The shunt regulator DSR 70/30 is responsible to regulate the voltage supply value in order to ensure a stable voltage supply for the Servo controller. The regulator limits the voltage increase, from the power supply, up to a permissible value and to transform the excess energy into heat.

From the datasheet of the constructor, Maxon Company, the cable diagram is shown in Figure 5. You need a power supply and an actuator, like a DC Motor.

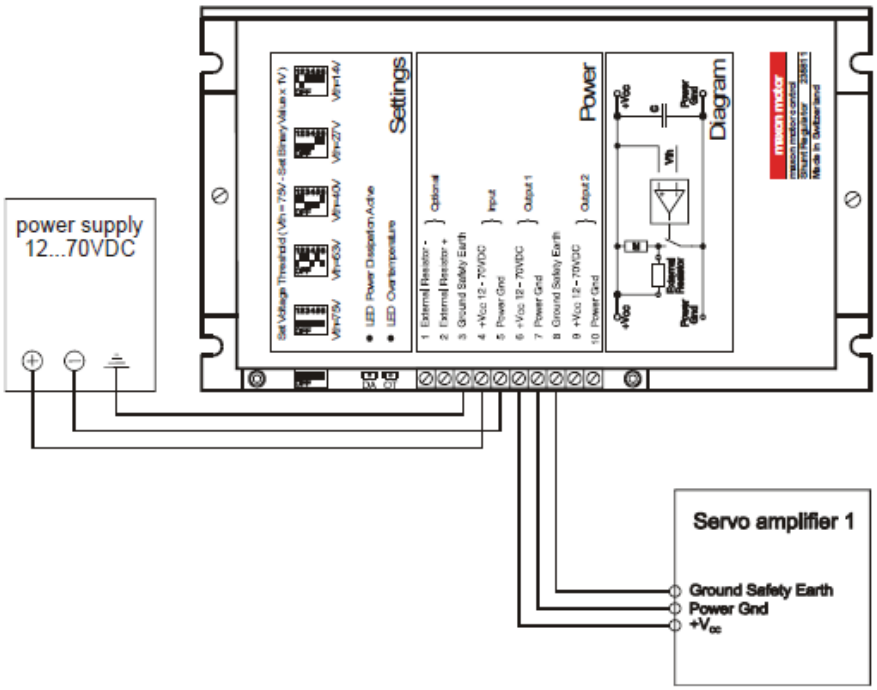


Figure 5: Minimal External Wiring from the Company datasheet

1.4 ESCON 50/5 Servo Controller

The ESCON 50/5 Servo Controller device allows us to generate a signal to control the DC motor: usually a PWM (being commanded by an analog set value). This device can be configured on voltage or current mode; we will work with the voltage mode.

From the datasheet of the constructor, the cable diagram is shown in Figure 6. Table 2 corresponds to our connection for our project.

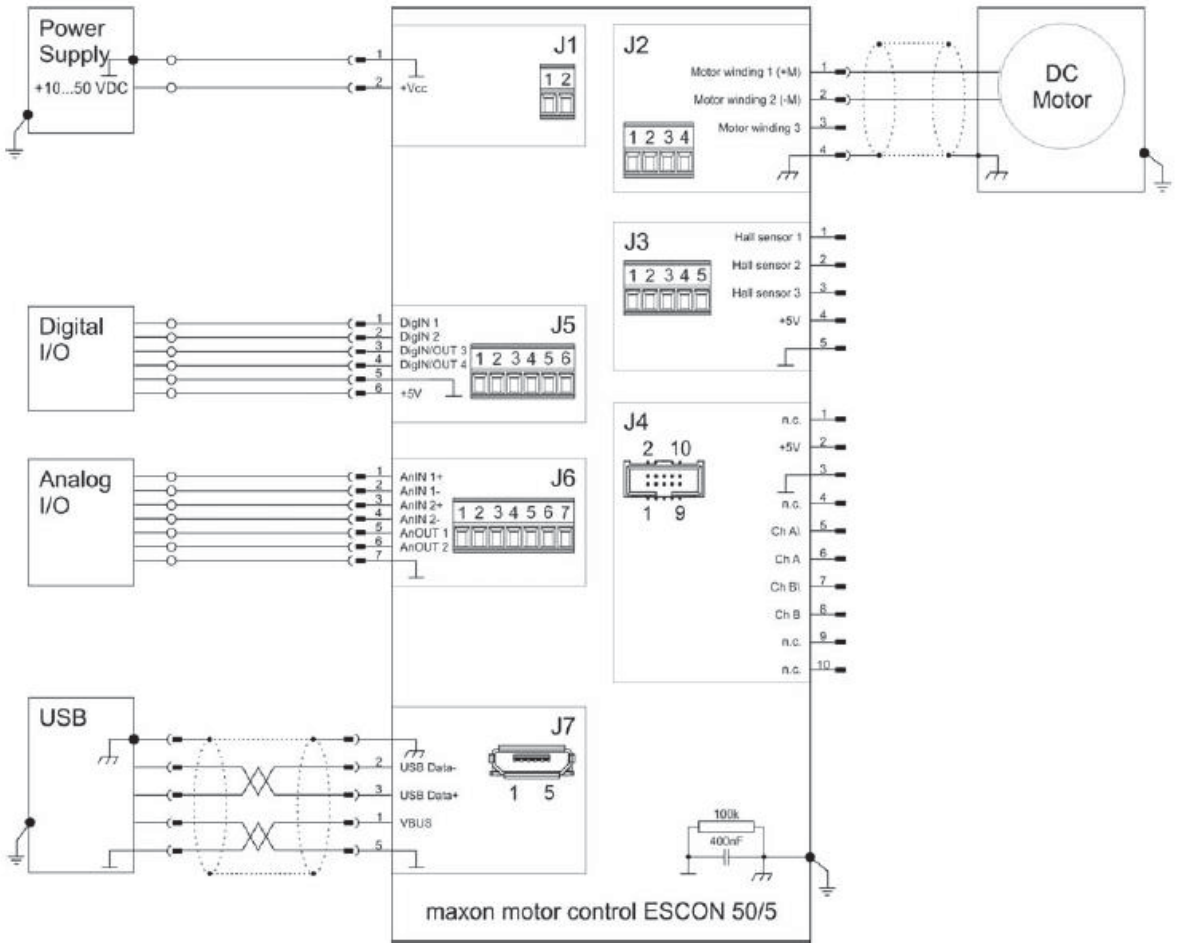


Figure 6: Wiring DC motor to the ESCON 50/5, Company datasheet

Table 2: Hardware connection

J1	Connected to the power supply voltage (in our case from the voltage regulator DSR 70/30 Maxon)
J2	Toward to the DC motor
J5	This section is responsible to monitoring the enabling supply for the DC motor. It is commanded by a logical signal coming from the Ni MYRIO board. This section can be configurable with software. In our case, the port "Digital input 1" has been chosen. The ground reference of this section must be the same ground reference from the board Ni MYRIO.
J6	This section is responsible to control the speed and the direction of the DC motor. It can sense the current running through the DC Motor.

	This section can be configurable with software. In our case: - Analog input 1: analog set value that command the speed and the direction of the DC motor. His value is between [-10V 10V] (generated from an analog output Ni MYRIO). The direction is determined by the sign of the analog set value. The ground reference of this section must be the same with the ground reference from the board Ni MYRIO.
J7	USB connection to configure the Maxon control device with the ESCON software

The device is designed to be configured via USB interface using the graphical user interface «ESCON Studio» for Windows PCs. The latest version of this software can be found in their website. The configuration we used for the motor Controller ESCON is shown in Table 3).

Table 3: Configuration for the motor Controller ESCON

Configuration for the motor Controller ESCON	
Analog output1	Image of the Motor current (1V represent 1A)
Analog input1	Speed Motor Order [-10V 10V] for [-300rpm +300rpm]
Digital input1	Enable motor supply

The maximal nominal speed that the motor can reach is too high (not safe at all for the table) (4620 rpm) for our system. It is necessary to reduce the maximal speed value when the analog set value is absolute (10V).

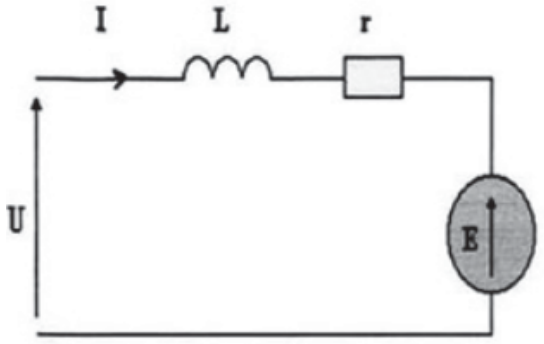


Figure 7: DC Motor equivalent circuit model, where U: the terminal voltage of the DC motor, I: the armature current, L: the armature inductance, r: the armature resistance, E: the back efm)

To control the speed, the « Maxon Motor Control » generates a PWM signal (Pulse Width Modulation) (see Figure 8). A DC motor have inductive load (see Figure 7). For the motor if we apply a PWM signal and if his frequency is high enough, the variation of the derivative of the measured current through the motor can be considered as constant. The speed value of the DC motor can controlled with the duty cycle parameter of the PWM signal.

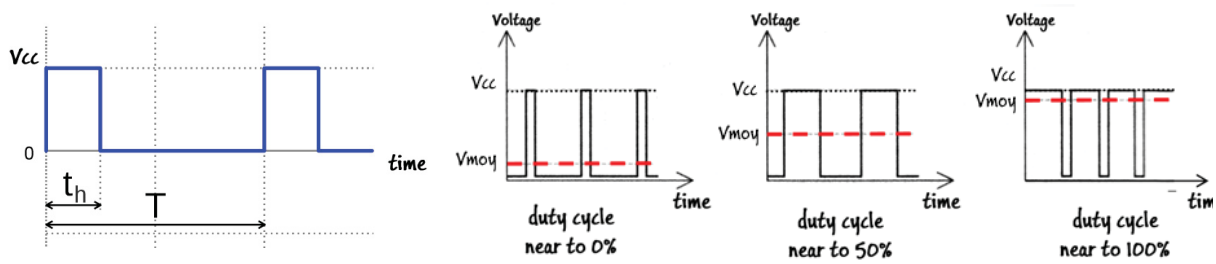


Figure 8: PWM waveform scheme principle. Where Vmoy: average voltage, the: time high state, T: Time periodicity, Vcc: Maximum voltage

The formula of the average voltage on the motor terminals is : $Vmoy = \frac{th}{T} \times Vcc$

It is possible to reduce the speed value for the motor when we apply the maximal voltage value on the analog input (absolute value (10V)). This option is very interesting in our case, because the maximal speed value that the DC motor can reach is 4630 rpm (too high for our case). We configure it from the MAXON Controller somehow when we apply absolute value (10V) in the analog input port, the maximal speed value just be 300 rpm (for the safety for our table). Electrically it will just change the maximal value of PWM duty cycle allowed to 7.5%.

1.5 Control the DC Motor with LabVIEW

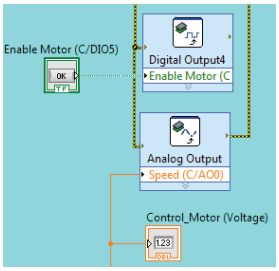


Figure 9: Management of the Supply voltage for the motor on LabVIEW (from the LabVIEW Program of our project)

NI MyRIO, the embedded device, created by National, is programmed to generate 2 important signals in order to control the DC motor (see Figure 9):

- Enable : a logical signal that allow the motor to be supplied
- An analog output signal between [-10V 10V] responsible to control the direction and the speed motor value

From the LABVIEW interface created, the user can choose 3 modes, see

Table 4.

Table 4: Modes to control the DC motor in LabVIEW

Manual mode	Allows to control the motor manually (speed and direction).
Feedback mode	Allows user to control the motor automatically. This mode should be selected if the user wants to use the feedback control he has implemented.
Waveform mode	Allows user to control the state of the motor by sending some configurable generated signals (like: sine, square, step waveform).

Manual mode:

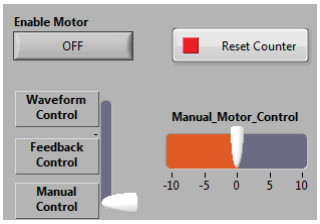


Figure 10: Part of the LabVIEW interface (from the LabVIEW Program of our project)

When the manual mode is selected,

- To supply the motor, press the Enable button to "ON".

The speed voltage and the direction of the motor can be controlled by the slide bar (represent the output voltage value of the analog output).

Feedback mode:

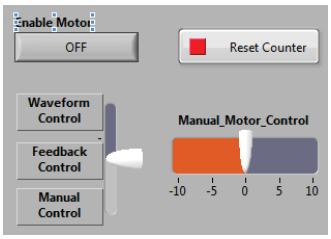


Figure 11: part of the LabVIEW Interface (from the LabVIEW Program of our project)

When the Feedback mode is selected,

- To supply the motor, press the Enable button to "ON".

The output voltage value, destined to fix the speed and the direction value of the DC motor, is calculated from the Feedback part.

Waveform mode:

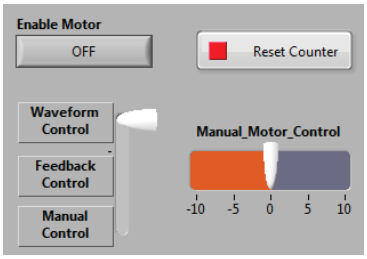


Figure 12: Part of the LabVIEW interface (from the LabVIEW Program of our project)

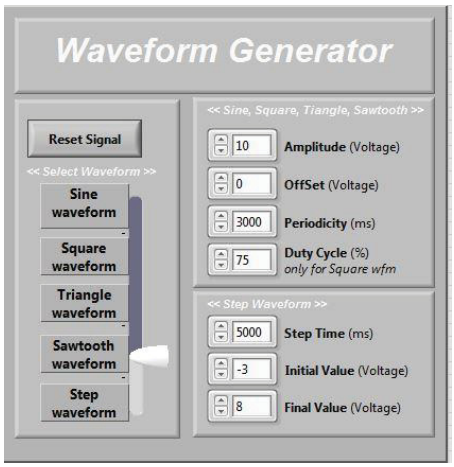


Figure 13: Waveform generator interface (from the LabVIEW Program of our project)

When the waveform mode is selected (see Figure 12):

- To supply the motor, press the Enable button to "ON",
- The output voltage value, destined to fix the speed and the direction value of the DC motor, is controlled from the signal generated (see Figure 13).

The signal generated is not a continuous signal but a discrete signal (the refresh time value depend to the refresh time to call the Task "control motor") and quantified (by the analogic feature output of MyRIO)

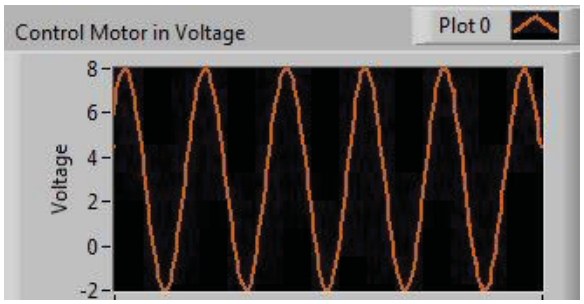


Figure 14: Sine waveform generated (amplitude: 5V, offset: 3V, periodicity: 3000 ms) (from the LabVIEW Program of our project)

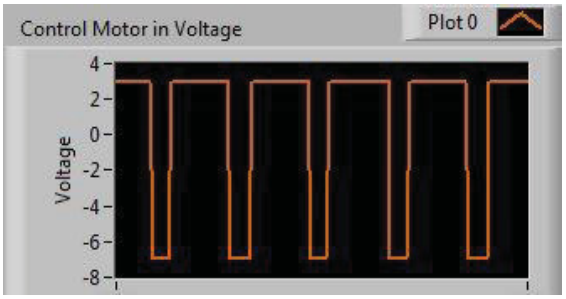


Figure 15: Square waveform generated: (amplitude: 5V, offset: -2V, periodicity: 3000 ms, duty cycle: 75%) (from the LabVIEW Program of our project)

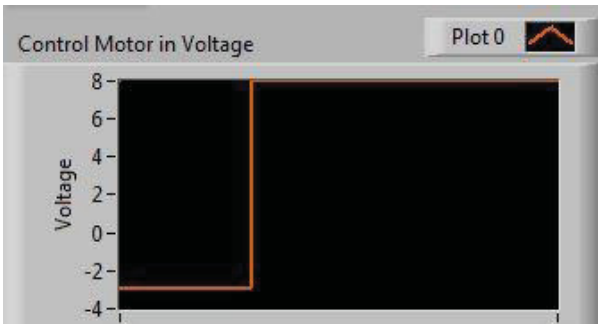


Figure 16: Step waveform generated (Step Time: 5s, initial value: -3V, final value: 8V (from the LabVIEW Program of our project)

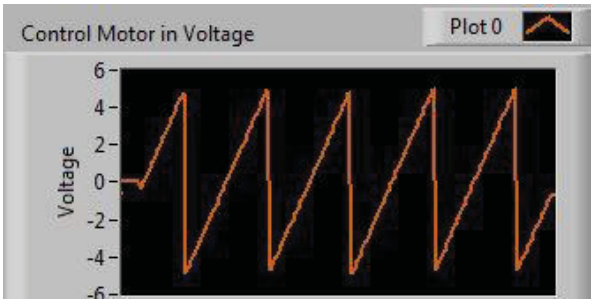


Figure 17: Saw tooth waveform generated: (amplitude: 5V, offset: 0V, periodicity: 3000 ms) (from the LabVIEW Program of our project)

1.6 Hardware connections

The hardware connection for our entire project is shown in Figure 18, Table 5 and Table 6. The USB camera is used as a sensor vision in order to get the angle position of the Second limb. The encoder is used as a mechanical sensor in order to get the angle position of the first limb.

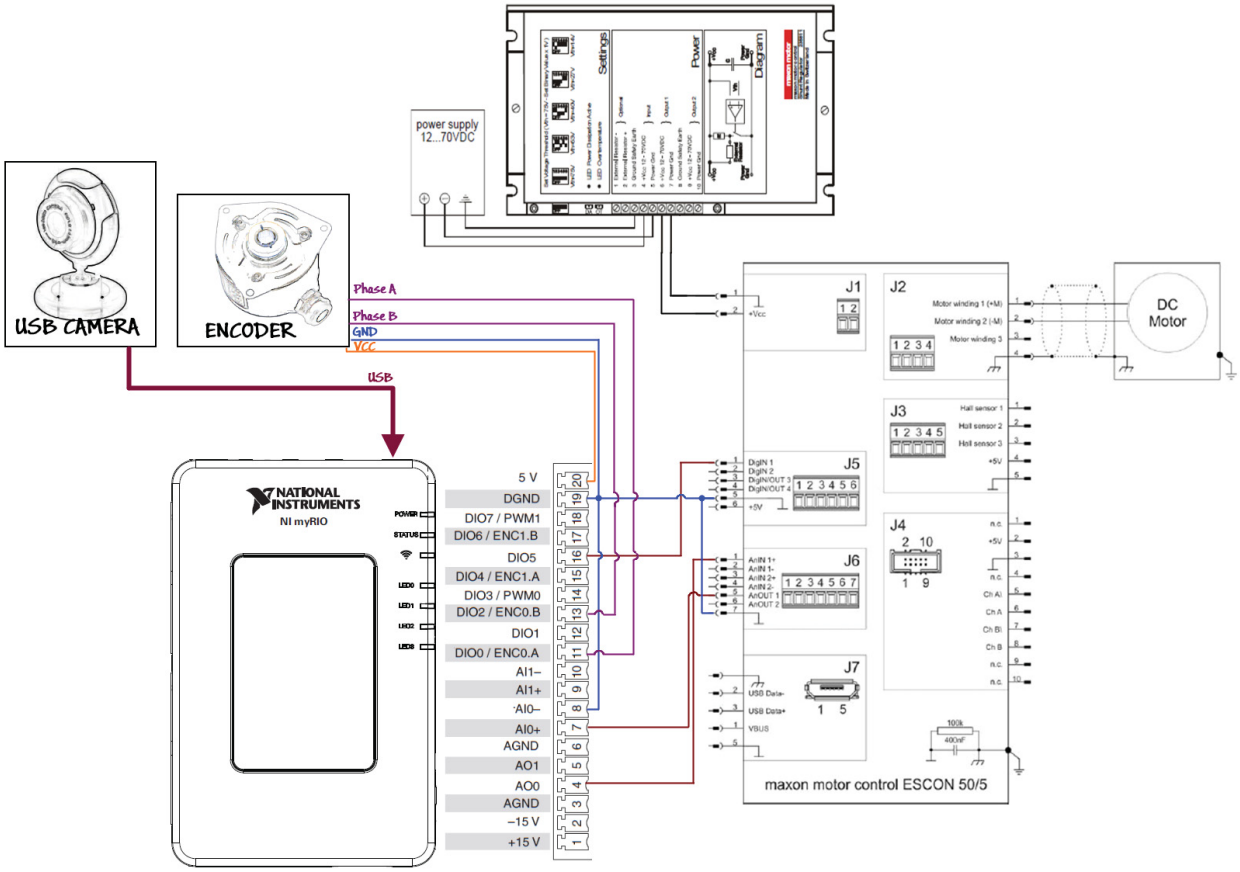


Figure 18: Hardware connections

Table 5: Configuration for the motor Controller ESCON

Configuration for the motor Controller ESCON	
Analog output1	Image of the Motor current (1V represent 1A)
Analog input1	Speed Motor Order [-10V 10V]
Digital input1	Enable motor supply

Table 6: Configuration for NI MYRIO

Configuration for NI MyRIO	
DIO2 DIO0	Encoder Phase A Encoder Phase B
C/AIO+ C/AIO-	Motor current (can be the analog input1 from ESCON device) GND (MyRIO)
C/AO0 C/DIO5	Speed Motor Order [-10V 10V] Enable motor supply

1.7 The Encoder

Introduction

Figure 19: Encode SCH50F, Taken from a web page

To get the angle position of the arm's motor, an incremental encoder sensor is used. This device is fixed on the shaft motor. Our industrial encoder is SCH50F (Figure 19).

1.7.1 Feature of the encoder

Encoder SCH50F works similar with other classical incremental encoder. The motor shaft is mechanically linked to the encoder shaft. The rotation of the encoder shaft moves a solidary disc (Figure 20).

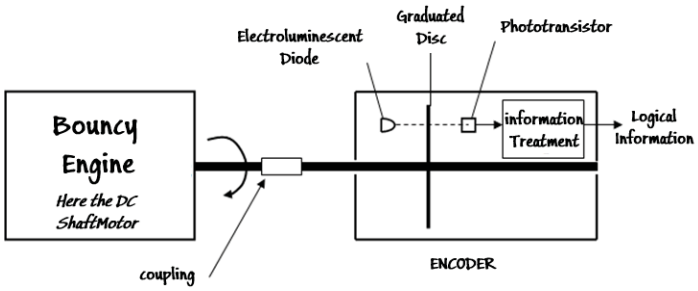


Figure 20: Scheme of the functioning of the encoder

The light beam emitted by the electroluminescent diode through the graduated disc slot allows the phototransistor to create an analog signal. This signal is then amplified and converted to a logical square signal (see Figure 21).

The Graduated disc includes at least:

- 2 surrounding paths usually called A and B
- 1 lower surrounding path: Z

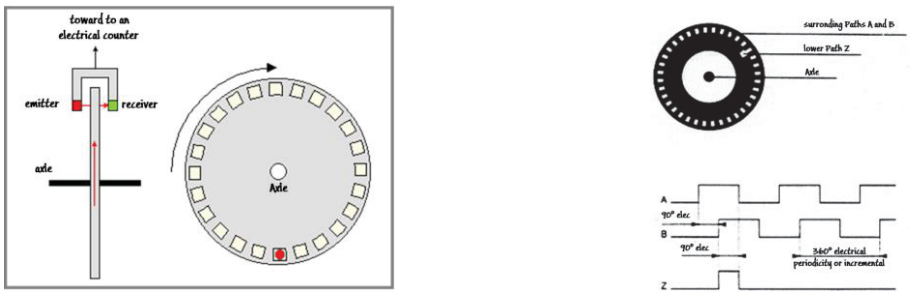


Figure 21: Scheme of the functioning of the encoder

The surrounding paths are decaled by a quarter of period (90°) and divided by N equal intervals that are alternatively opaque and transparent. A complete revolution means that the light beam was interrupted N times and N analog consecutive signals were generated. Behind the path, 2 phototransistors generate 2 logical signals A and B, out of phase by 90°.

This out of phase contains the direction information (Figure 22):

- Direction 1, B = 0 with the signal A rising edge
- Direction 1, B = 1 with the signal A rising edge

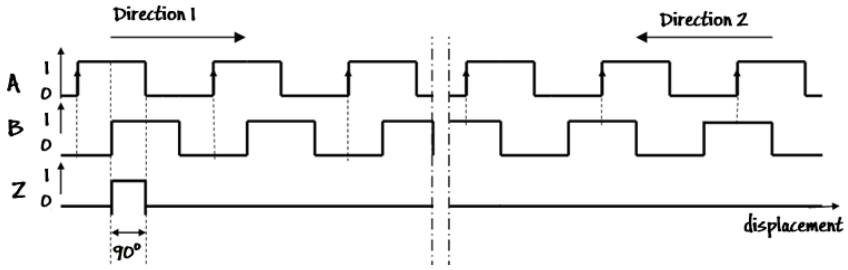


Figure 22: Scheme of the functioning of the output signals from the encoder

When we detect a change state of one of the two signals (A and B), in function of the out of phase signal A to signal B (determine the direction), the counter will increment or decrement his value.

1.7.2 Getting the angular position of the first limb

By knowing the necessary incremental numbers for a complete revolution, it will be easy to calculate the angle position:

$$Angle = \frac{CounterValue}{nbTickperRevolution} * 360$$

$$AbsoluteAngle = \left(\frac{CounterValue}{nbTickperRevolution} * 360\right) - 360 * E\left(\frac{CounterValue}{nbTickperRevolution}\right)$$

Angle: angular displacement of the first limb (in degree)

Absolute Angle: absolute angular motor displacement $\in [0\ 360]$ (in degree)

1.7.3 Implementation of the shaft angle sensor on LABVIEW

It is necessary to link the signal from the encoder (A and B) to a system that can watch every changing state. A counter with a high memory value is needed to save all number of impulsion captured. MYRIO is equipped with a FPGA, which some section are entirely configured for analyzing logical quadrature signal. By this way, it is easy to implement the incremental encoder on LABVIEW, because a block for that application is already made (memory and input ports are automatically managed after some configuration).

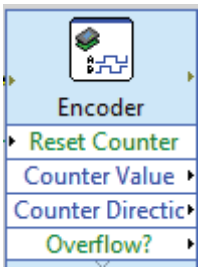


Figure 23: Encoder LabVIEW Block (from the LabVIEW Program of our project)

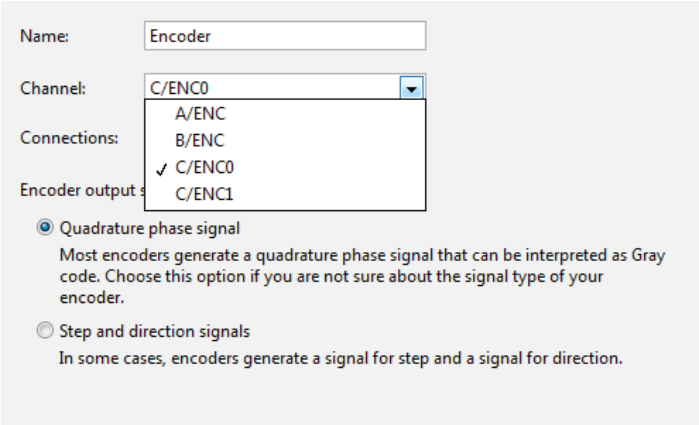


Figure 24: Configuration of the encoder block (from the LabVIEW Program of our project)

LabVIEW block destine to manage an incremental encoder (Figure 23 and Figure 24). We Configure this block by choosing the input ports for the logical signals A and B. Using of this block is very interesting materially, because the block operating will be completely autonomous from the rest of the program.

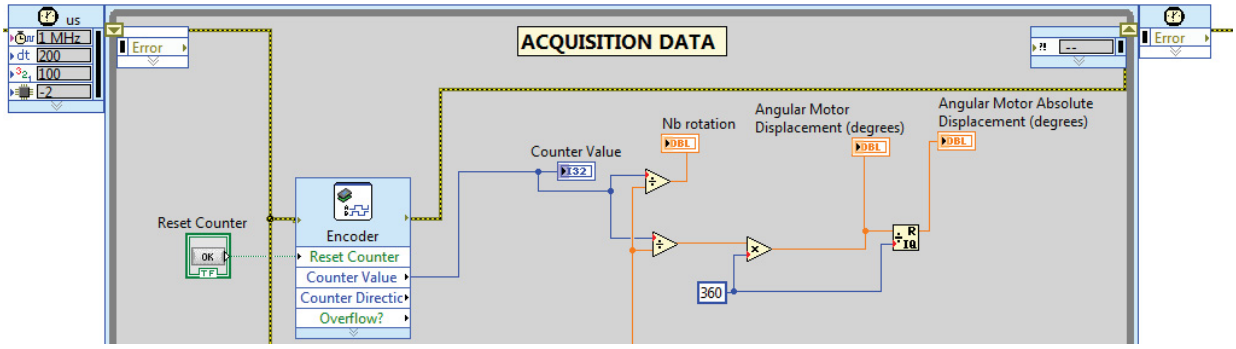


Figure 25: Block acquisition data responsible to refresh the angle position value from the encoder (from the LabVIEW Program of our project)

2 Motor`s Current measurement

The current through the motor can give us some useful information about the state of the shaft:

- Can detect an over current if the shaft motor is blocked
- Get an image of the state of the torque. In fact, the 2 parts of the double inverted pendulum have a significant impact weight. It is possible to have an image of each displacement part of the inverted pendulum (first limb and the second limb) by interpreting the torque value.

2.1 Getting the current motor with the "ESCON Servo Controller" device

There are many ways to catch an image of the current through the motor (value in tension). The easiest way (already implemented) is to use « ESCON 50/5 Servo Controller » device. It is possible to configure one port of the "Analog output part" to an output responsible to send an image of the current through the motor. A problem that can come from the device feature is:

The bandwidth size is really small (20 KHz) and the output value is quantified (output analog voltage allowed between -4V to 4V with a resolution 2.3mV, we lost accuracy)

2.2 Getting the current motor with an electrical scheme

2.2.1 General view

Our suggestion for getting the image of the motor`s current is a resistor shunt with amplifier and galvanic isolation (to separate power section to the command section). The general structure of the electrical circuit is shown in Figure 26.

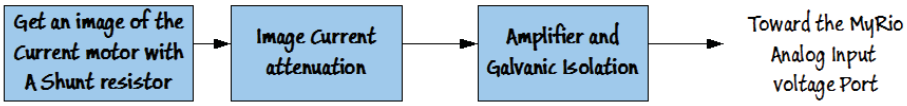


Figure 26: General view of the processing chain for the current measurement in the motor

The first part (Figure 27) of the electrical circuit scheme is responsible to get an image of the current motor and to resize this value for the second part (the amplifier with galvanic isolation).

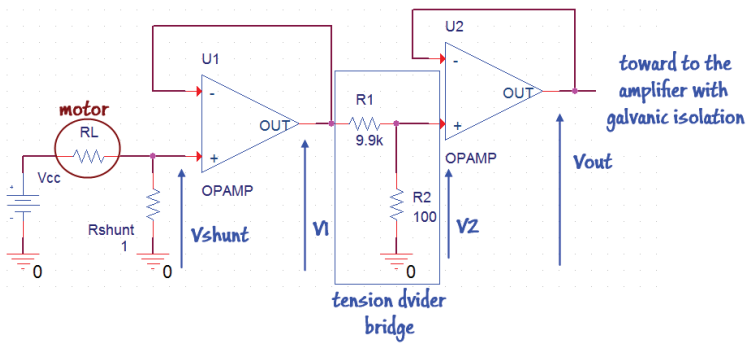


Figure 27: First part of the electrical scheme for the current motor measurement

The 2nd part of the electrical circuit scheme (Figure28) is responsible to transfer the image current value between the power section to the command section (NI MYRIO for example), through an amplifier with galvanic isolation.

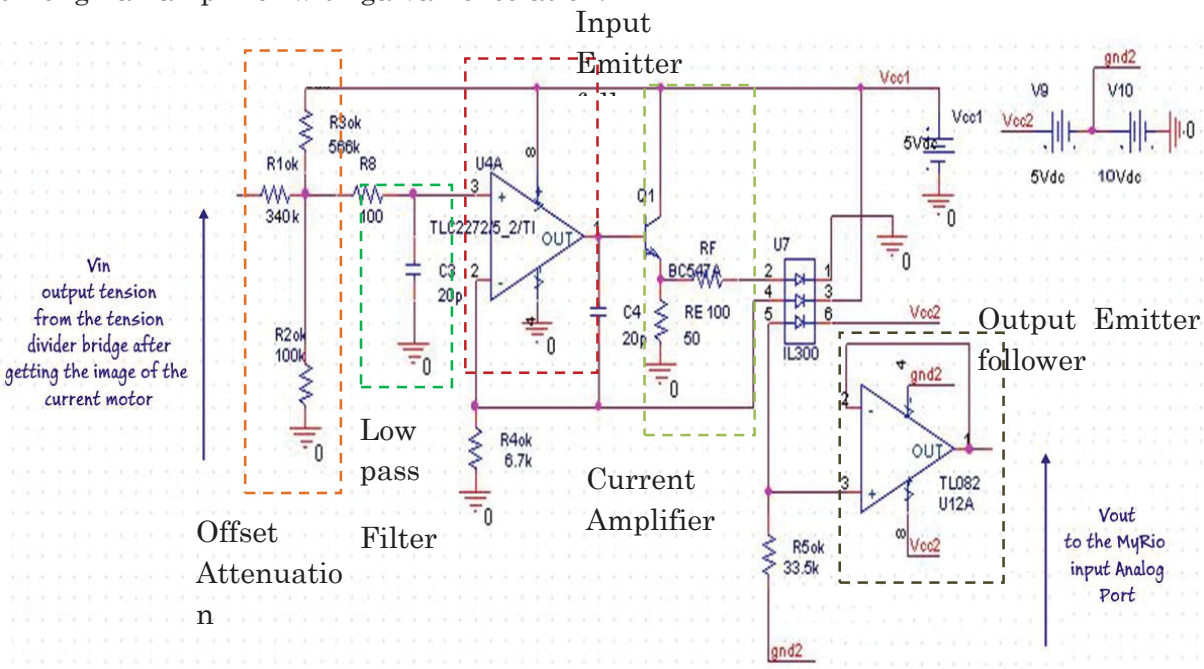


Figure28: Second part of the electrical circuit scheme: amplifier with galvanic isolation

2.2.2 Shunt Resistor + tension Divider Bridge

To observe the current through the motor, we can use a shunt resistor (low value) serial to the motor. By measuring the voltage between the terminals of the shunt resistor, we can get an image of the current for the motor (voltage value).

The position of the shunt resistor in the electrical scheme is so important (see Figure 29), because if the electrical circuit share a common ground with a measurement tool, or a data acquisition device, the shunt resistor must be placed the closest possible from the ground. Otherwise the shunt resistor could add a common mode voltage that may result from getting wrong measurement.

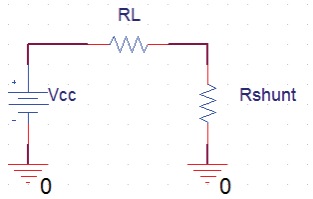


Figure 29: How to place the shunt resistor

RL : motor impedance
often $R_{shunt} \ll R_L$ *and* $R_{shunt} = 1\Omega$

$$I = \frac{Vcc}{R_L + R_{shunt}} \cong \frac{Vcc}{R_L}$$

$$V_{shunt} = R_{shunt} * \frac{Vcc}{R_L + R_{shunt}} \cong R_{shunt} * \frac{Vcc}{R_L} = R_{shunt} * I$$

As $R_{shunt} = 1\Omega$ V_{shunt} is the current image through the motor
R_{shunt} is a power resistor.

The current value through the motor can be between [-10A 10A] which means the voltage value between the terminals of the shunt resistor can be between [-10V 10V]. It is

necessary to resize this value to be able to use it. We suggest to divide this value by 20 to have a value of the current image between [-0,5V 0,5V].

By using Emitter follower we can isolate the impedance from each part. The tension divider bridge is responsible to divide the current image value (by 100 in our case, see the result in Figure 31). This value will be explained later.

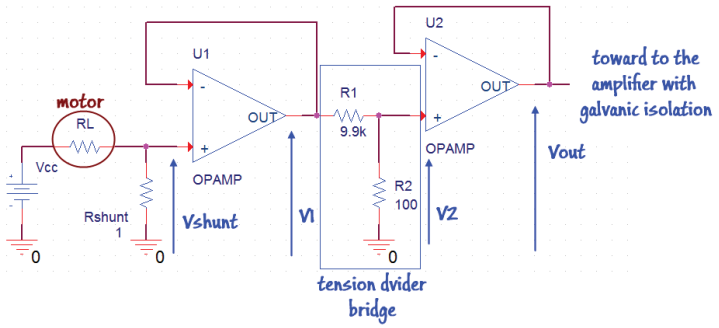


Figure 30 : First part of the electrical scheme for the current motor measurement

$$tension\ divider\ bridge:\ \frac{V2}{V1} = \frac{R2}{R2+R1} = \frac{1}{100} \quad we\ can\ choose\ R1 = 9{,}9k\Omega\ \ and\ R2 = 100\Omega$$

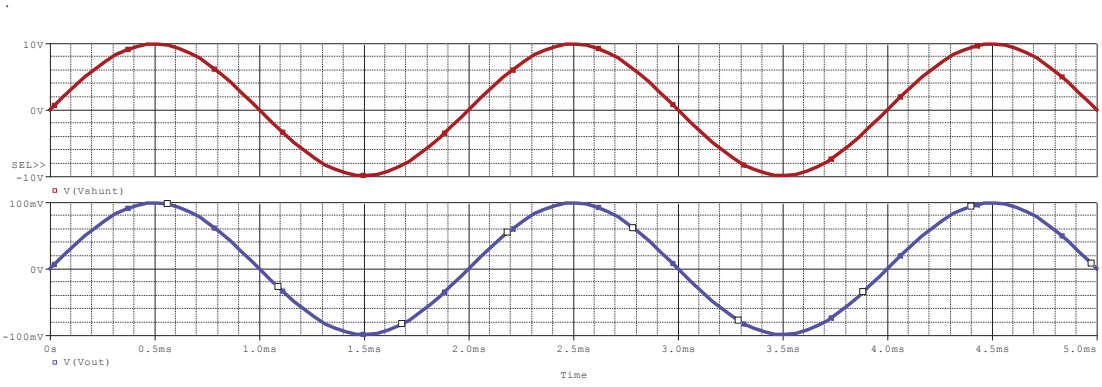


Figure 31: PSpice Simulation V2= Vout = V1/100 =Vshunt/100

2.2.3 Amplifier with Galvanic isolation

The amplifier with galvanic isolation aims to send the analog signal between 2 parts with no common ground (here the image of the current motor comes from the power section, and should be transmitted to MYRIO command section which they have different ground reference).

To implement the galvanic isolation, we will use the opto-coupler component: the IL300.

The IL300 opto-coupler component

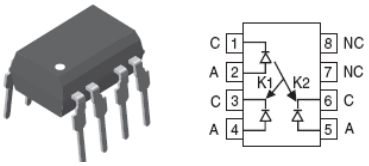


Figure 32: Taken from a web page

Main feature	
Work with the signals	AC et DC
Servo linearity	0.01%
bandwidth	< 200KHz
Transfer Gain (ΔK3)	0,557
Transfer Gain linearity	1,618

The IL300 is an electrical device that has 2 independent electrical elements, however they are optically coupled. This component allows transferring information between 2 parts that are electrically isolated. The first element is an emitter that produces light, which is why it is called opto. The second element is a sensible receiver to the emitting light. The connection between these both elements work completely with light.

Opto-coupler can be used for:

- isolate electrically some parts (power part to the command part)
- Avoid ground loop

Suggested scheme for the IL300 by the constructor

The IL300 datasheet from "Vishay Semiconductors" suggests an application that can allows us to using current, generated by one operational amplifier, to control the LED, located in this opto-coupler. The following electrical scheme is shown in Figure 33.

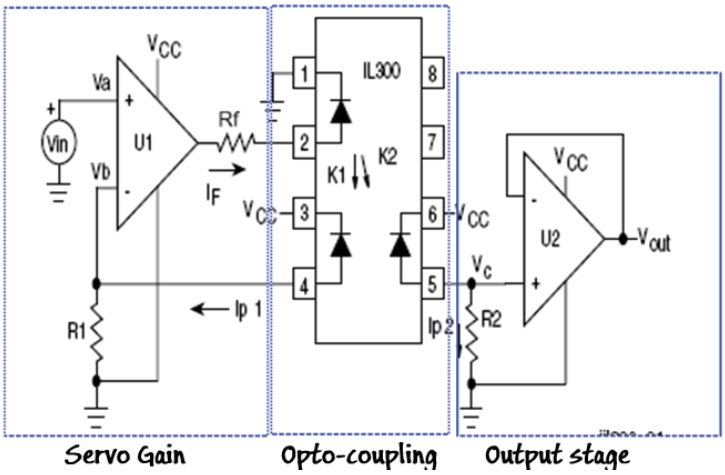


Figure 33: scheme from the IL300 datasheet

Principle

I_{p1} is the feedback current from the photodiode : $I_{p1} = \frac{Vin}{R1}$

I_{p1} is proportional to the feedback value of the transfer gain K1 of the LED :

$$I_{p1} = \frac{Vin}{R1} = K1 * I_F \qquad R1 = \frac{Vin}{K1 * I_F}$$

The OPA placed at the entrance, should provide enough current for the LED to maintain the relation $V_a = V_b$. This is the role of the Servo gain part

$$Rf = \frac{Vin - Vdiode}{If}$$

The output photodiode is connected to the non-inverting amplifier stage (to isolate the impedance). The generated current from the photodiode I_{p2}, the resistor R_2 (we consider the opamp as ideal) allows it to convert the output signal from the opto-coupler, current type, to a voltage signal.

The tension V_{out} depends to the gain output value K_2 (Forward gain), to current from the photodiode and to the load R_2.

$$Vout = If * K2 * R2 \qquad K2 = \frac{I_{p2}}{If}$$

$$\frac{Vout}{Vin} = \frac{If*K2*R2}{If*K1*R1} = \frac{K2*R2}{K1*R1} \qquad output\ transfer\ gain$$

The Transfer gain K_3 doesn't depend to the current value I_f : $K3 = \frac{K2}{K1}$

For the output stage, we have $V_{out}= V_{b2}$ $V_{out}=V_{a2}$

$$Vout = I_{p2} * R2 \quad or \quad I_{p2} = If * K2 \qquad R2 = \frac{Vout}{If*K2}$$

$$Vout = R2 * If * K2 = R2 * K2 * \frac{Vin}{R1*K1} \qquad Vout = Vin * \left(\frac{R2*K2}{R1*K1}\right) \quad or \quad K3 = \frac{K2}{K1}$$

$$Vout = Vin * K3 * \frac{R2}{R1}$$

We can observe that V_{out} is proportional with the relation $\frac{R2}{R1}$.

IL300 study

To use the opto-coupler, we must know these 3 values:

- $K1 = \frac{I_{p1}}{If}$ *Servo Gain*
- $K2 = \frac{I_{p2}}{If}$ *Forward Gain*
- $K3 = \frac{K2}{K1}$ *Transfer Gain*

Simulation

We are seeking for I_f value which will excited the diode and will trigger the other diodes placed on port 4 and 5. We simulate the following scheme (Figure 34).

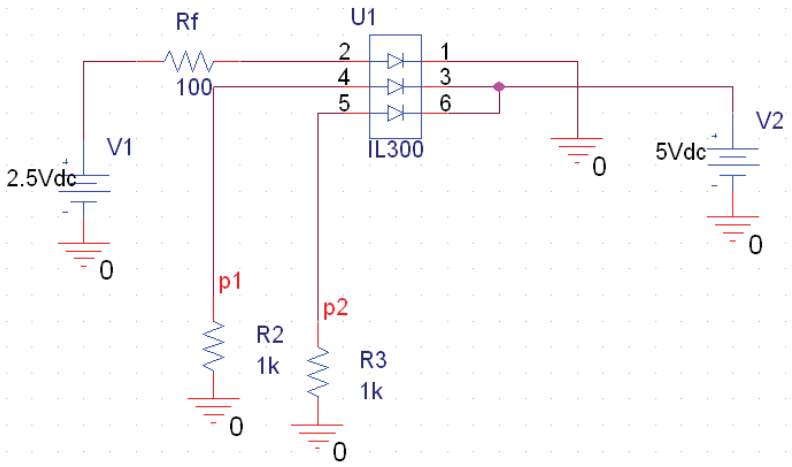


Figure 34: study scheme for the IL300

The voltage V1 value was varied in order to find for what voltage value we have the current I_{p1} and I_{p2}. From the datasheet, with 25°C and when I_f = 0,8mA we have a current I_{p1} (linearity). We must have a minimal current I_f equal to 0,8mA.

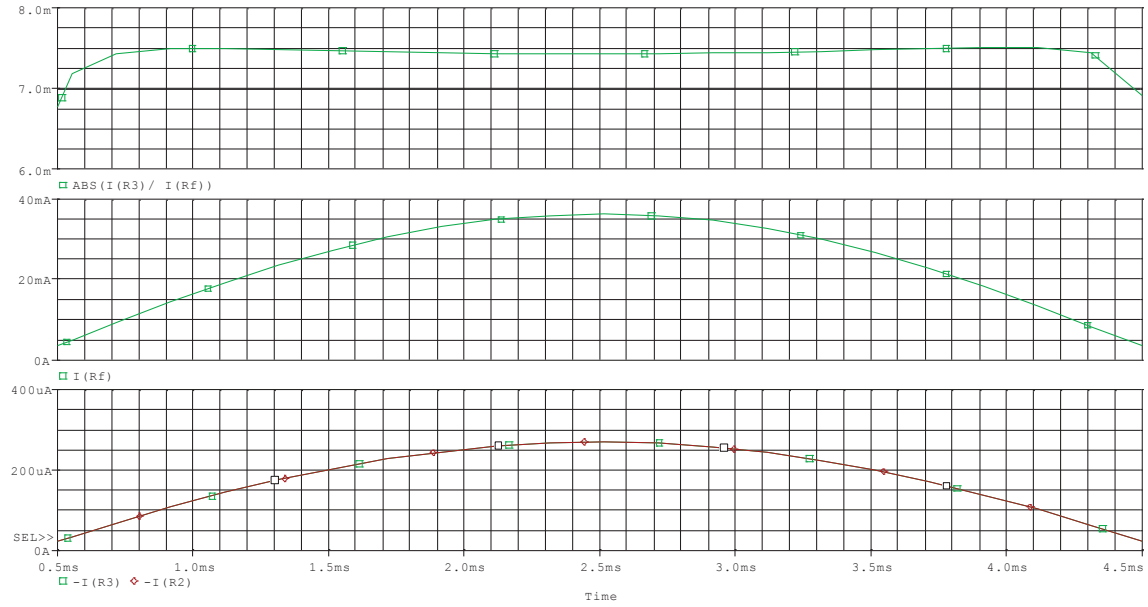


Figure 35: simulation result from PSPICE software

We applied a sine waveform V_{sin} instead of the dc signal V_{dc} (see Figure 35). We observe the evolution of the current I_{p1} and I_{p2} in function to the current I_f. We have I_{p1}=I_{p2}. The evolution of the curve for the current I_f and I_p are the same, there is a linearity relation between both variables. The factor values are:

$$K_1 = \frac{Ip1}{If} = 0{,}00749 \qquad K_2 = \frac{Ip2}{If} = 0{,}00749 \qquad K_3 = \frac{K2}{K1} = 1$$

From the simulation result, the linearity region for K_1,K_2 and K_3 is when Irf≥10 mA, being Ip1>0 ,075mA. We have the gain values for the ideal case, now we have to verify these values by a true experimentation, in real case.

Experimentation

We keep the same electrical scheme from PSPICE. We want to know in the real case the value of K_1, K_2, K_3. We get different values for different input values $V_{entrée}$:

Table 7: Experimentation result

Ventrée	Ip1 (mA)	Ip2 (mA)	K1*1000	Delta K1 en %	K2	Delta K2 en %	K3	DeltaK3
2,5	0,08	0,08	7,14	4,72	6,56	12,41	0,92	8,07
3,8	0,17	0,16	6,82	8,93	6,29	15,98	0,92	7,74
4,5	0,22	0,2	6,8	9,25	6,23	16,85	0,92	8,37

We can see (Table 7: Experimentation result Table 7) that K_3 value is unchangeable and equal to 0.92, a different value that we have in our theoretical analysis (that was equal to 1).

We can also see that the linear region for these 3 gains are located around I_{p1}=0,17 mA.

2.2.4 Realization of the amplifier with galvanic isolation

After the IL300 study, we will study each construction stage of the amplifier with galvanic isolation.

The final scheme of this part is shown in Figure 36.

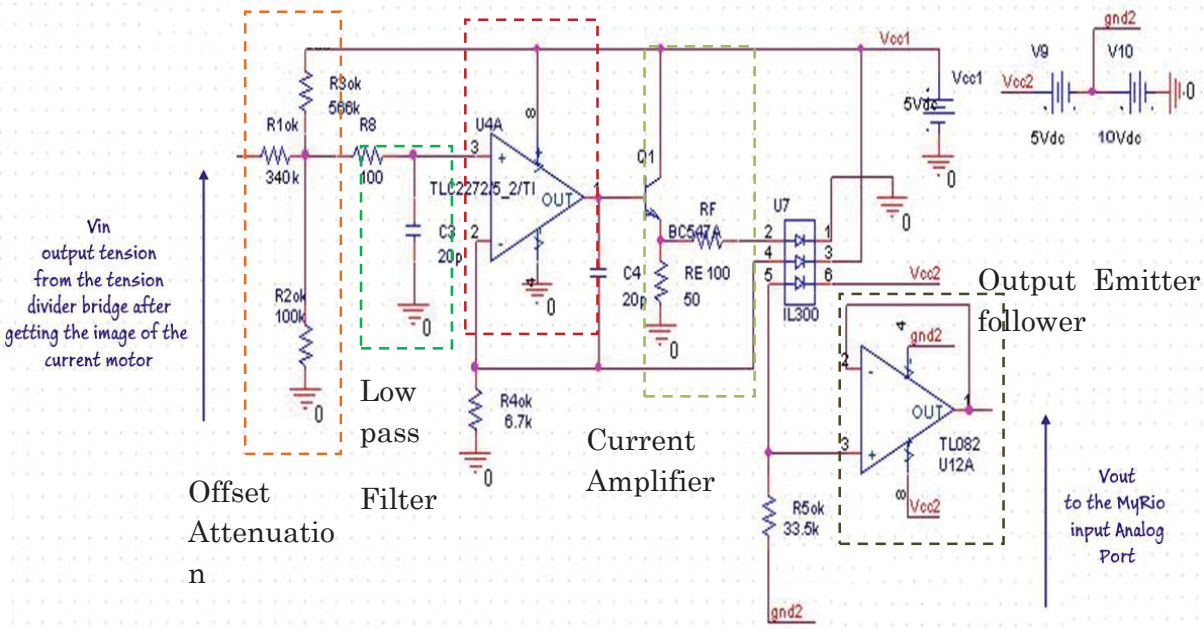


Figure 36: Final schema for the galvanic isolation

Signal attenuation and offset

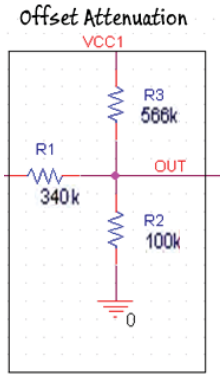


Figure 37: Offset and attenuation stage

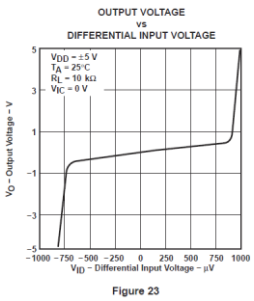


Figure 38: Threshold figure from the TLC2272 datasheet

From the datasheet of the TLC2272 amplifier (Figure 38), this component has a threshold voltage around 1mV. In experiment, we found the same value. We have to add an offset stage to overcome this default (Figure 37).

To ensure that the photodiode's opto-coupler can emit a light, the current through this device should be positive (direction diode convention). But the input signal should be between [-0,5V 0,5V]. It is necessary to add an offset stage and attenuate the alternative part. We will use an electrical scheme like a tension divider bridge.

Vout Expression

The output signal is the sum of attenuated Vin signal (alternative part) and the offset value (continuous part).

$$V_{OUT} = V_{OUT\_DC} + V_{OUT\_AC}$$

Vout expression can be found by using the superimposing principle.

To calculate V_{OUT_AC}, the continuous source must equal to 0V, therefore V_{in}=0. We obtain the following formula :

$$V_{OUT\_DC} = \frac{R_2//R_1}{R_3 + R_2//R_1} V_{cc1}$$

$$V_{OUT} = \frac{R_2//R_3}{R_1 + R_2//R_3} V_{in} + \frac{R_2//R_1}{R_3 + R_2//R_1} V_{cc1}$$

Resistors calculation:

$V_{OUT\_DC} = \frac{R_2//R_1}{R_3+R_2//R_1} V_{cc1}$ ←→ $\frac{V_{cc1}}{V_{OUT\_DC}} = \frac{R_3+R_2//R_1}{R_2//R_1}$ ←→ $R_3 = R_2// R_1\left(\frac{V_{cc1}}{V_{OUT_{DC}}} - 1\right)$

$V_{OUT\_AC} = \frac{R_2//R_3}{R_1+R_2//R_3} V_{in}$ ←→ $\frac{V_{in}}{V_{OUT\_AC}} = \frac{R_1+R_2//R_3}{R_2//R_3}$ ←→ $R_1 = R_2//R_3\left(\frac{V_{in}}{V_{OUT_{AC}}} - 1\right)$

We decided to fixe $R_2 = 100k\Omega$

$R_3 = \frac{R_2R_1}{R_{2+}R_1}\left(\frac{V_{cc1}}{V_{OUT_{DC}}} - 1\right)$ $R_1 = \frac{R_2R_3}{R_{2+}R_3}\left(\frac{V_{in}}{V_{OUT_{AC}}} - 1\right)$

We obtain the R_3 expression :

$$R_3 = \frac{R_2 \frac{R_2R_3}{R_{2+}R_3}\left(\frac{V_{in}}{V_{OUT_{AC}}} - 1\right)}{R_{2+} \frac{R_2R_3}{R_{2+}R_3}\left(\frac{V_{in}}{V_{OUT_{AC}}} - 1\right)}\left(\frac{V_{cc1}}{V_{OUT_{DC}}} - 1\right) = \frac{V_{OUT_{AC}}}{V_{in}}\left[R_2\left(\frac{V_{in}}{V_{OUT_{AC}}} - 1\right)\left(\frac{V_{cc1}}{V_{OUT_{DC}}} - 1\right) - R_2\right]$$

In our case we want to have an offset value to 0,6V and the alternative part equal to 0,1V. The choice of the offset and the attenuation value will explain later.

$\frac{V_{OUT\_AC}}{V_{in}} = \frac{1}{5}$ $V_{OUT_{DC}} = +0{,}6V$ V_{cc1}=+5,0V

$R_3 = 566{,}6k\Omega$ we obtain $R_1 = 340k\Omega$

PSpice simulation

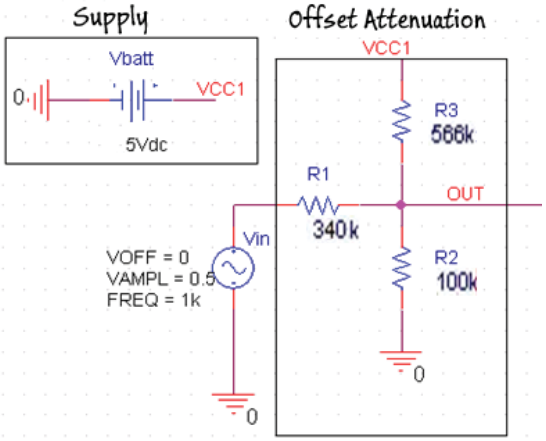


Figure 39: scheme of the simulation in PSPICE

By using the resistors values that we found previously to get an offset to +0,6V and an alternative par to 0,1V, we have in the Figure 40:

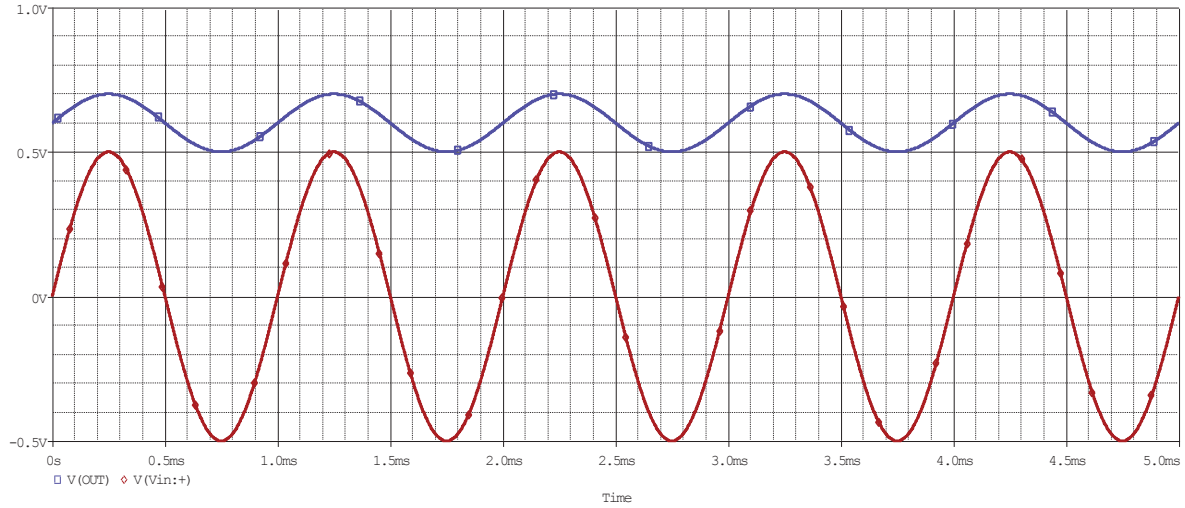


Figure 40: simulation result from PSPICE
(Calibration of the components value for the Offset Attenuation Stage)

Servo Gain and amplifier current stage

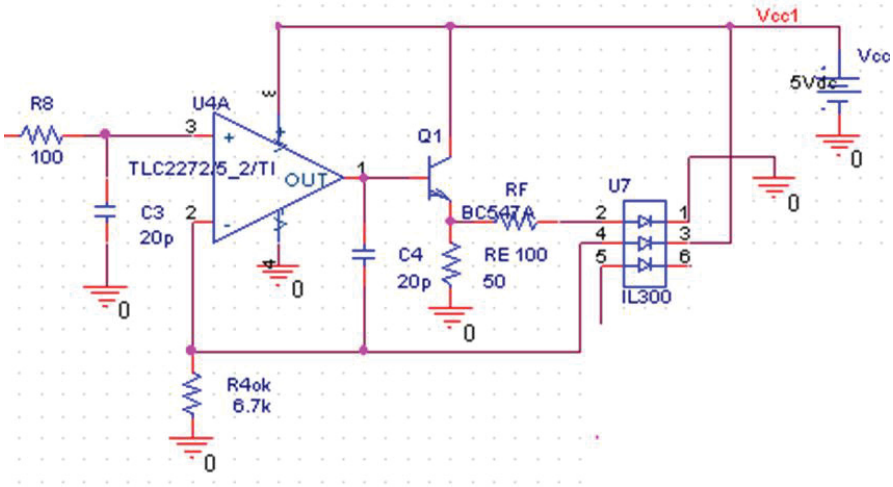


Figure 41: servo and amplifier current electrical scheme

To be in the linearity region of the opto-coupler, the current value *If* should be at least *If*>8mA. But the Opamp TLC2272 cannot provide more than 2mA, it is necessary to add a current amplifier stage (see Figure 41).

To resolve this default, a transistor - Common Emitter can be added. The ß value (or hfe) of the transistor (here a BC547B) will amplified the current output value from the AOP. The components R8, C3 and C4 are used to stabilize the system, that will explain later in the part "gain and stability study".

We saw that we need at least *If*=8mA, we will take 10 mA. The maximal current value is 22mA, from the datasheet. We will try to apply these values to the diode current.

1st try:

We simulate the "attenuation and the offset" stage by generating an input sine waveform with an alternative part equal to 0,25V and an offset equal to 1V:

We obtain the result from PSPICE (Figure 42).

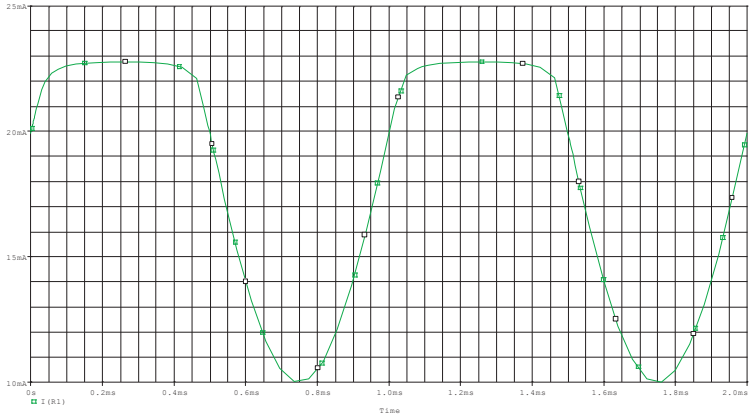


Figure 42: current I_f result with an input signal (offset +1V and an alternative part equal to 0.25V)

We can see that the current thought the diode is superior to 10V but we can observe saturation after 22mA. In fact, from the IL300 datasheet, the diode saturates after this value. The amplifier value of the Common Emitter is too high.

2nd try:

We try to change the resistors values to avoid the signal saturation (to not exceed 22mA) by changing the offset value of the input signal. We try with V_{off} = 0,75V (>0,5V) and V_{ampl}=0,25V.

We obtain the result from PSPICE (Figure 43).

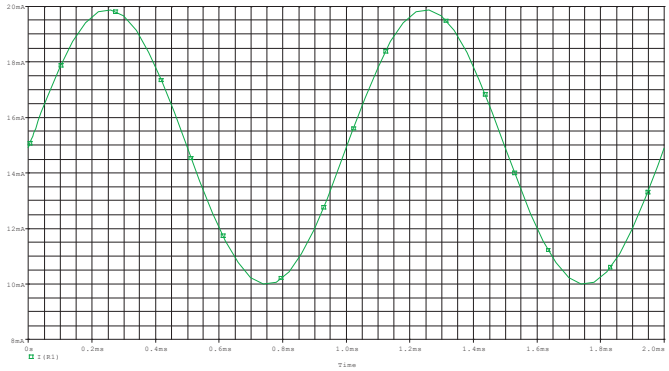


Figure 43: current I_f result with an input signal (offset +0,75V and an alternative part equal to 0.25V)

The current result from the simulation through the diode is now correct. But after realizing experimentation, we have observed a little saturation from the output signal. This default comes from the ß value of the transistor that was a bit higher than the value used for the simulation. We have to change the offset and the attenuation value of the input signal again

3rd try:

We try with V_{off} = +0,6V (>0,5V) and V_{ampl}=0,1V.

We obtain the result from PSPICE (Figure 44).

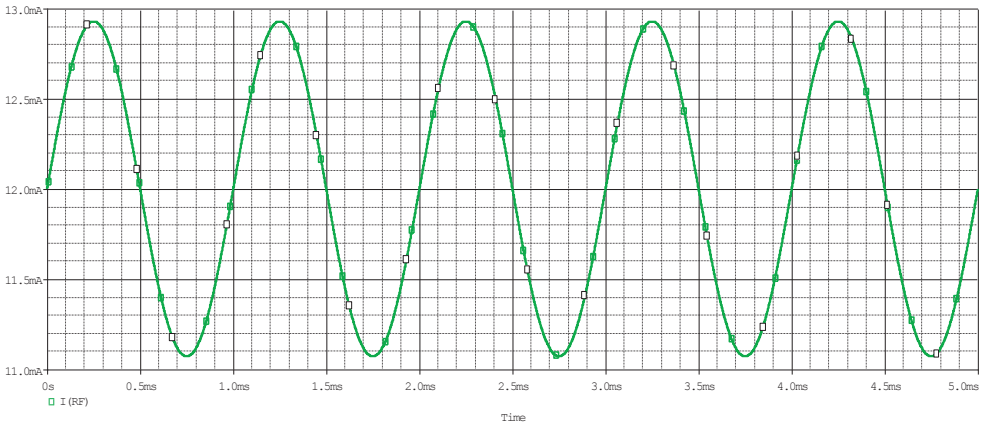


Figure 44: current I_f result with an input signal (offset +0,6V and an alternative part equal to 0.1V)

We stay this time in the linearity region of the opto-coupler. After realizing experimentation, the output signal from the opto-coupler is correct, there is no deformation of the signal observable.

Output Stage of the Opto-coupler

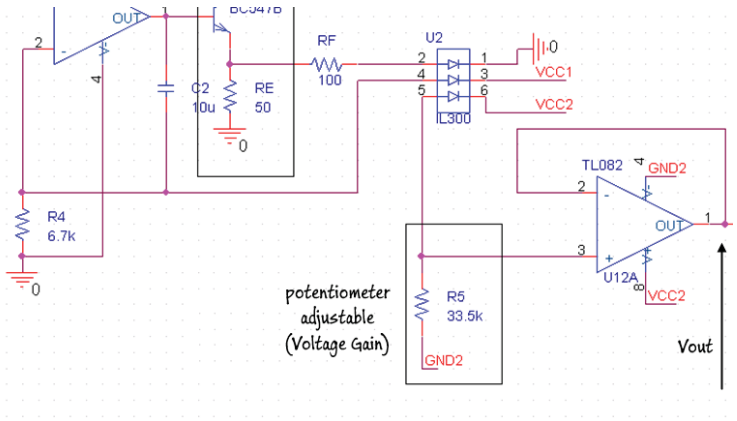


Figure 45: Output Stage

The output stage has 2 parts (see Figure 45):

- A potentiometer R5, to control the transfer gain value K_3.
- An emitter follower, to isolate impedance

Calculation of the resistor R5

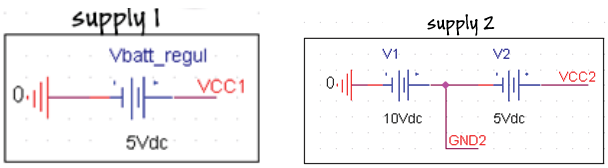
We know that $Vout = Vin * K3 * \frac{R5}{R4}$ with K3=1.

But R_4=6,7kΩ and Vin_alt= 0.1V and we would like to have Vout_alt=1vpp

R5= 5xR4= 5 x 6,7k = 33,5kΩ

Introduction of the second ground

The amplifier with galvanic isolation must be able to send an analog signal between 2 parts that were electrically isolated (don't share the same ground reference). To simulate this situation on PSPICE, we create 2 grounds reference:



The potential difference between both grounds is 10V here.
L'alim1 : from the supply of the power part
L'alim1 : from the supply of the command part

Simulation

We use PSPICE for simulation. With an input signal 1VPP and with the resistor R5 value which we previously found we will have the new result (Figure 46).

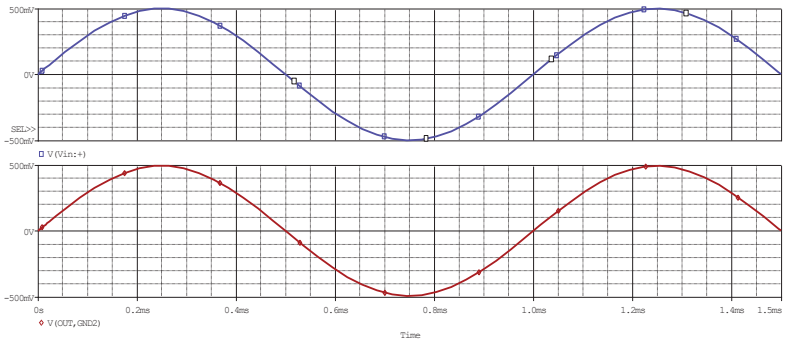


Figure 46: PSPICE simulation result (test by using two different grounds references)

We have $V_{IN=}\ V_{OUT\ AC}$ with the galvanic isolation. The galvanic isolation of the opto-coupler is operating. We get the same value of the input signal to the output signal.

Gain and stability study

The stability and the gain study of the amplifier with galvanic isolation is shown in Figure 47 and Figure 48 (from PSPICE).

Simulation1:

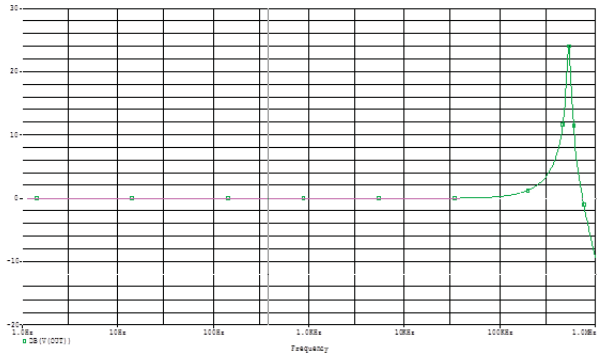


Figure 47: PSPICE result Bod diagram

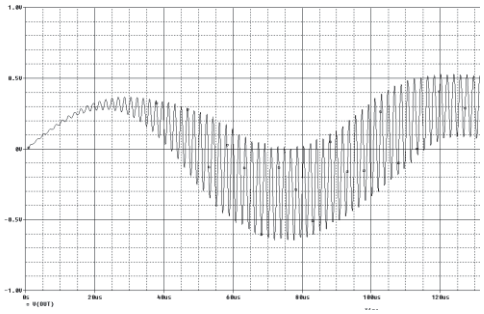


Figure 48: PSPICE result *time domain: instable signal output*

The circuit works for some frequency value, but there is a high gain value that shows us instability. We have an important amplitude peak around the frequency 80 KHz.

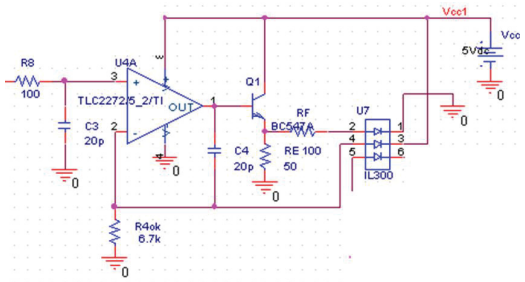


Figure 49: suppression of the instability by adding C2, R8, and C4 components (filter function)

To reduce the instability, we can place a capacitor C4 between the Opamp output and before the current amplifier (Figure 49). This capacitor can compensate a pole and to stabilize the circuit without adding disturbance in the bandwidth.

Simulation 2:

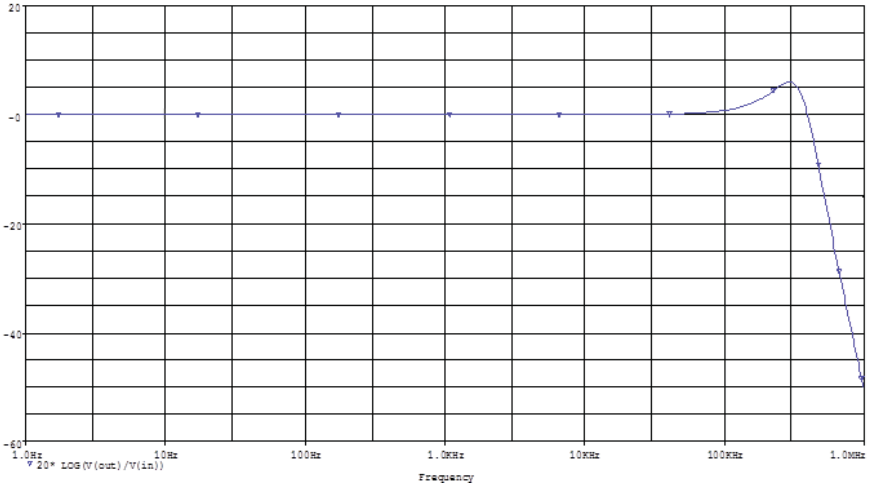


Figure 50: Gain and stability study after adding the capacitor C4

We have improved the stability of our system by reducing the peak value, but still be present (see Figure 50). We will placed a low pass filter (block "low pass filter" on the final electrical scheme) (R8 with C3) to suppress the peak by limiting the bandwidth. To have a limiting frequency to 80 KHz, we will choose R8=100 and C=20pF.

Simulation 3

We realize another simulation with the low pass filter added to our scheme (Figure 51).

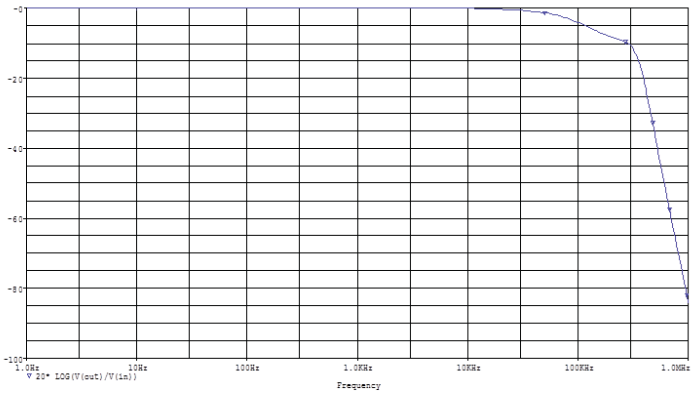


Figure 51: Gain and stability study after adding the capacitor and the resistor C2 and R8

The simulation (Figure 51) Result shows us the suppression of the peak.
The bandwidth of our system is 100KHz)

THD Analysis

We calculate the THD (Total Harmonic Distortion) on PSPICE (Figure 52).

FOURIER COMPONENTS OF TRANSIENT RESPONSE V(OUT)

DC COMPONENT = -7.945486E-05

HARMONIC NO	FREQUENCY (HZ)	FOURIER COMPONENT	NORMALIZED COMPONENT	PHASE (DEG)	NORMALIZED PHASE (DEG)
1	1.000E+04	4.983E-01	1.000E+00	-6.913E+00	0.000E+00
2	2.000E+04	2.372E-04	4.760E-04	1.650E+02	1.789E+02
3	3.000E+04	1.513E-05	3.036E-05	7.133E+01	9.207E+01
4	4.000E+04	5.288E-06	1.061E-05	-4.419E+01	-1.654E+01
5	5.000E+04	2.578E-06	5.174E-06	1.541E+02	1.886E+02
6	6.000E+04	2.396E-06	4.809E-06	-1.157E+02	-7.424E+01
7	7.000E+04	1.987E-06	3.987E-06	-7.510E+00	4.088E+01
8	8.000E+04	3.096E-06	6.213E-06	1.754E+02	2.307E+02
9	9.000E+04	1.328E-06	2.665E-06	-8.564E+01	-2.341E+01

TOTAL HARMONIC DISTORTION = 4.771657E-02 PERCENT

THD for 10 KHz

FOURIER COMPONENTS OF TRANSIENT RESPONSE V(OUT)

DC COMPONENT = -7.881497E-04

HARMONIC NO	FREQUENCY (HZ)	FOURIER COMPONENT	NORMALIZED COMPONENT	PHASE (DEG)	NORMALIZED PHASE (DEG)
1	1.000E+03	4.998E-01	1.000E+00	-5.981E-01	0.000E+00
2	2.000E+03	3.428E-05	6.859E-05	1.369E+02	1.381E+02
3	3.000E+03	2.497E-05	4.995E-05	9.091E+01	9.271E+01
4	4.000E+03	2.274E-05	4.549E-05	8.997E+01	9.236E+01
5	5.000E+03	2.263E-05	4.528E-05	8.976E+01	9.275E+01
6	6.000E+03	2.287E-05	4.575E-05	8.935E+01	9.293E+01
7	7.000E+03	2.286E-05	4.573E-05	8.849E+01	9.268E+01
8	8.000E+03	2.271E-05	4.543E-05	8.835E+01	9.314E+01
9	9.000E+03	2.275E-05	4.552E-05	8.836E+01	9.375E+01

TOTAL HARMONIC DISTORTION = 1.401482E-02 PERCENT

THD for 1KHz

Figure 52: PSPICE result: THD Analysis for 1 KHZ and 10 KHZ

At 1KHZ the THD= 0,014%

At 10KHz the THD= 0,048%

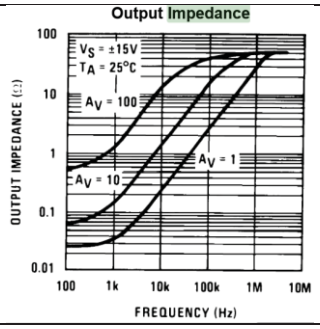
The experimental result gives us:

THD at 1KHz = 0,049%

THD at 10KHz= 0,05%

2.2.5 Conclusion: Feature of the current measurement electric scheme

Table 8: Main Feature of the current measurement electric scheme

Output Voltage (image of the current in the motor)	(10mV for 1mA) + Voffset 1mA: current in the motor
Voffset	+0,6V
Vshunt maximal voltage recommended	abs(10V)
Galvanic Isolation	Yes
Output impedance (from the TL082 datasheet)	
THD (Total Distortion Harmonic)	For 1kHz: 0,049% For 10kHz: 0,05%

3 Vision System

Introduction

There are many ways to getting the angular position of the second limb by vision system. But In our case, some feature are very important: the image processing should be very fast, with a good resolution (like 0.5 degree of accuracy) and can be easily implemented on the board MyRIO. We will present on this report some technique that we studied to detecting the angular position.

3.1 First method: The Hough transform [18]

The Hough transform, created by Paul Hough in 1962, is an algorithm developed to detect simple geometric figures like circle, ellipse, and straight line.

3.1.1 Hough Transform principle

Let $\mathbb{R}^n$ be the study Image space, and Ɛ be the ensemble of all N points selected by the pretreatment

$$\mathcal{E} = \{ M_i, i = 1 \dots N\} \in \mathbb{R}^n$$

One point M from $\mathbb{R}^n$described by his position x

Let $\Omega \subset \mathbb{R}^p$ be the parameter space and F a family of curve on $\mathbb{R}^n$parameterized by "a":

$$F = \{\{x: f(x,a) = 0, x \in \mathbb{R}^p\}, a \in \Omega\}$$

We call the Hough transform associated to the family F, a transformation corresponded to the ensemble Ɛ, a function g defined on Ω.

There are a lot of Hough transform variety, like the transformation "M to 1" and also the transformation "1 to M". We will study both of them in particularly.

3.1.2 Hough Transform M to 1

Let m be the minimal number point on $\mathbb{R}^n$ for describing a waveform F.

Let $\mathcal{E}^m$ be an ensemble of all m-uples from Ɛ.

$$\mathcal{E}^{(m)} = \{ M|^{(m)} = \{M_{i1,}, M_{i2}, \dots, M_{im,}\, M_{ik} \in \mathcal{E}\},\}$$

With the cardinal value: $Card\left(\mathcal{E}^{(m)}\right) = C_N^m$

For all m-uples of $M|^{(m)}$, $\mathcal{E}^{(m)}$ is associated a waveform from F with a_i parameter. Let C(a) the characteristic function of $\mathbb{R}^p$. The Hough transform "M to 1" is defined by:

$$g(a) = \sum_{M|^{(m)} \in \mathcal{E}^{(m)}} c(a - a_i)$$

Application example:

Describing a straight line by "*y=ax+b*" is generally a bit clumsy, because the space (X={a,b}) is strongly inhomogeneous. A polar description is more appropriate a={ρ,σ}. The straight line equation becomes ρ= xcos(σ) + ysin(σ) (see Figure 53).

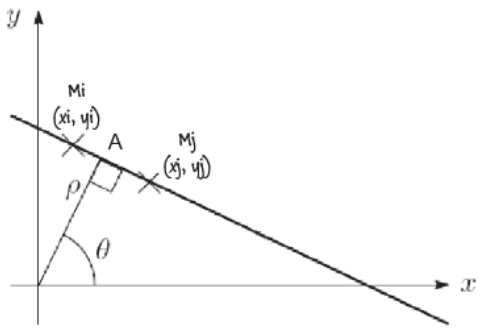


Figure 53: Straight line in Cartesian axes

A straight line is described by his equation ρ= xcos(σ) + ysin(σ). The Hough Transform "M to 1" (here 2 to 1), is associated with the dipole (M_i M_j), the point pt A={ρ_A, σ_A} with the coordinate:

$$\rho = \frac{|x_i * y_j - x_j * y_i|}{\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}} \qquad \sigma = -\text{Actan}\left(\frac{x_j - x_i}{y_j - y_i}\right)$$

3.1.3 Hough Transform 1 to M

For all point $M_i from\ \mathbb{R}^n$ cross m waveform from F. Let A_i the ensemble "a" value such as $f(x_i, a) = 0$

$$A_i = \{a_k = f(x_i, a_k) = 0\}$$

The Hough Transform "M to 1" is defined by:

$$g(a) = \sum_{M_i \in \mathcal{E}} \sum_{a_k \in A_i} c(a - a_k)$$

Application example:

The Hough transform "1 to M" associate all point M_i the ensemble $A_i \subset \Omega$ the straight lines through M_i. These straight lines verify: ρ= xcos(σ) + ysin(σ) (Figure 54). It is a representation of a sine waveform in the Ω space (the Hough plane) (Figure 55). Hopefully Ω is marked and quantified, then m is not infinite in this plane and can be calculate.

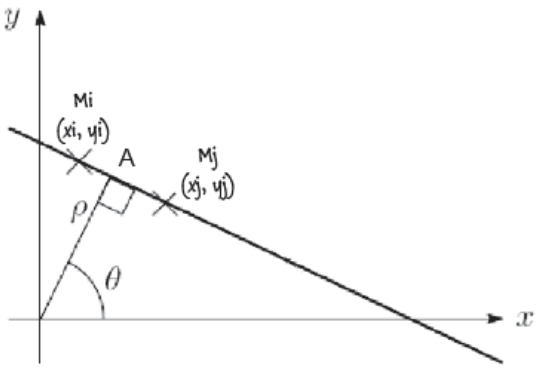


Figure 54: Straight line in Cartesian axes

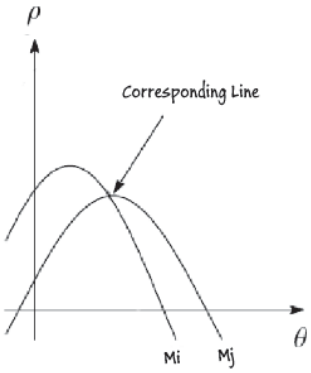


Figure 55: Straight line transposed to the Hough Plane

3.1.4 Operations Processing Chain

To limit the number of point to analyze with the Hough transform, it is necessary to choose the right element to analyze from the picture. We suggest using a rectangle with an insensitive color placed on the second limb (see Figure 56). We will apply a colored filter to extract this element from the picture and after that an edge extractor, like Canny Edge, will be applied in order to get only the position of the edge of the rectangle.

The Hough operation returns a result between [-90° to 90° degree]. It is necessary to use another reference point in order to calculate entirely the angular position of the second limb. We suggest to using a blue marker placed on the pivot pendulum (pivot between the 2 limbs).

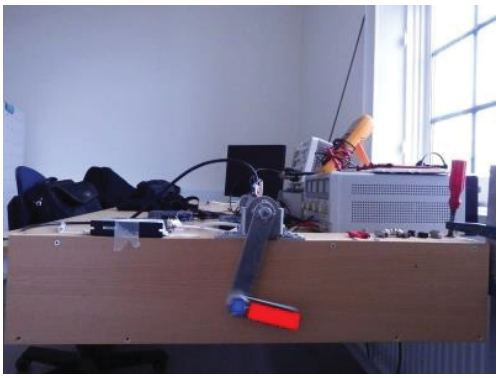


Figure 56: inverted pendulum with red and blue marker added with Paint: picture jpg (400x300)

The general operating processing chain is shown in Figure 57.

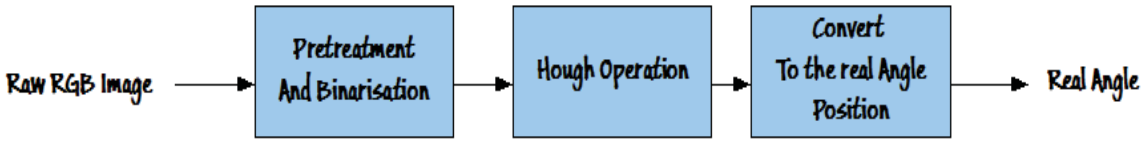


Figure 57: Hough processing chain

By adding the blue and red markers detector, we have in Figure 58:

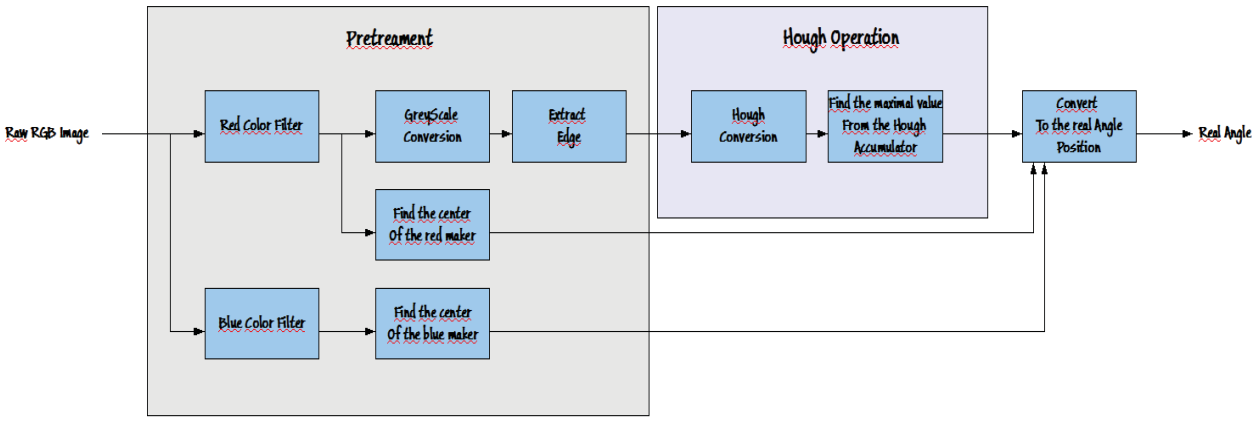


Figure 58: Hough processing chain

3.1.5 Implementation on MATLAB

Some Hough function already exist on Matlab, but they are not optimized for our application (they used a lot of memory and resource and calculation that are not needed), and too hard to implement on LABVIEW. We wrote our own Hough transform code, see Appendix 2, by being aware that it will be implemented on NI MYRIO. The calculations used are often basic transport operation (add, subtract …).

Result with the Hough Transform "M to 1"

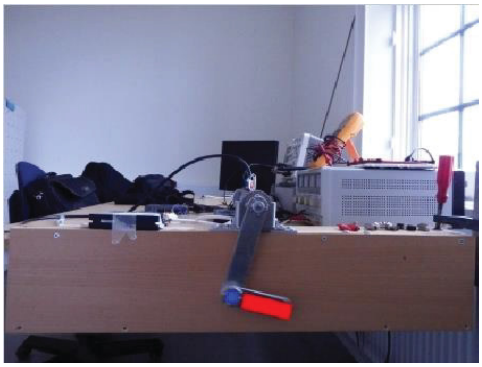


Figure 59: inverted pendulum with red and blue marker added with Paint: picture jpg (400x300)

Figure 60: result after the pretreatment (color filter + greyscale conversion + edge detector) (from our Matlab project)

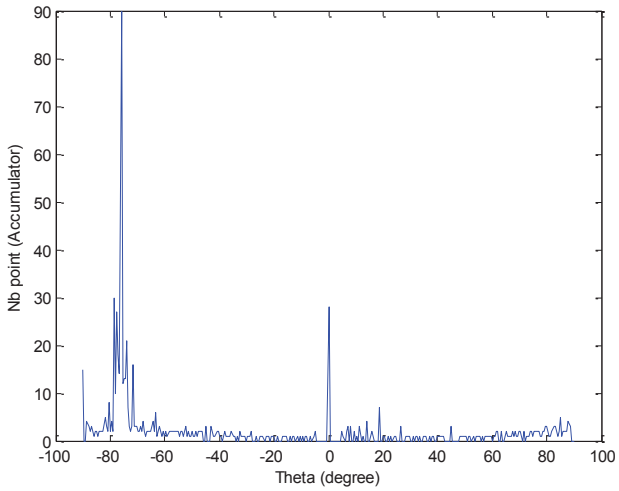


Figure 61: Accumulator result from the Hough Transform (we only keep the maximal value of the accumulator for each angle). The Hough operation returns us an angle result about the angular position of the second limb equal to -76,5° (with 0,5° degree of accuracy). By knowing the position of the red and the blue marker, we can calculate the true angle value: 76,5 degree (with 0,5 degree of accuracy)

Hough Transform "1 to M":

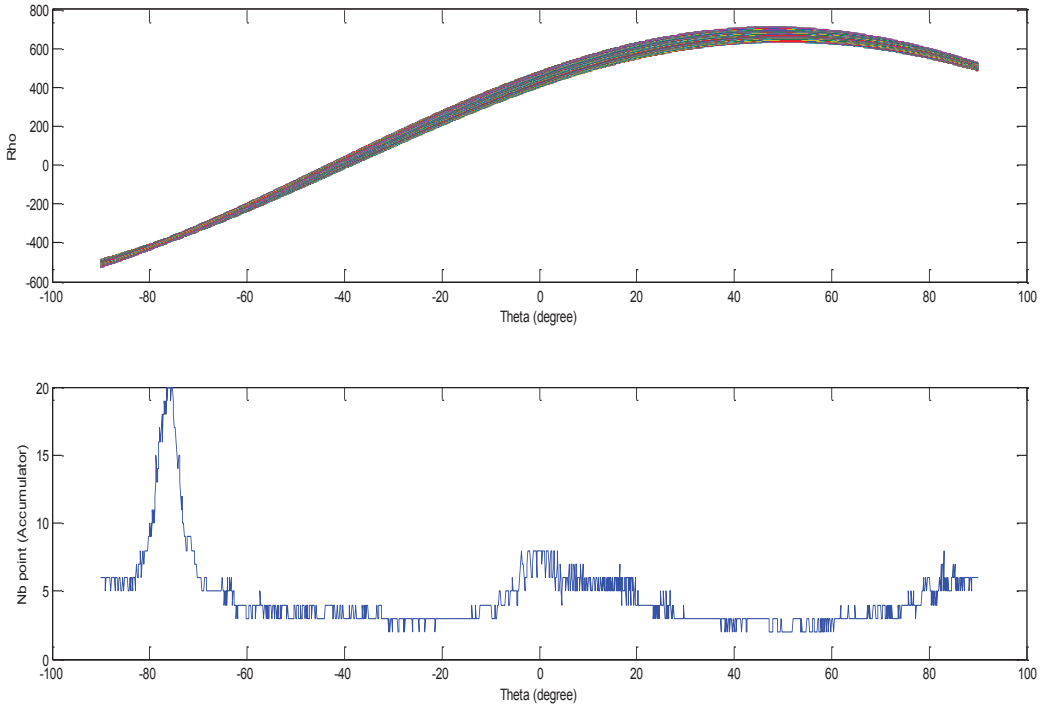


Figure 62: Accumulator result from the Hough Transform "1 to M"

The upper plot (Figure 62) represents the Hough transform result in the space Ω. The lower plot represents the accumulator results getting from the Hough transform result (we only keep the maximal value of the accumulator for each angle). The Hough operation returns us an angle result about the angular position of the second limb equal to -76,5° (with 0,5° degree of accuracy) (Figure 62). By knowing the position of the red and the blue marker, we can calculate the true angle value: 76,5 degree (with 0,5 degree of accuracy).

Conclusion

The Hough transform "M to 1" may be deduced from all junctions of the sine waveforms from the Hough Transform "1 to M", that can easily generalize all the parameterize function. In practice, the Hough transform "1 to m" has less heavy operations than his version "M to 1". In fact It avoid all the combinatory research for all the points (less operations). In other part, the Hough transform "1 to M" is more suitable for an implementation to the embedded system, because his structure is easily parallelizable.

For the implementation of the Hough transform on LABVIEW we will use the "1 to M" version.

3.1.6 Implementation of the Hough Transform on LabVIEW

Using Matlab .m files are possible in LabVIEW environment. LabVIEW program can easily manage the multithreading, in a transparent way. LabVIEW divide each task to an executable thread. The complexity of the threads management is entirely transparent when LabVIEW program is executed.

The Hough transform "M to 1" structure is very efficient for the implementation destined to the embedded system, because his structure is easily parallelizable. When we implement this program on LabVIEW, it is important to use several time the same block function in order to divided the tasks.

The parallelizable structure of the implementation of the Hough transform on LabVIEW is shown in Figure 63.

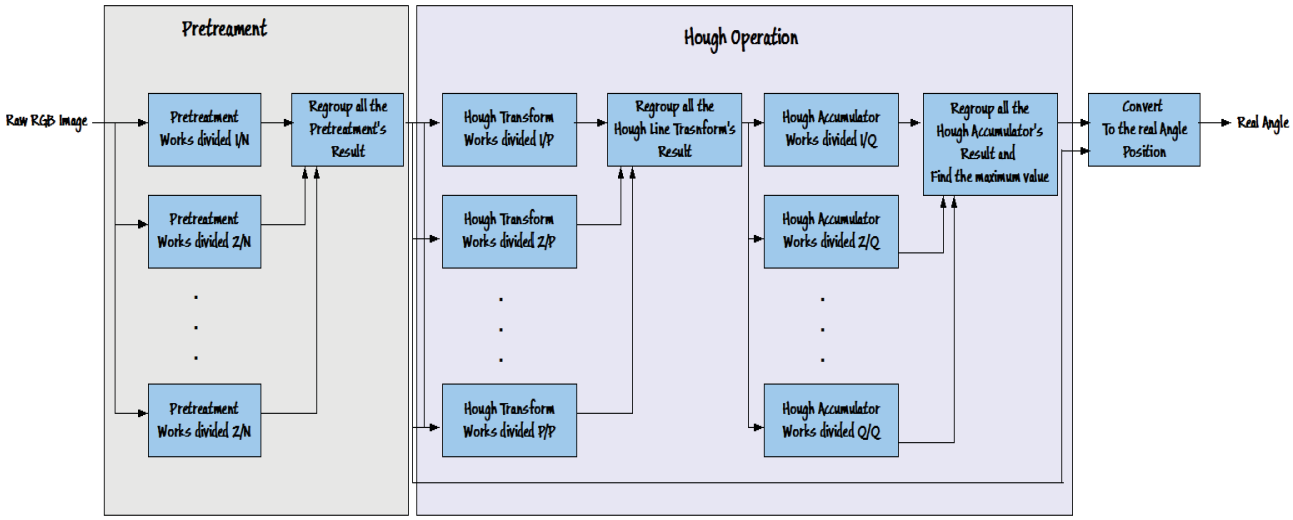


Figure 63: Hough processing chain for LabVIEW implementation

Result:

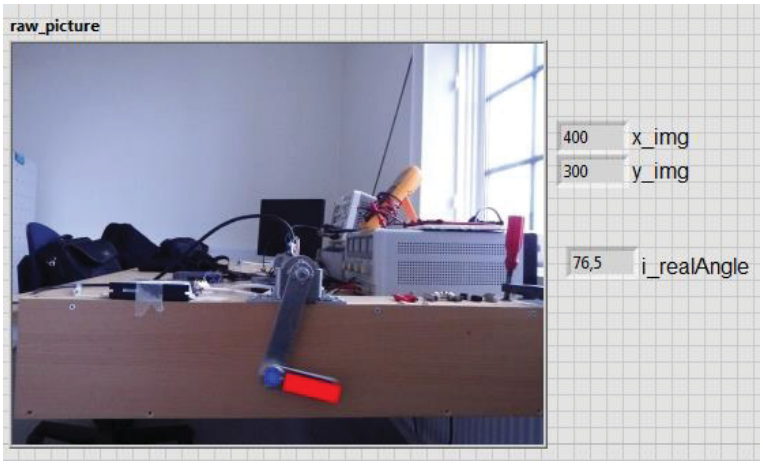


Figure 64 : Test of the implementation of the Hough transform
on MyRIO by loading a jpg image (400x300)

Accuracy:

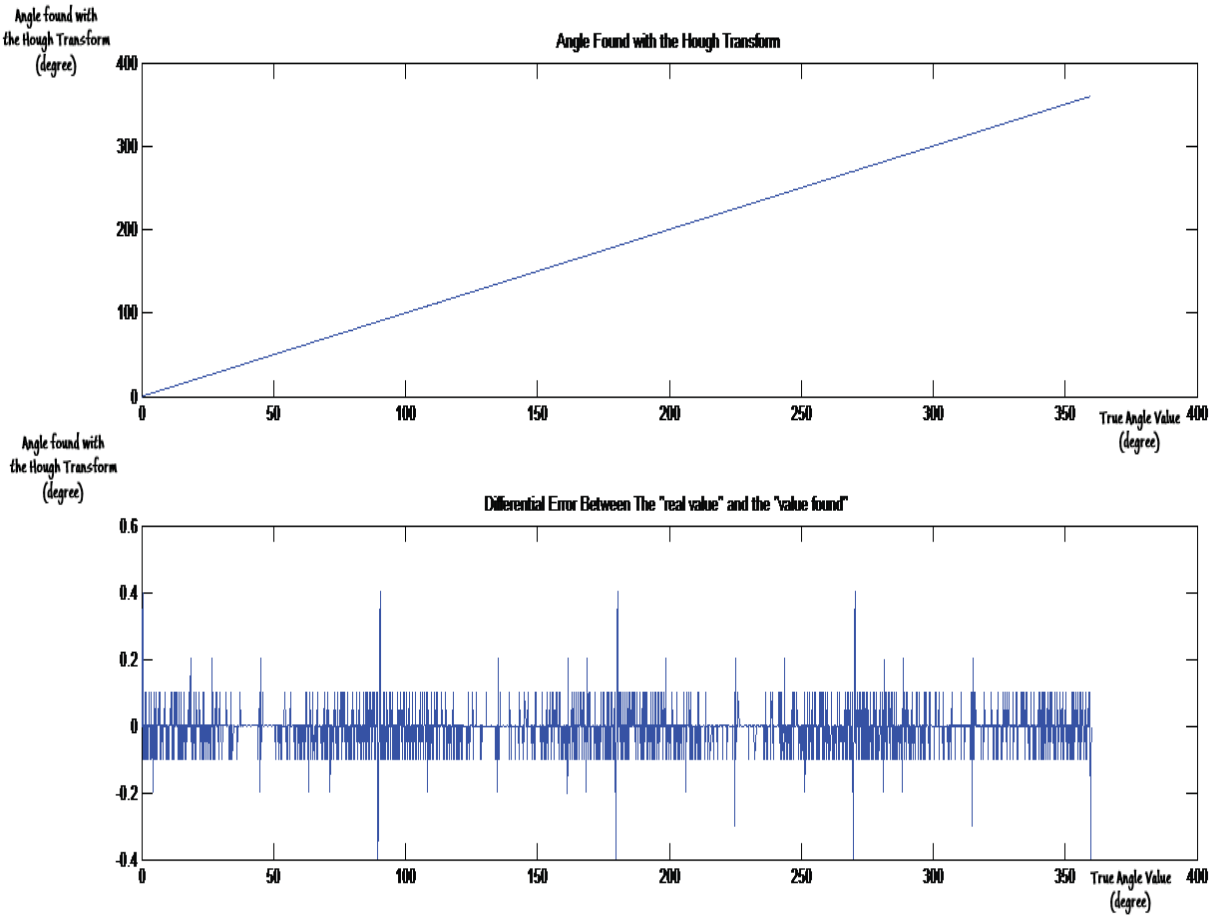


Figure 65: Accuracy for the Hough Transform for each true angle value that we should found (resolution choose for he Hough operation: 0,1 degree). The average error value for each angle value is 0.0303 degree. The maximal error value is 0,4 degree.

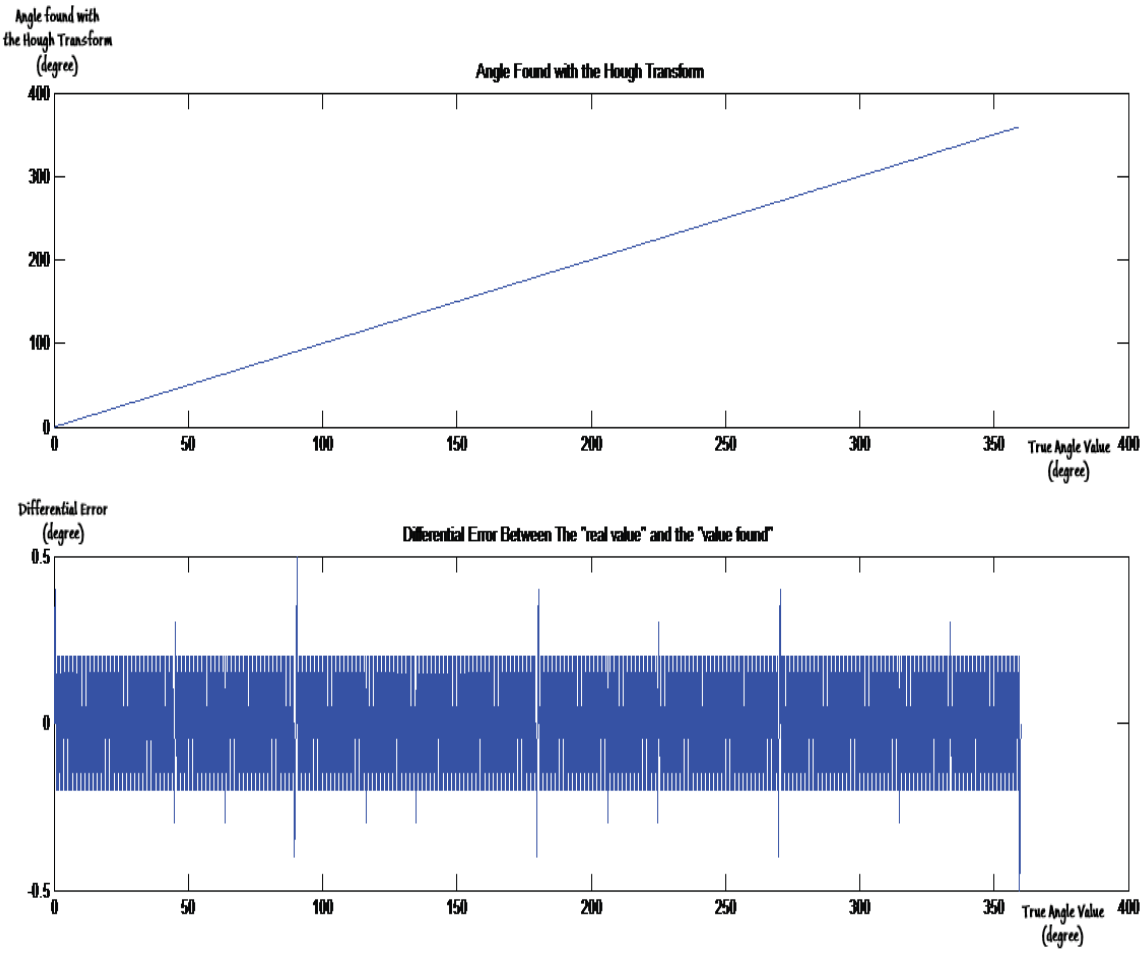


Figure 66: Accuracy for the Hough Transform for each true angle value that we should found (resolution choose for the Hough operation: 0,5 degree). The average error value for each angle value is 0.1214 degree. The maximal error value is 0,5 degree.

Conclusion:

The Hough Transform is a strong image feature extraction technique to find straight lines and their angle of inclination (see result in Figure 66). The disadvantage is about of his utilization that needs a big memory space and an important processing time (can be reduced if we use the "1 to M" version through his parallelizable structure).

The problem comes from the utilization of .m file (Matlab) that provoke a major slowdown for the processing time (we just need 13ms on Matlab, and we need more than 300ms with MyRIO by analyzing a jpg picture 400x300). If we want to implement a Hough Transform on LabVIEW, we must not use .m file and we need to describe all of the functions with the LabVIEW tools.

3.2 Second method: by detecting some markers

Another way to getting the angular position of the second limb is to use marker on the second limb (see Figure 67). This method is less precise than the Hough transform, and need to be initialized with a lot of parameters. But this method doesn't ask a lot of calculation and can have a high time processing.

By knowing the position of the pivot pendulum (we know all the time the angle of the motor arm through the encoder), and if we can find the position of the marker placed on the second limb, we can calculate the angle position of this part.

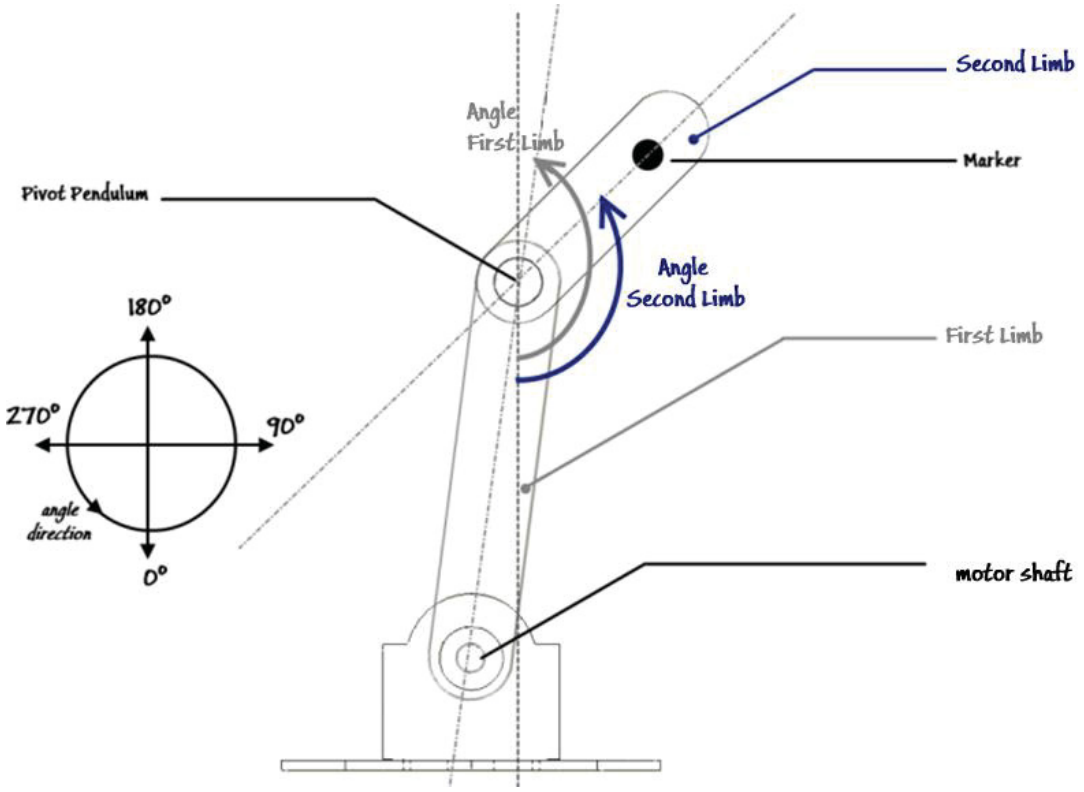


Figure 67: double inverted pendulum

The steps for the calculation of the angle position of the second limb are:

- Calculating the position of the pivot pendulum point by knowing the angular displacement of the first limb and the length of this part.
- To limit the analyzing region around the pivot pendulum in order to reduce the pixels number to analyze.
- Finding in the center of the black marker in the limited region
- By knowing the position of the marker, we can calculate the angular position of the first limb.

The general operating processing chain is shown in Figure 68.

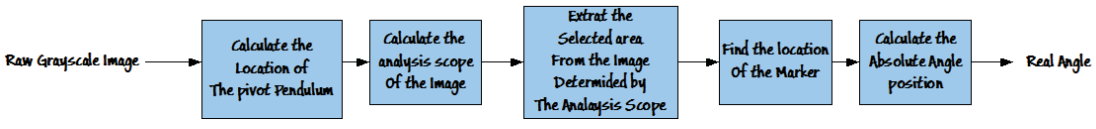


Figure 68: general operating processing chain to get the angle position with the marker detection

3.2.1 Picture format

It is recommended to use a camera that can return a picture in greyscale format. It spares us some processing time to convert data (converting rgb to greyscale takes time).

The camera feature can be configurable by using the LabVIEW block « vision and acquisition». In the library « Vision Express ».

3.2.2 Marker Detection

To detect the center of the marker placed on the second limb we need to analyze a limited region around the pivot pendulum in order to reduce the number of pixels for analyzing. There are a lot of methods for pattern recognition, to detect circle for example, but using a pattern recognition function proposed by LabVIEW take a lot of time to process (+70ms just by analyzing a region 100x100).

Another method, simple and easily to implement, is to analyze pixel by pixel from the limited region. (Figure 69). When we analyze a pixel and if the grey level is enough to belong to the marker, the position of the pixel is saved. When the analyzing of the entire pixel in the region is done, we calculate the center of the marker position by using a simple average operation (Figure 70). The advantage of this method is the simple implementation and a parallelizable structure, but the marker should be easy to identify from the rest element of the picture.

Result:

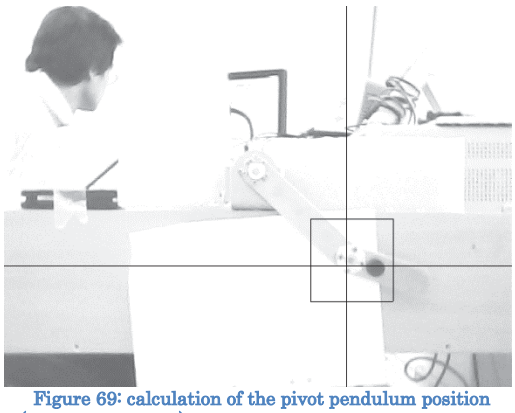


Figure 69: calculation of the pivot pendulum position (showed by a cross). The limited region around the pivot pendulum is represented by the square. The picture is obtained by a low cost usb camera for webcam application.

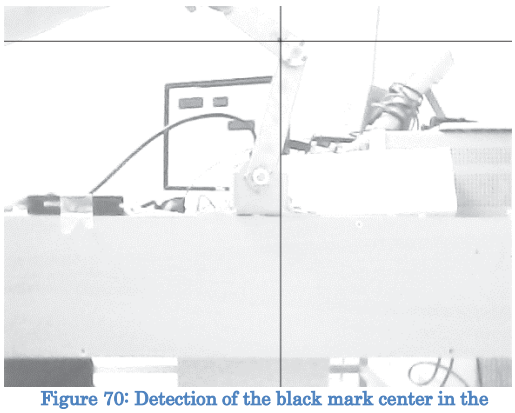


Figure 70: Detection of the black mark center in the analyzing region. The picture is obtained by a low cost usb camera for webcam application.

Initialization phase

One of the inconvenient of this method is the initializing phase that can be very fastidious (see Figure 71). The user needs to register:

- The center position of the shaft motor
- The center position of the pivot pendulum
- The center position of the black marker (during rest) needed to calculate the initial inclination of the camera
- The limited of the analyzing region

It is necessary to get a picture and stop the experimentation in order to know the position in the picture for each point.

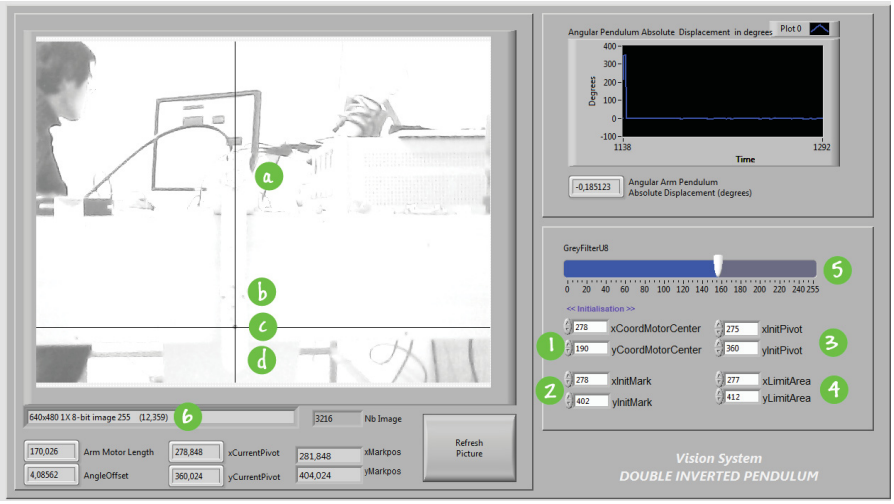


Figure 71: Vision System interface in LabVIEW

(1) initial center position of the shaft motor *(a)*
(2) initial center position of the marker *(c)*
(3) initial center position of the pivot Pendulum *(b)*
(4) limited Area *(d)*
(5) Filter Greyscale value
(6) position of the mouse cursor

Feature of camera

The camera we used (Figure 72) for testing our program was a low cost camera « camera usb 38 4754 ». This device has a lot of default in particularity by adding noise in the picture. The frame rate value is pretty low (30 fps), and there is no pretreatment available integrated and etc.

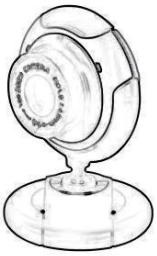


Figure 72: camera usb 38 4754, taken from the company web site

Specifications

Picture sensor	CMOS sensor
Resolution	1.3 megapixels (up to 3,200–2,400)
Video mode	RGB24/YUY2
Interface	USB 2.0
Video capture	640 x 480 at 30 frames per second 1600 x 1200 at 10–15 frames per second
SNR	Better than 48 dB
Dynamic range	Better than 72 dB
Focus	From 3 cm to infinity
White balance	Automatic
Lens	High-quality lens
Microphone	Built-in
Bracket	Folding base bracket
Still image file format	.bmp and .jpg

Processing times:

The processing time to get the angular position of the second limb is an important factor (see Figure 73).

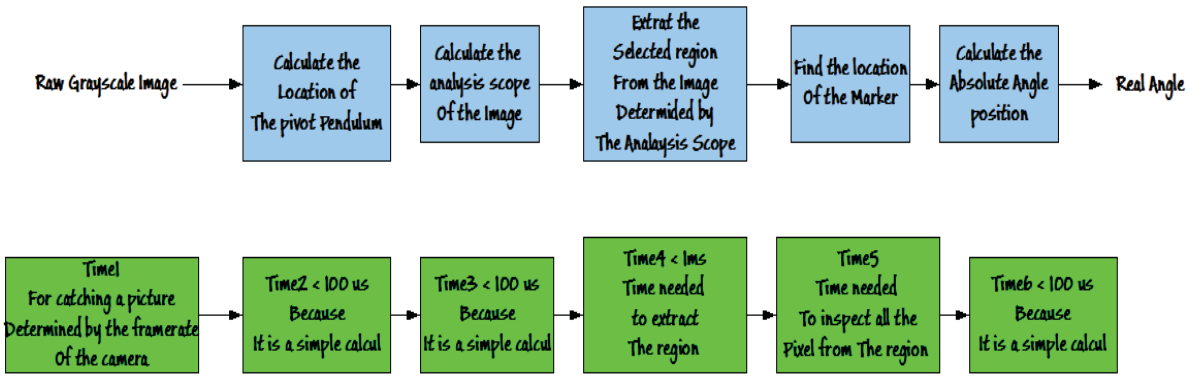


Figure 73 : processing times for the vision system with marker detection

As it shown in Figure 73, there are only some blocks which have a variable processing time in function of the equipment and the picture resolution: Time1, Time6.

Table 9: processing time

Time1	Associated with the frame rate value of the camera: time needed to capture a picture, greyscale U8 in preference.
Time6	Associated with the processing time to analyze the pixel region. The time value is essentially associated with the size of the region. By timing on Ni MYRIO, we find, on average, that for 1ms 666pixels are analyzed. Approximately, the time needed is: Time6 = nbpixels*1,5us

The equation of the processing time value for the execution of the task "image processing" is $T \cong T1 + T6 = \frac{1}{FrameRate} + nbpixels * 1{,}5us$

T2, T3, T4, T5 are neglected in comparing the T1 and T6 value. Nb pixels represent the number of the pixels in the analyzing region.

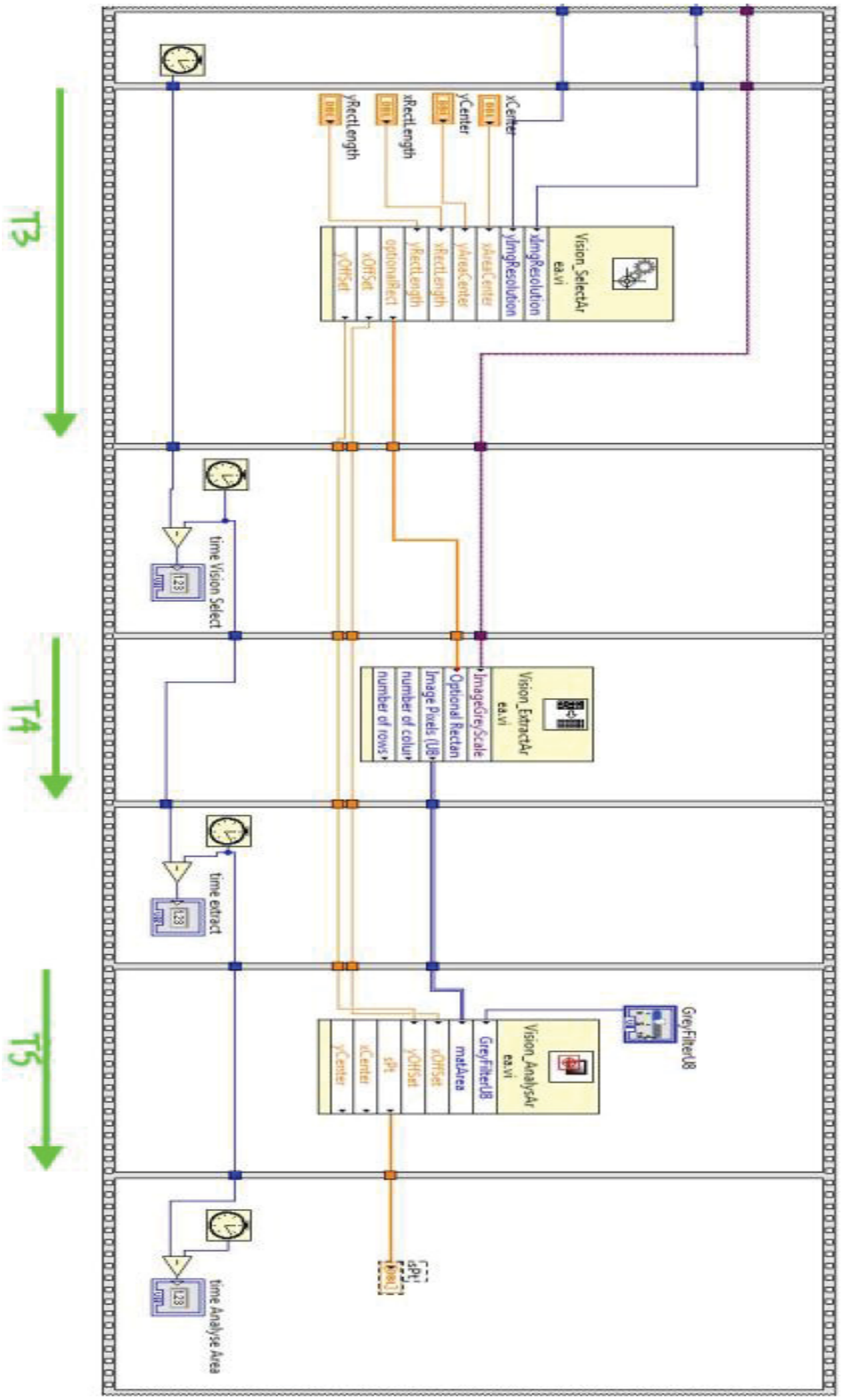
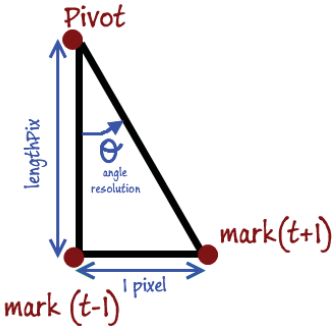


Figure 74: A way to measure the time needed for each part

Accuracy:

The accuracy of this method depends of the picture resolution, especially the length value (in pixel) between the pivot Pendulum and the black marker center.

In fact, if we want a good accuracy with $\boldsymbol{\theta}$ degree, that means we can detect a displacement by 1 pixel:



The minimum size of the analysis region is :
$Size > (2 * \text{LengthPix}) * (2 * \text{LengthPix}) = 4 * \text{LengthPix}^2$

We can calculate the time needed for the image treatment:

$$T6(\theta) = Size * 1{,}5us = 4 * \left(\frac{1}{\tan(\theta)}\right)^2 * 1{,}5us$$

$$\text{LengthPix} > \frac{1}{\tan(\theta)}$$

Table 10 : time needed for the image treatment depending to the size of the analysis region

Resolution $\boldsymbol{\theta}$ (degree)	***Size region***	***T6***
0,1	1146x1146	1,97 s
0,25	458x458	315 ms
0,5	229x229	78,8 ms
1	115x115	19,7 ms

We can see, Table 10, for a resolution by 0,5 degree, we will need at least 78,8ms for the processing time. This time value is too long, but can be reduce if we optimized the analyzing region around the pivot pendulum.

3.3 Third method: Using pattern recognition

3.3.1 Introduction

For finding the degree between two crossed lines we need four points. One of four points is common, then we can draw two crossed lines with three points. In our case one of them is always perpendicular to the common point, then just two points are remain. We used this concept for finding the angle of the second limb. In Figure 75 there are two points, the big point which is located on the joint between two limbs and the small point which is located on the second limb. Based on the above concept the big point is the common point. One perpendicular line is connected to the common point and it indicates the angle zero. We select clockwise for the direction of the angle. An example is shown in Figure 75, where by using two points we draw two lines and then the angle which these two lines make with each other.

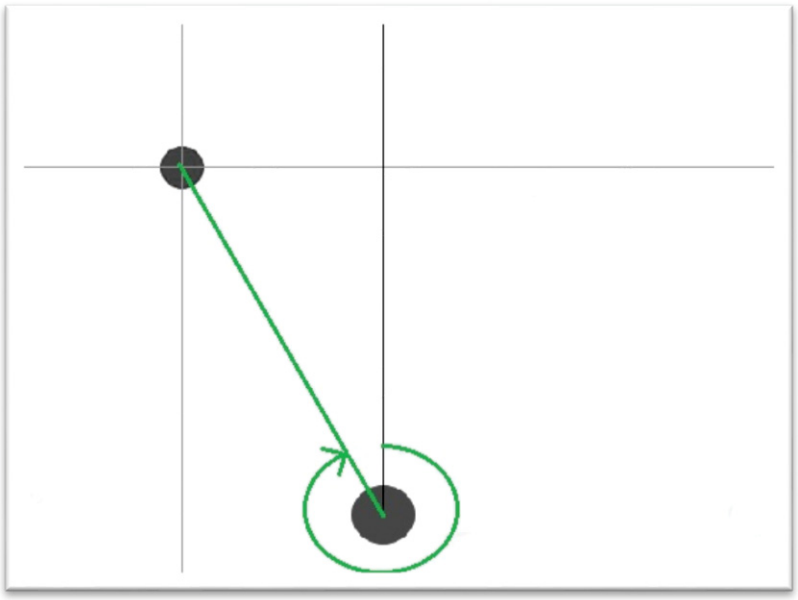


Figure 75: The strategy for finding the angle of the second limb

3.3.2 MATLAB Function

For applying the mentioned concept in MATLAB first we need to identify the center of circles and the function *imfindcircles* do this for us. This function has this option for finding the circles in a definable diameter range. This option helps us to finding our circles and avoids to facing a lot of other unwanted circles. For more information about the function please refer to MATLAB helps center. A white background helps the function to finding the right circles then in physical experiment by using white papers we make the background of the pendulum white.

The rest is defining which circle is the big one and which one is the small one and according to them drawing the lines.

Function *atan2* in MATLAB finds the angle between two lines. The MATLAB codes for the function of the third method are shown in below:

```
function [angledegree] = anglecalculator(img)
```

```
[centers, radii] = imfindcircles(img,[15 30],'ObjectPolarity','dark', ...
    'Sensitivity',0.95)
sizeradii = size(radii);
if sizeradii(:,1) >= 3 ;
    disp('more than 2 points are detected,the results will not be correct')
end
if radii(1,1) >= 20 ;
    bigpoint = centers(1,:);
else
    bigpoint = centers(2,:);
end
if radii(1,1) <= 20 ;
    smallpoint = centers(1,:);
else
    smallpoint = centers(2,:);
end
mypoint = [1,bigpoint(1,2)];
v1 = smallpoint - bigpoint;
v2 = mypoint - bigpoint;
Angle = -atan2(v1(1)*v2(2)-v1(2)*v2(1), v1*v2'), 2*pi) * 180/pi +90
angledegree = Angle ;
```

Features of this strategy are:

- As a gray image is enough for this strategy, this reduces the size of the images and the process considerable,
- The strategy has a simple functions which increases the speed,
- By changing some settings like brightness, contrast and sharpness inside of the camera then we do not need any filter and also this helps us for a more nimble strategy.

The Simulink file of the strategy is shown in Figure 76.

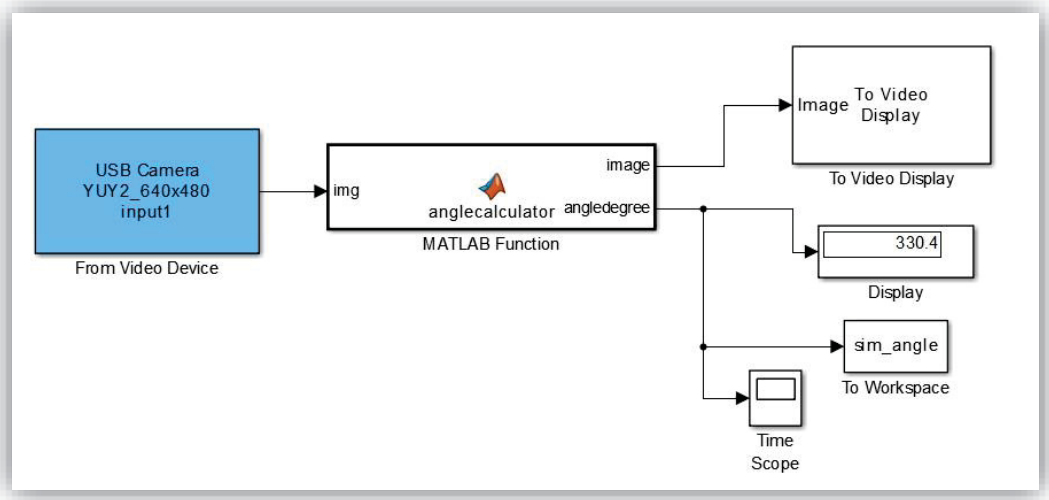


Figure 76: The Simulink figure of the vision system

Figure 76 is contain two main blocks. The input block “From Video Device” is a block from MATLAB which is for taking the pictures from the USB cameras. In the output of this Block we have 30 gray images with 640*480 resolution each second. The second important block is the MATLAB function contain function *anglecalculator* which represented before. The rests are the outputs, a text contain the angle or an image like Figure 77.

3.3.3 Running an experiment with the vision system

For testing the vision system an experiment on physical pendulum was run. Figure 77 is shown the experiment. In the experiment the pendulum limb has 161° by assuming the 0° is in the top position. The green and red lines added for better understating of how the angle calculated.

In Figure 77 the body of the pendulum cannot be seen easily and the reasons are two. First we put a white background for the pendulum as we mentioned before and second we changed the input camera settings to high brightness and contrast.

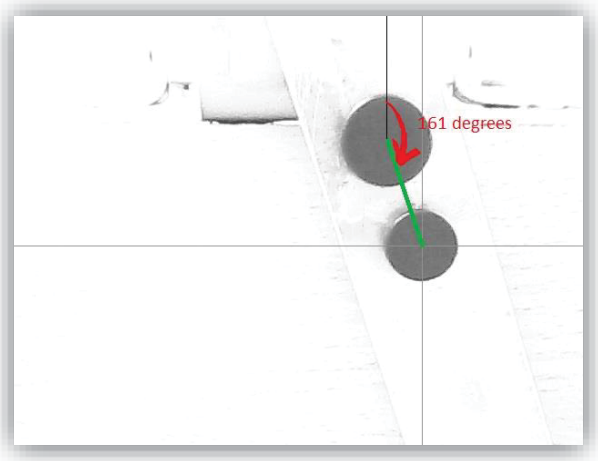


Figure 77: The output of MATLAB function in Figure 76 which shows the detected circles, drawn lines and the angle between them.

3.3.4 Verifying the vision system

For verifying the vision system and in absence of any reference for the second limb we decided to put the both circle marks on the first limb which is connected to the encoder and getting the angle from the encoder and the vision system simultaneously. In this way we can compare the result of the encoder and the vision system with each other.

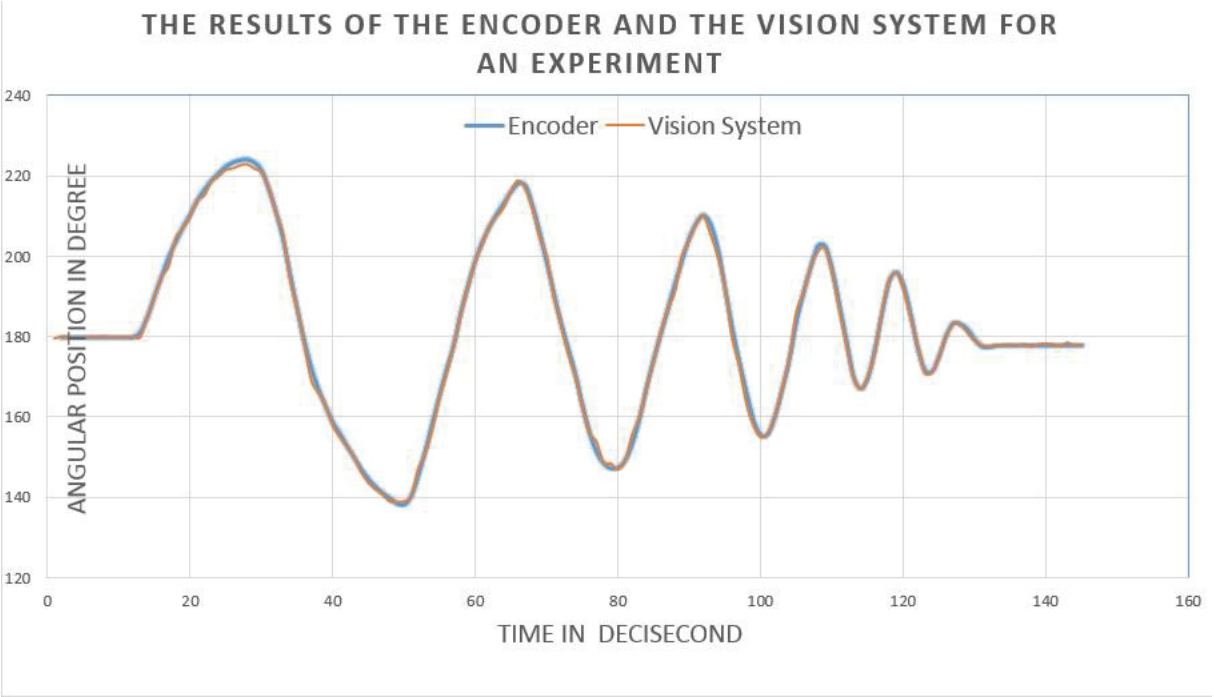


Figure 78: The results of the encoder and the vision system for an experiment

We run an experiment on physical pendulum by swinging the pendulum several times. The results of the experiment are recorded simultaneously by the encoder and the vision system and are shown in Figure 78. The orange line, the result of the vision system could follow the blue line, the result of the encoder with less than one degree average error. The errors are depicted in Figure 79.

There are two interesting points about Figure 78.

- **First**, for the experiment we did not drop the pendulum from 45˚. The reason is the speed of the pendulum for swinging is high for a 30fps camera and the images becomes blur and then the circle detector cannot detects the circles. There is a direct relationship between the speed of the pendulum displacement and the camera frame rate. For higher speed, a higher frame rate is needed.
- **Second**, the sampling interval for the encoder is 0.1 second but for the vision system is 0.033 which comes from 30 frames per second. We used a down sampling filter to fit the sampling intervals together.

Figure 79 shows the difference between the results of the encoder and the vision system. The red dotted line is the average error. The average error is around 0.37° and by decreasing the speed, the error decreased too.

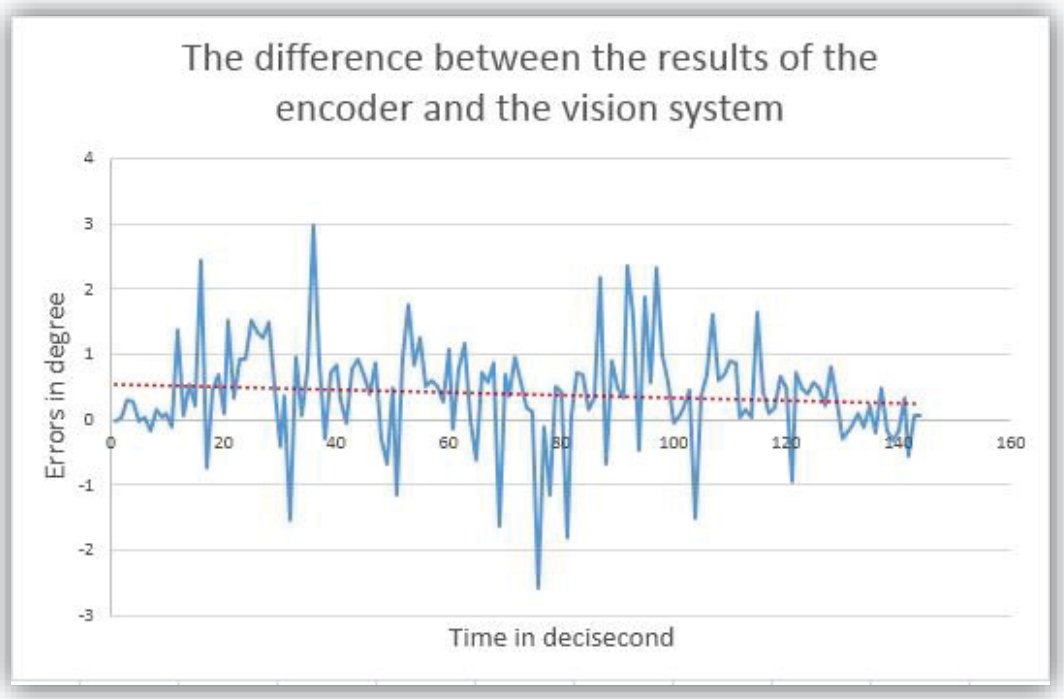


Figure 79: the difference between the results of the encoder and the vision system

What can be done for reducing the error? A list of some affecting factors are listed in below.

- Using a camera with higher shutter speed can produce lesser blur image and consequently lesser error,
- Using a camera with higher frame rate is necessary for high speed movements, higher frame rate causes lesser error too, in appendix the suggested frame rate is discussed,
- The camera should fix and align with the pendulum horizontally because one of the three necessary points for drawing the lines and calculating the angle is a virtual point perpendicular to the common point,
- The camera should connect tightly to the table and the table should be enough big and heavy for preventing the pendulum from vibration,
- The circle marks on the pendulum should align together in the center of pendulum, any disorganize can cause error for calculating the angle,
- The circle marks should have low width as much as possible because width of the marks can cause shadow on the pendulum and the shadow can cause problem for finding center position of circles and this brings error in angle calculation,
- The circle marks should not have any light reflection because light reflection from the marks or any other objects in the picture has a grave consequences on the whole picture,
- The pendulum should be clean, especially around the circle marks because can cause a shadow effect.

Conclusion

First method, Hough Transform is quite accurate, but it takes a lot of time for LabVIEW to perform it. Also it is not affordable to use Matlab .m file in LabVIEW environment, then it is inevitable to using more nimble methods.

The second and the third method are fast enough and on the other hand are enough accurate for our purpose. In overall pattern recognition is suitable for our case.

According to appendix 1 which is a study about the specification of the suitable camera for the vision system at least a camera with 72 fps is needed.

4 Modelling and Control

In this chapter first we derive the equations of motion, then by using the equations we derive the nonlinear model, afterward we verify the nonlinear model and at the end we will linearize the nonlinear model which is necessary for deriving the control law.

In This section research study [7] helps us for modelling our double pendulum. The writers of [7] cover the whole process of modelling, control, simulation and data analysis for a physical double pendulum. The reader is referred to the work [5] and [6] for useful materials about the modelling of a physical double pendulum.

We have two possibilities for deriving the equation of motions:

- **Classical mechanics**, Newton's laws of motion which are three physical laws that together laid the foundation for classical mechanics. They describe the relationship between a body and the forces acting upon it, and its motion in response to said forces,
- **A re-formulation of classical mechanic**, Lagrangian mechanics, Euler–Lagrange equation by using Hamilton's principle of stationary action. Lagrangian mechanics applies to systems whether or not they conserve energy or momentum, and it provides conditions under which energy, momentum or both are conserved.

Any mechanical system with N particles in $\mathbb{R}^3$, which is subject to R constraints, features in general $f = 3N - R$ degrees of freedom represented by generalized coordinates $q_1, q_2, \dots, q_f$ compatible with all constraints. These general coordinates may assume any value independently of each other, and the position of all N particles is uniquely determined by the f generalized coordinates.

For a given physical system the choice of generalized coordinates is not unique, and often there exists another set of generalized coordinates $Q_i (i = 1, \dots, f)$ which is equally well suited for description of the system. It is a matter of personal taste and experience which set to choose.

The number of equations for Euler–Lagrange equation is *3N-R* but for Newton's laws of motion is *3N*. Also Lagrangian has the advantage that it takes the same form in any system of generalized coordinates, and it is better suited to generalizations [8]. Due to mentioned benefits for using Euler–Lagrange rather than Newtonian mechanic we use Euler–Lagrange for deriving the equation of motions.

Before starting to derive the Euler–Lagrange equations, we work on two prerequisites, the moment of inertia and the gravity of earth.

4.1 Moment of Inertia

A simple pendulum or mathematical pendulum is consist of a mass suspended from a massless string or rod on a frictionless pivot. Any effect from the string or rod itself can be neglected. In contrast for modeling a physical pendulum by a rod that is not massless and may be in an arbitrarily-shape, we should calculate the moment of inertia precise.

In this chapter we neglect the length of a small part of the pendulum above the shaft, the area which is shown in Figure 80. This neglecting is contain the length of the pendulum,

not the weight. For bringing this part in our calculations we should give it a role of another pendulum.

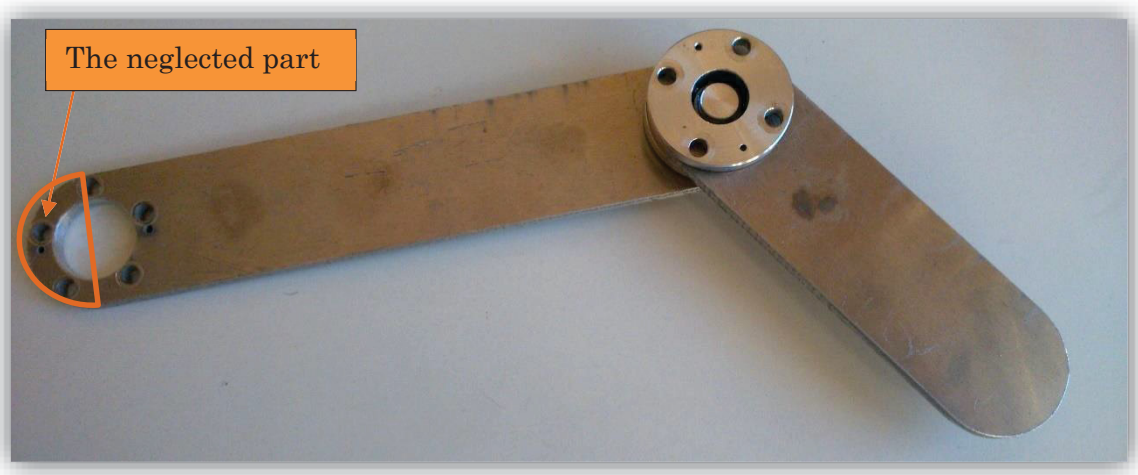


Figure 80: The neglected part of the pendulum for modeling

In this project we calculate the moment of inertia for a physical pendulum, first by considering the pendulum in rod shape and then in flat plate shape. At the end of this capture we compare the results for two different shapes.

For illustration, let us consider a uniform rigid rod, pivoted from a frame as it shown in Figure 81. Clearly, the center of mass is at a distance *L/2* from the point of suspension.

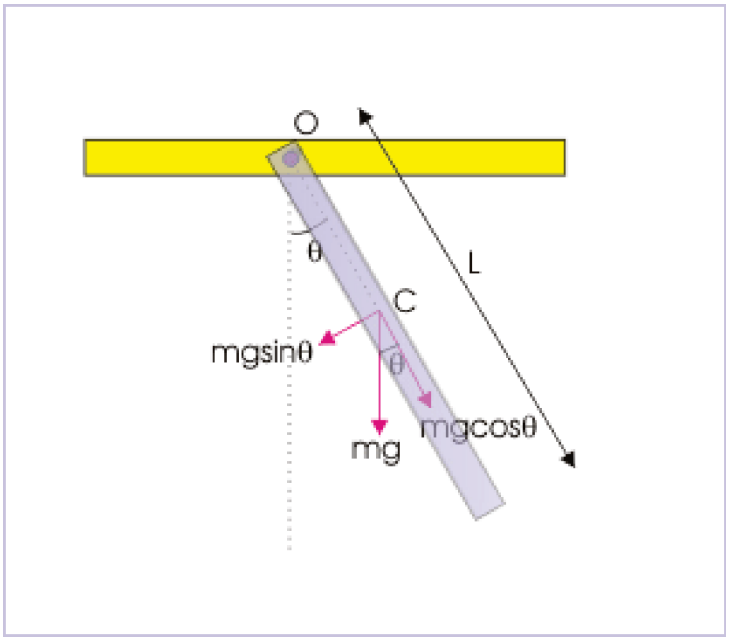


Figure 81: A rigid rod with uniform mass distribution hangs from a pivot point

In our physical pendulum the center of mass is not in the center of rod and then we cannot consider it *h=L/2*. Due to the shape and the mass of the joints the center of mass will be in another point. Then by considering the mass of the joint we should change the center of

masses for our pendulum. Moreover, we need to evaluate the moment of inertia about the pivot point, not the center of mass, so we should apply the parallel axis theorem too as it shown in Figure 82.

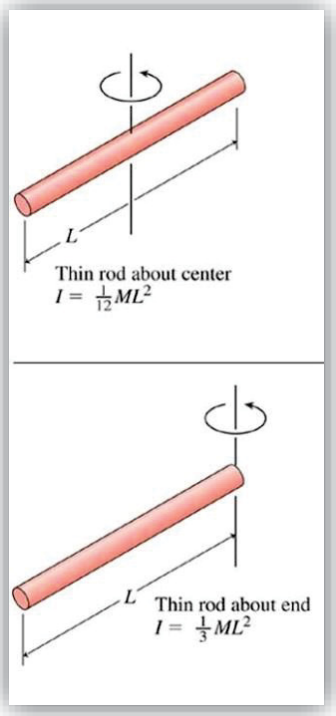


Figure 82: Moment of inertia in a physical pendulum

The values for calculating the moment of inertia including the center of masses are shown in below:

```
m1 = 0.105 ; %kg  %mass of rod 1
l1 = 0.15 ; %m    % length of rod 1
l1c = 0.085 ; %m  % distance to the center of mass of rod 1
j1 = ((m1*(l1^2))/12)+(m1*(l1c^2)) %kg.m^2 %moment of inertia of rod 1

m2 = 0.049 ; %kg  %mass of rod 2
l2 =0.1; %m       % length of rod 2
l2c = 0.035 ; %m      % distance to the center of mass of rod 2
j2 = ((m2*(l2^2))/12)+(m2*(l2c^2)) %kg.m^2 %moment of inertia of rod 2
```

We assumed our pendulum is a rigid rod but our physical pendulum is more look like a flat plate rather than a rod, then we rewrite the moment of inertia for a flat plate that is more look like Figure 83.

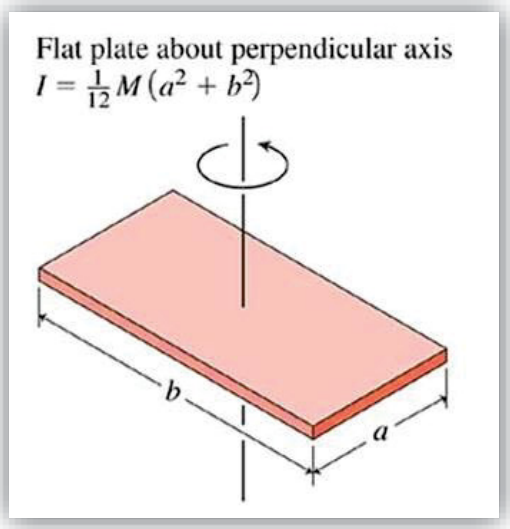


Figure 83: Moment of inertia in a flat plate

This time the values for calculating the moment of inertia including the center of masses for a flat plate shape are shown in below.

```
m1 = 0.105 ; %kg  %mass of part 1
b1 = 0.15 ; %m %length of part 1
a1 = 0.015; %m %width of part 1
b1c = 0.085 ; %m %length of part 1, center of mass
a1c = 0.0075; %m %width of part 1, center of mass
J1 =((m1*(a1^2+b1^2))/12)+(m1*(a1c^2+b1c^2)) %kg.m^2 %moment of inertia of part1

m2 = 0.049 ; %kg  %mass of part 2
b2 = 0.1; %m %length of part 2
a2 = 0.015; %m %width of part 2
b2c = 0.035 ; %m %length of part 2, center of mass
a2c = 0.0075; %m %width of part 2, center of mass
J2 = ((m2*((a2^2)+(b2^2)))/12)+(m2*((a2c^2)+(b2c^2)))%kg.m^2 %moment of inert of
part 2
```

In Table 11 a comparison for the moment of inertia by considering the shape of the pendulum as rod or flat plate is shown.

Table 11: comparison of moment of inertia if we consider the pendulums as rod or flat plate shape.

	Moment of inertia for Rod shape	Moment of inertia for Flat plate shape
J1_limb1	9.555000000000001e-04	9.633750000000002e-04
J2_libm2	1.008583333333334e-04	1.045333333333333e-04

A precise value for the moment of inertia helps us for verifying the model and subsequently having more precise control signal.

4.2 Gravity of Earth

For having a precise model based on the physical system we should considering all of the details. One of the parameters in our model is the gravity of earth that we always consider it as 9.81 but in reality it is a little bit different based on the distance from the surface of the earth and also location.

The gravity of earth, which is denoted by *g*, refers to the acceleration that the earth imparts to objects on or near its surface. In *SI* units this acceleration is measured in meters per second squared (in symbols, *m/s2*) or equivalently in newton per kilogram (*N/kg* or *N·kg−1*). It has an approximate value of 9.81 *m/s2*, which means that, ignoring the effects of air resistance, the speed of an object falling freely near the earth's surface will increase by about 9.81 meters (32.2 *ft.*) per second.

The equatorial bulge and the effects of the earth's inertia mean that sea-level gravitational acceleration increases from about 9.780 *m/s2* at the equator to about 9.832 *m/s2* at the poles, so an object will weigh about 0.5% more at the poles than at the equator. Anywhere on earth away from the equator or poles, effective gravity points not exactly toward the center of the earth, but rather perpendicular to the surface of the geoid, which, due to the flattened shape of the earth, is somewhat toward the opposite pole. About half of the deflection is due to inertia, and half because the extra mass around the equator causes a change in the direction of the true gravitational force relative to what it would be on a spherical earth. [10]

Briefly, the value of *g* varies inversely with the distance from the center of the earth. In fact, the variation in g with distance follows an inverse square law where *g* is inversely proportional to the distance from earth's center. This inverse square relationship means that as the distance is doubled, the value of *g* decreases by a factor of four. As the distance is tripled, the value of *g* decreases by a factor of nine and so on. This inverse square relationship is depicted in the Figure 84. [11]

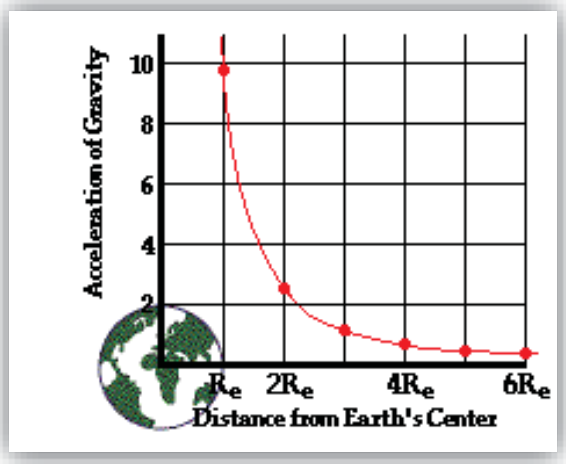
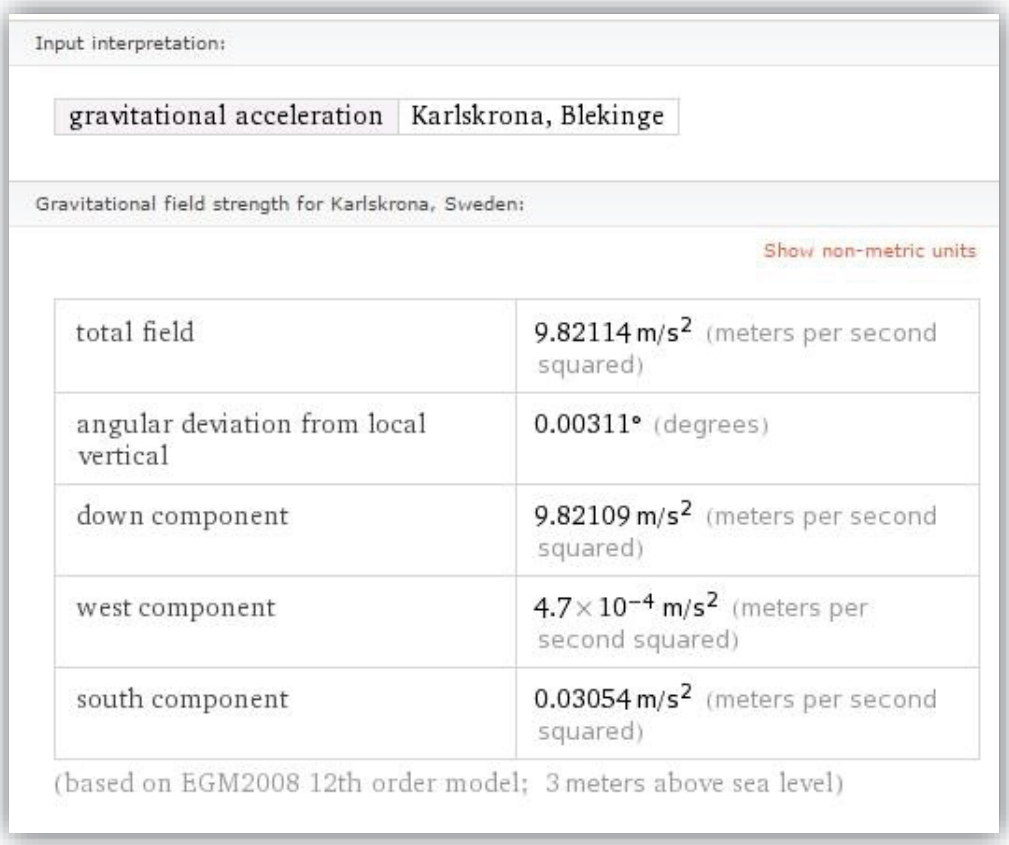


Figure 84: value of g if we become far away from the Earth's center [11]

On the website http://www.physicsclassroom.com/class/circles/Lesson-3/The-Value-of-g by entering the name of the city the gravity of earth on that city will appear. Here are the

results for Karlskrona (Figure 85 and Figure 86), where the lab is situated in coordinates 56.1608° N, 15.6500° E.

Figure 85: Karlskrona city in Sweden, the location of the laboratory [11]



Input interpretation:

gravitational acceleration | Karlskrona, Blekinge

Gravitational field strength for Karlskrona, Sweden:

Show non-metric units

total field	9.82114 m/s^2 (meters per second squared)
angular deviation from local vertical	0.00311° (degrees)
down component	9.82109 m/s^2 (meters per second squared)
west component	4.7×10^{-4} m/s^2 (meters per second squared)
south component	0.03054 m/s^2 (meters per second squared)

(based on EGM2008 12th order model; 3 meters above sea level)

Figure 86: Results from the gravity calculator based on the Karlskrona city [11]

Then we consider the gravity of earth for our project **g = 9.82114** instead of g=9.81.

In the next section we derive the equations of motion. Our strategy for deriving the equations is, first we derive the equations for a double pendulum without considering the DC motor and second we will add the motor`s role in our equations.

4.3 Double Pendulum without the motor

The core element of Lagrangian mechanics is the Lagrangian function, which summarizes the dynamics of the entire system in a very simple expression. The physics of analyzing a system is reduced to choosing the most convenient set of generalized coordinates, determining the kinetic and potential energies of the constituents of the system, then writing down the equation for the Lagrangian to use in Lagrange's equations. It is defined by Equation (1).

$$L = T - V \tag{1}$$

Where

- T is the total kinetic energy,
- V is the total potential energy.

Figure 87 illustrates the schematic of a double pendulum which contains two limbs, limb 1 (blue one) and limb 2 (red one). Limb 1 is attached to the motor`s shaft. Also there is a

rotational joint between limb 1 and limb 2. Moreover limb 2 may freely rotate around the joint.

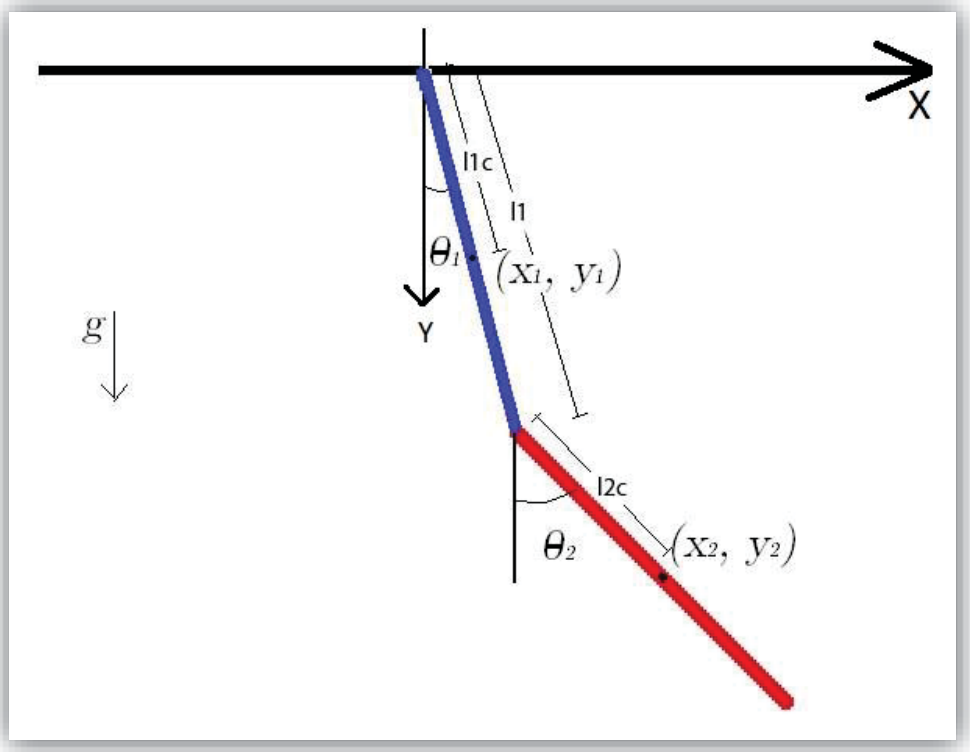


Figure 87: Sketch of double pendulum

We have 2 particle here, the center of mass of limb 1 with length l_{1c} and center of mass of limb 2 with length l_{2c}. The length of limb 1 is $l_1$and the length of limb 2 is l_2. The Degree Of Freedom, DOF for this system is 2. Two DOF for the double pendulum is shown in Figure 88.

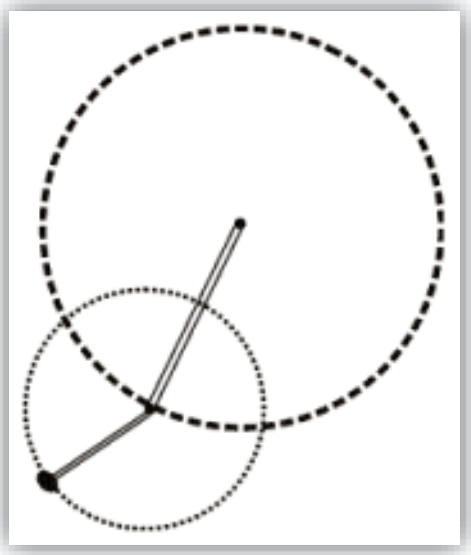


Figure 88: Degree of freedom in double pendulum

First we calculate the kinetic energy, T in Equation (1). T has two parts:

- Translational kinetic energies,
- Rotational kinetic energies.

Translational kinetic energies of the center of mass for the two limbs are given by Equation (2).

$$T_{1,translational} = \frac{m_1}{2}\left|\dot{\vec{r_1}}\right|^2$$
$$T_{2,translational} = \frac{m_2}{2}\left|\dot{\vec{r_2}}\right|^2 \quad (2)$$

Where

- $\vec{r_1}$, is the trajectory of particle1, the center of mass of limb1,
- $\vec{r_2}$, is the trajectory of particle2, the center of mass of limb2.

The trajectories are given in Equation (3) and (4).

$$\vec{r_1} = \begin{pmatrix} X_1 \\ Y_1 \end{pmatrix} = \begin{pmatrix} l_{1c}\sin\theta_1 \\ -l_{1c}\cos\theta_1 \end{pmatrix} \quad (3)$$

$$\vec{r_2} = \begin{pmatrix} X_2 \\ Y_2 \end{pmatrix} = \begin{pmatrix} l_1\sin\theta_1 + l_{2c}\sin\theta_2 \\ -l_1\cos\theta_1 - l_{2c}\cos\theta_2 \end{pmatrix} \quad (4)$$

According to the equation (2) we calculate $\left|\dot{\vec{r_1}}\right|^2$ and $\left|\dot{\vec{r_2}}\right|^2$ which are given in Equation (5) and (6).

$$\left|\dot{\vec{r_1}}\right|^2 = {l_{1c}}^2{\dot{\theta}_1}^2 \quad (5)$$

$$\left|\dot{\vec{r_2}}\right|^2 = {l_1}^2{\dot{\theta}_1}^2 + {l_{2c}}^2{\dot{\theta}_2}^2 + 2l_1l_{2c}\,\dot{\theta}_1\dot{\theta}_2\cos(\theta_1 - \theta_2) \quad (6)$$

By using Equations (5) and (6) we can derive the translational kinetic energies which given in Equation (7).

$$T_{1,translational} = \frac{m_1}{2}{l_{1c}}^2{\dot{\theta}_1}^2$$
$$T_{2,translational} = \frac{m_2}{2}{l_1}^2{\dot{\theta}_1}^2 + \frac{m_2}{2}{l_{2c}}^2{\dot{\theta}_2}^2 + m_2\,l_1l_{2c}\,\dot{\theta}_1\dot{\theta}_2\cos(\theta_1 - \theta_2) \quad (7)$$

Rotational kinetic energies of the limbs around their respective centers of mass are given by Equations (8) and (9).

$$T_{1,rotational} = \frac{J_1}{2}{\dot{\theta}_1}^2 \quad (8)$$

$$T_{2,rotational} = \frac{J_2}{2}\dot{\theta}_2{}^2 \tag{9}$$

Where

- J_1 is the moments of inertia for the limb 1,
- J_2 is the moments of inertia for the limb 2.

Hence the total kinetic energy of the system is given in Equation (10).

$$\begin{aligned} T &= T_{1,translational} + T_{2,translational} + T_{1,rotational} + T_{2,rotational} \\ &= \frac{m_1}{2} l_{1c}{}^2 \dot{\theta}_1{}^2 + \frac{m_2}{2} l_1{}^2 \dot{\theta}_1{}^2 + \frac{m_2}{2} l_{2c}{}^2 \dot{\theta}_2{}^2 + m_2\, l_1 l_{2c}\, \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2) \\ &+ \frac{J_1}{2}\dot{\theta}_1{}^2 + \frac{J_2}{2}\dot{\theta}_2{}^2 \end{aligned} \tag{10}$$

The gravitational potential energies of the two limbs are given in Equations (11) and (12).

$$V_1 = m_1 g Y_1 = -m_1 g l_{1c} \cos\theta_1 \tag{11}$$

$$V_2 = m_2 g Y_2 = -m_2 g (l_1 \cos\theta_1 + l_{2c} \cos\theta_2) \tag{12}$$

Where

- g is the gravity of earth,
- y_1 is y elemnt of $\overrightarrow{r1}$,
- y_2 is y elements of $\overrightarrow{r2}$.

In Figure 87, y is in the direction of g. Hence the total potential energy of the system is given in Equation (13).

$$V = V_1 + V_2 = -m_1 g l_{1c} \cos\theta_1 - m_2 g l_1 \cos\theta_1 - m_2 g l_{2c} \cos\theta_2 \tag{13}$$

We return again to Equation (1) and by putting Equations (10) and (13) we can derive the Lagrangian which is given in Equation (14).

$$L = T - V = c_1 \dot{\theta}_1{}^2 + c_2 \dot{\theta}_2{}^2 + c_3 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2) + c_4 \cos\theta_1 + c_5 \cos\theta_2 \tag{14}$$

Where

- $c_1 = \frac{m_1}{2} l_{1c}{}^2 + \frac{m_2}{2} l_1{}^2 + \frac{J_1}{2}$,
- $c_2 = \frac{m_2}{2} l_{2c}{}^2 + \frac{J_2}{2}$,
- $c_3 = m_2\, l_1 l_{2c}$,
- $c_4 = (m_1 l_{1c} + m_2 l_{2c}) g$,
- $c_5 = m_2 g l_{2c}$.

The evolution of the system is determined by the Euler-Lagrange equation.

$$\frac{\mathrm{d}}{\mathrm{dt}} \left(\frac{\partial L}{\partial \dot{\theta}_i}\right) - \frac{\partial L}{\partial \theta_i} = 0 \tag{15}$$

Where

- i, is 2 for our double pendulum.

For the double pendulum, Equation (15) gives us two coupled second order ordinary differential equations which are given in Equations (16) and (17).

$$c_4 sin\theta_1 + 2c_1\ddot{\theta_1} + c_3\ddot{\theta_2}\cos(\theta_1 - \theta_2) + c_3\dot{\theta_2}^2\sin(\theta_1 - \theta_2) = 0 \tag{16}$$

$$c_5 sin\theta_2 + 2c_2\ddot{\theta_2} + c_3\ddot{\theta_1}\cos(\theta_1 - \theta_2) - c_3\dot{\theta_1}^2\sin(\theta_1 - \theta_2) = 0 \tag{17}$$

Equations (16) and (17) are the equations of motion for the double pendulum. Due to sin and cos in our equation our system is nonlinear. Now by using the equations (16) and (17) we can model the double pendulum. Here we neglect the friction but in the next chapter we will add friction and off course the motor role.

4.4 Double pendulum with DC motor

Now we will add the motor`s role and also friction in our Euler-Lagrange equations. If we have a non-conservative force, the equation (15) will change to equation (18).

$$\frac{\mathrm{d}}{\mathrm{dt}}\left(\frac{\partial L}{\partial \dot{\theta}_i}\right) - \frac{\partial L}{\partial \theta_i} = Q_j \tag{18}$$

Where

- Q_j, are the generalized forces and it is include all of non-conservative forces which can not be written as potential energy e.g. driving force and friction force.

In some books the equation (18) is showed in the shape of equation (19).

$$\frac{\mathrm{d}}{\mathrm{dt}}\left(\frac{\partial L}{\partial \dot{\theta}_i}\right) - \frac{\partial L}{\partial \theta_i} = -\frac{\partial F}{\partial \dot{\theta}_i} \tag{19}$$

We named the generalized coordinated in equation (19) as θ to better understating in our case but the common symbol is q. In equation (19) F is a positive definite quadratic form in the generalized velocities.

If the generalized forces of friction are

$$Q_i = -b_i\dot{\theta}_i$$

Then F is given by

$$F = \frac{1}{2}\sum_i b_i\dot{\theta}_i$$

Such that

$$Q_i = -\frac{\partial F}{\partial \dot{\theta}_i}$$

We can come to this result that equation (18) and (19) are equal.

Our motor is a PMDC, Permanent Magnet DC motor. The dc motors can be divided in four types:

- shunt wound dc motor,
- series wound dc motor,
- compound wound dc motor,
- permanent magnet dc motor.

According to [12], This type of the DC motors has some interesting characteristic such:

- Higher efficiency since no electrical energy is used or losses incurred for developing or maintaining the motor's magnetic field,
- Higher torque and power density,
- Linear torque speed characteristics that are more predictable,
- Better dynamic performance due to higher magnetic flux density in air gap,
- Simplified construction and essentially maintenance-free,
- More compact size.

In Figure 89 the model of a general PM DC Motor is presented.

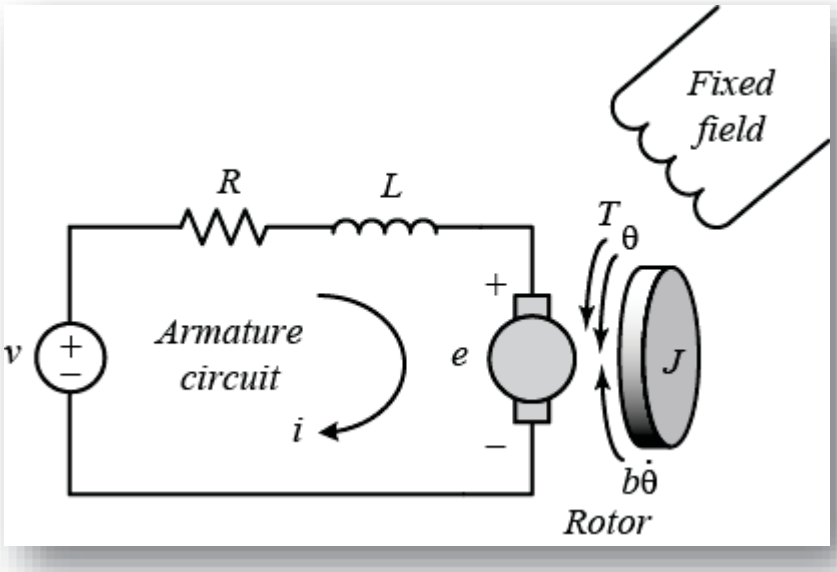


Figure 89: Degree of freedom in double pendulum [13]

In Figure 89, v is the input voltage of the system and e is the back emf which is proportional to the angular velocity of the motor`s shaft by a constant factor K_e.

$$e = K_e\dot{\theta}$$

In *SI* units, back emf constants and the motor torque are equal, then $K = K_t = K_e = 0.0934\frac{Nm}{A}$ for our motor. [13]

From Figure 89, we can derive the following governing equations based on Kirchhoff's voltage law.

$$L_a \frac{d}{dt} i_a + R_a i_a + K \frac{d}{dt}\theta - V = 0 \tag{20}$$

Where

- L_a, is electric inductance which for our motor is 0.423 *10^-3 Henry from the company datasheet of the Motor,
- R_a, is electric resistance which for our motor is 0.608 Ohm from the company datasheet of the Motor.

Equation (20) is our equation for the motor and also our third equation for the double pendulum with the motor.

The first limb of the double pendulum is connected to the motor`s shaft then in Equation (16) which if the equation of the first limb we should add the torque of the motor. Also by considering the friction forces Equations (16) and (17) will change to Equations (21) and (22).

$$\frac{\mathrm{d}}{\mathrm{dt}}\left(\frac{\partial L}{\partial \dot{\theta_1}}\right) - \frac{\partial L}{\partial \theta_1} = \tau_m - b_1\dot{\theta}_1 - T_{c1} sign(\dot{\theta}_1) - b_2(\dot{\theta}_1 - \dot{\theta}_2) - T_{c2} sign(\dot{\theta}_1 - \dot{\theta}_2) \tag{21}$$

$$\frac{\mathrm{d}}{\mathrm{dt}}\left(\frac{\partial L}{\partial \dot{\theta_2}}\right) - \frac{\partial L}{\partial \theta_2} = -b_2(\dot{\theta}_2 - \dot{\theta}_1) - T_{c2} sign(\dot{\theta}_2 - \dot{\theta}_1) \tag{22}$$

Where

- $\tau_m = Ki_a$, is the torque produced by the motor,
- b_1 and b_2 are coefficients of Viscos friction,
- T_{c1} and T_{c2} are coefficients of Coulomb friction.

By putting Equations (16) and (17) in the left hand side of equation (21) and (22) respectively we will have our final equations in expansion from.

$$\begin{aligned}(m_1 l_{1c} + m_2 l_{2c}) g sin\theta_1 + (m_1 {l_{1c}}^2 + m_2 {l_1}^2 + J_1)\ddot{\theta_1} + m_2 l_1 l_{2c}\ddot{\theta_2} cos(\theta_1 - \theta_2) \\ + m_2 l_1 l_{2c}{\dot{\theta_2}}^2 sin(\theta_1 - \theta_2) \\ = Ki_a - b_1\dot{\theta}_1 - T_{c1} sign(\dot{\theta}_1) - b_2(\dot{\theta}_1 - \dot{\theta}_2) - T_{c2} sign(\dot{\theta}_1 - \dot{\theta}_2)\end{aligned} \tag{23}$$

$$\begin{aligned}m_2 g l_{2c} sin\theta_2 + (m_2 {l_{2c}}^2 + J_2)\ddot{\theta_2} + m_2 l_1 l_{2c}\ddot{\theta_1} cos(\theta_1 - \theta_2) \\ - m_2 l_1 l_{2c}{\dot{\theta_1}}^2 sin(\theta_1 - \theta_2) = -b_2(\dot{\theta}_2 - \dot{\theta}_1) - T_{c2} sign(\dot{\theta}_2 - \dot{\theta}_1)\end{aligned} \tag{24}$$

$$L_a \frac{d}{dt} i_a + R_a i_a + K \frac{d}{dt}\theta_1 - V = 0 \tag{25}$$

Equations (23), (24) and (25) are our equations of motion for the physical double pendulum that in in short from can be seen as Equations (26), (27) and (28).

$$c_4 sin\theta_1 + 2c_1\ddot{\theta_1} + c_3\ddot{\theta_2}\, cos(\theta_1 - \theta_2) + c_3\dot{\theta_2}^2 sin(\theta_1 - \theta_2) = Ki_a - b_1\dot{\theta}_1 - T_{c1} sign(\dot{\theta}_1) - b_2(\dot{\theta}_1 - \dot{\theta}_2) - T_{c2} sign(\dot{\theta}_1 - \dot{\theta}_2) \quad (26)$$

$$c_5 sin\theta_2 + 2c_2\ddot{\theta_2} + c_3\ddot{\theta_1}\, cos(\theta_1 - \theta_2) - c_3\dot{\theta_1}^2 sin(\theta_1 - \theta_2) = -b_2(\dot{\theta}_2 - \dot{\theta}_1) - T_{c2} sign(\dot{\theta}_2 - \dot{\theta}_1) \quad (27)$$

$$L_a\dot{i_a} + R_a i_a + K\dot{\theta}_1 - V = 0 \quad (28)$$

The only values that are unknown in Equations (23), (24) are 4 coefficients of frictions, b_1, b_2 , T_{c1} and T_{c2} . In the next section some experiments will design to estimate the coefficients of friction.

4.5 Friction

In our system there is a torque which acts in opposite direction of motor`s angular velocity and that is friction and it comes from the friction between the brushes and commutator as well as in the bearing in joint between limb 1 and limb 2.

Our work in this section is based on two studies [14] and [15]. Those studies have common purpose, that is, what model of friction for an inverted pendulum is the most proper to represent the real friction. Moreover the master thesis [15] for estimating and compensating of the friction in an inverted pendulum from Blekinge institute of technology, BTH helps us considerable.

Here we will cover two types of friction, Viscous and Coulomb. For studying the friction in this section we model a simple pendulum. The schematic of a simple pendulum is shown in Figure 90.

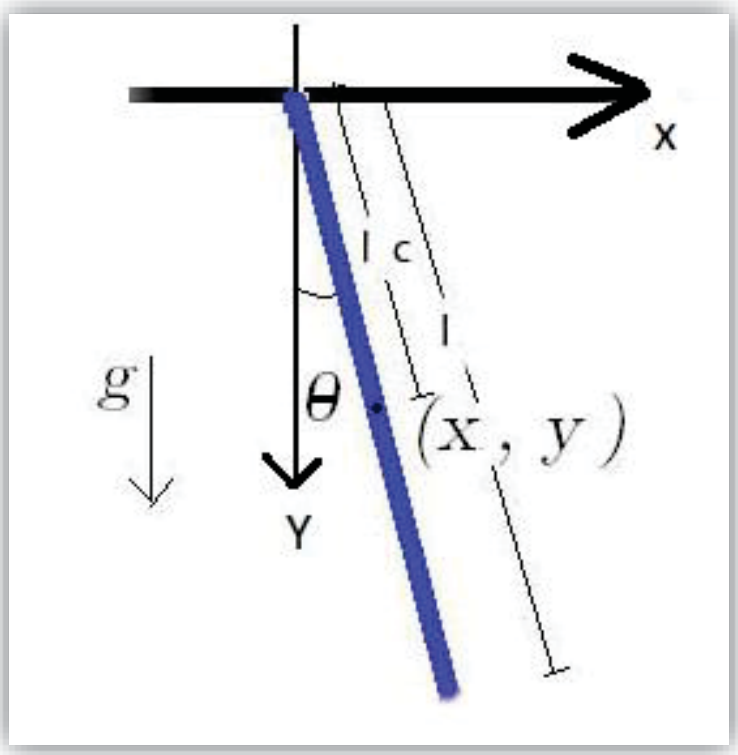


Figure 90: Sketch of a simple pendulum

The procedure for deriving the equation of motion for the simple pendulum is look like to the double pendulum, even much easier. Again based on equation (18), the general Euler-Lagrange equation, the equation of motion for the simple pendulum is equation (29).

$$J\ddot{\theta} + mgl_c sin\theta = -b\dot{\theta} - T_c sign(\dot{\theta}) \tag{29}$$

Where

- J is the moment of inertia for the simple pendulum,
- m is the mass of the simple pendulum,
- l_c is the length from the pivot to the center of mass,
- b is the coefficient of viscous friction,
- T_c is the coefficient of coulomb friction.

Two parameters are unknown inside of the Equation (29), b and T_c. For identifying these two parameters we will do two experiments with the physical double pendulum in two different ways:

- First, we fix the joint between the two limbs then the whole pendulum acts as a simple pendulum and we release it from 90° to reaching the coefficients of friction between limb 1 and the motor,
- Second, we fix the limb 1 and releasing the limb 2 again from 90° as a simple pendulum to reaching the coefficients of friction in the joint between limb 1 and limb 2.

By comparing the results from the experiments and also the simulations for two experiments we can estimate the coefficients of friction. Here we faced an unusual

behavior from the pendulum and that was, for dropping the pendulum from a constant degree, like 90° the pendulum reacts different. In Figure 91 the results for 5 similar experiments are shown. The small rectangles in the figure show the exactly degree of the first limb in the time of dropping. The numbers above the black arrows are the exactly degree of the first limb in the second turning point after dropping.

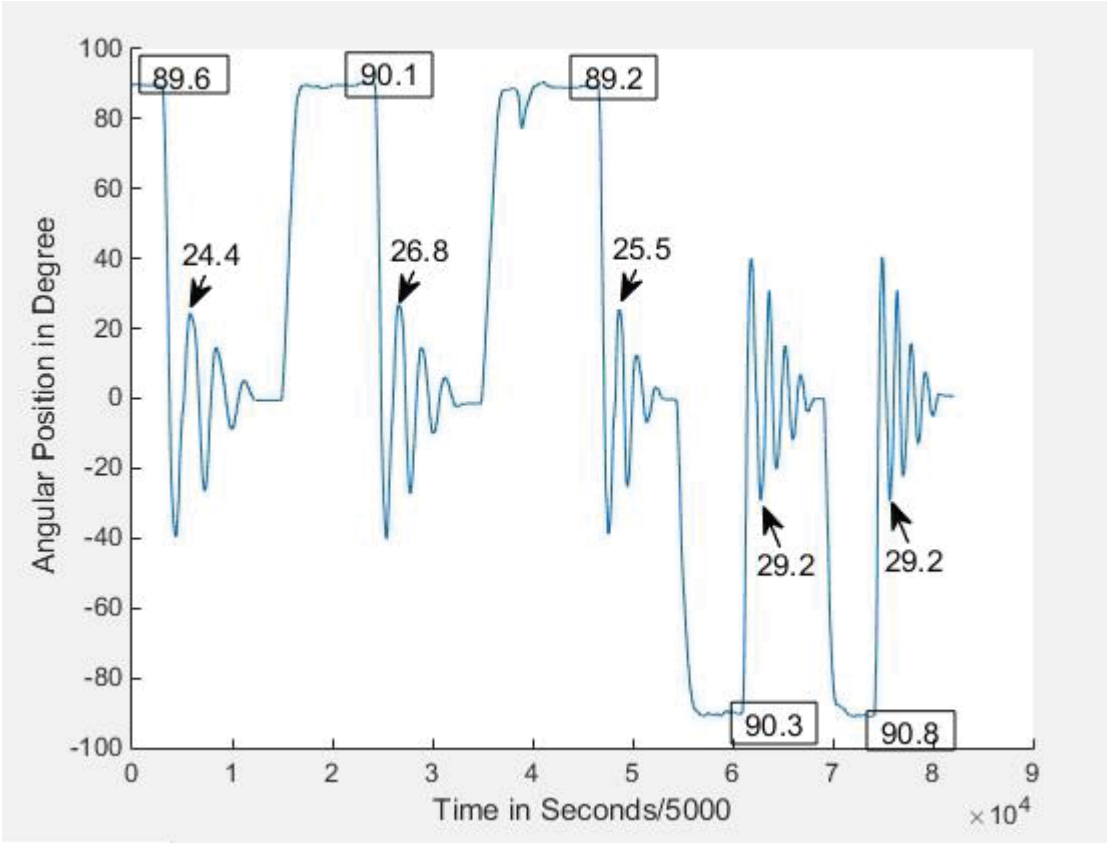


Figure 91: The difference between the results of 5 experiments with physical pendulum

When we drop the pendulum from the right hand side for several times and immediately we drop the pendulum from the left hand side the result is different as it is shown in Figure 91. By considering the first oscillate there is around 4° different for the left and right hand side experiments. A sound from the motor which comes just in one direction can cause this high friction in one side.

Another point in Figure 91 is about the different time periods. If you look at the most left experiment and compare it with the most right experiment, a squeezing in data is happening from the left to the right. The reason is MyRIO can be suitable for incremental encoder but in the other hand, MyRIO is not completely suitable to save continuously measurement data. It is necessary to manage the memory correctly (especially for saving the measurement data continuously). One solution is to define and fix the dimension of each memory block associated for each data measurement.

4.5.1 First Experiment

In this section we run an experiment with the physical pendulum by fixing the joint and dropping the two fixed limbs from 90° and then by simulating the model of a simple pendulum we try to estimate the right values for the coefficients of friction.

Equation (29) for our first experiments has below values.

- m is the total mass of limb1 and limb2, $m_1 + m_2$,
- l_c is the center of mass $m_1$and m_2,
- J is the moment of inertia by considering m and l_c in a flat plate shape.

Figure 92 shows the Simulink file of Equation (29).

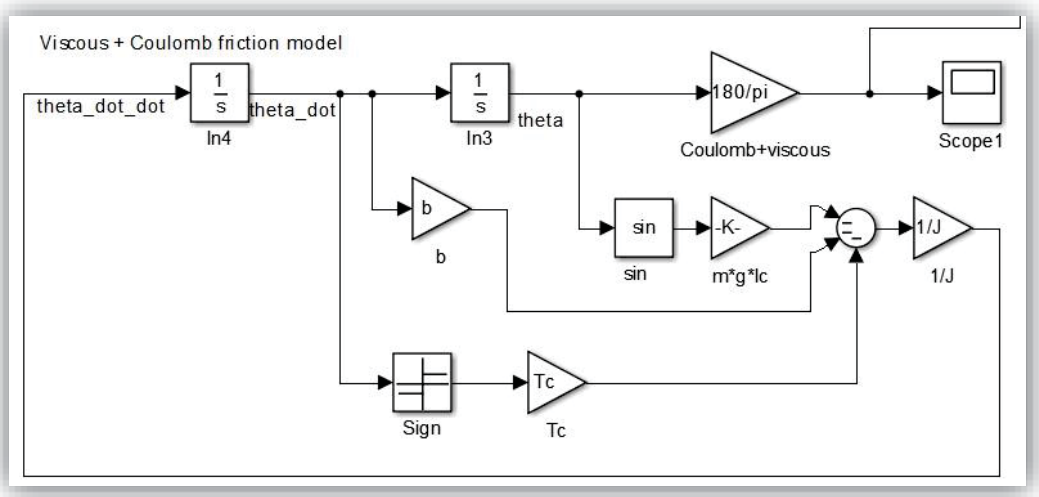


Figure 92: Simulink file of equation (29), a pendulum with 2 types of friction

In the first experiment we measure the angle of pendulum, θ. The results of the first experiment from the encoder and from the simulation is shown in Figure 93.

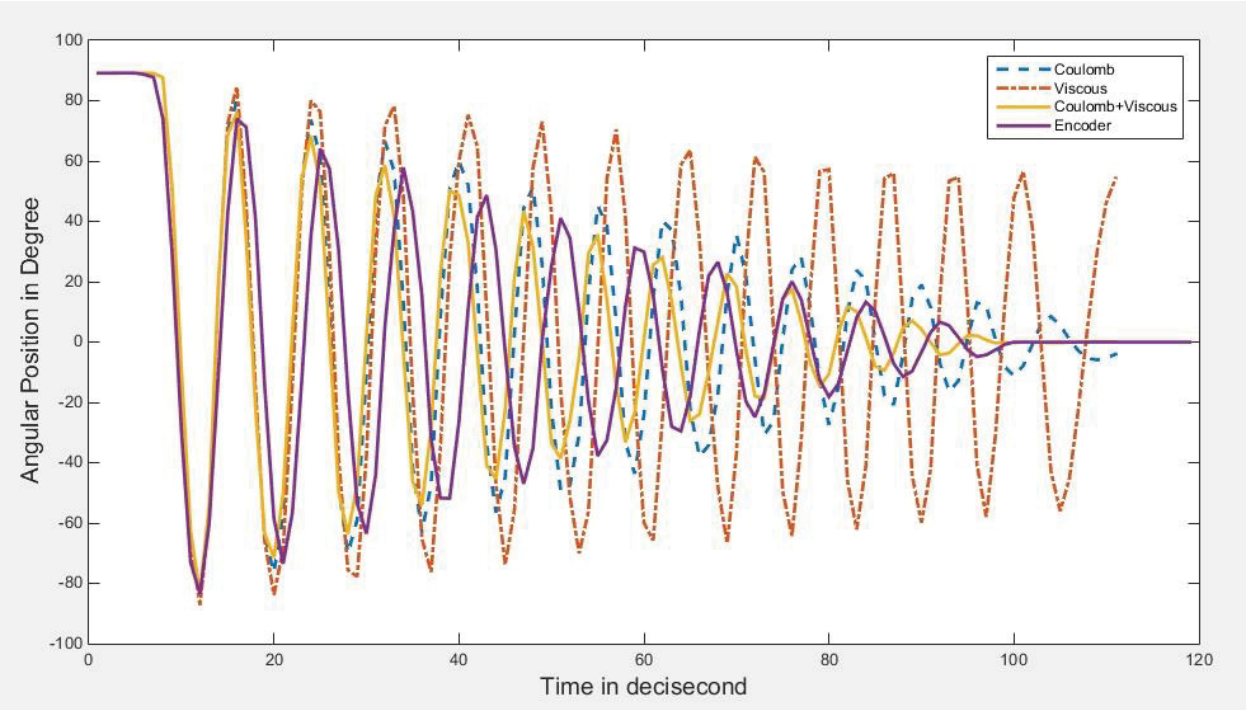


Figure 93: The results of Simulation and Encoder for the first experiment

If just viscous and coulomb friction acts as the friction force on the pendulum the pendulum will reacts like the dashed lines in Figure 93. The yellow line is the combination of both frictions. The purple line is also the result of the experiment on the physical pendulum from the encoder.

The results of simulation and encoder have similar pattern. By tuning the values for coefficient of viscous and coulomb friction we found the closest values is 0.00025 for viscous and 0.0047 for coulomb.

The above values are coefficients of friction between the motor and the first limb and now for the friction between the first and the second limb we run the second experiment.

4.5.2 Second Experiment

For the second experiment we fix the first limb and then we drop the second limb from 90°. Again by simulating the model of a simple pendulum we try to estimate the right values for the coefficients of friction between the two limbs.

This time we use the below values for Equation (29).

- m is the total mass of limb2, $m_1 + m_2$,
- l_c is the center of mass m_2,
- J is the moment of inertia by considering m and l_c in a flat plate shape.

For the second limb we do not have the encoder and any reliable sensor to measure the angular position precise, then we recorded a video for the second experiment. The first scene of the video is shown in Figure 94.

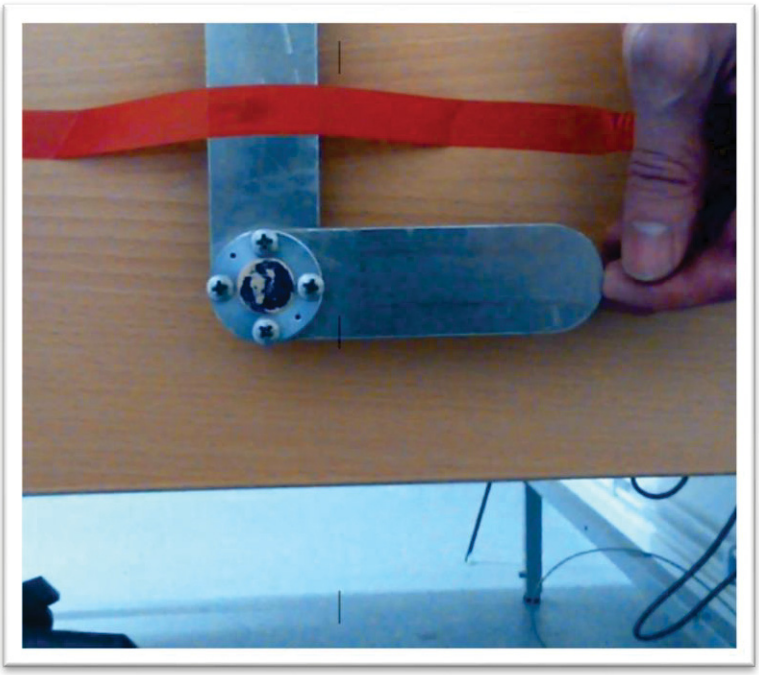


Figure 94: A scene from the recorded video for the second experiment

By playing the video with low speed, we emerge some key parameters like the duration time of the experiment, the number of zero crossings in plus and minus degrees and also the final degree. The results are shown in Table 12.

Table 12: Comparing the results from the video file and Simulation for second experiment

	Physical experiment	Simulation results
Time	Around 3 Seconds	Around 3 Seconds
Zero crossing	9	9
Turning points in –	5	5
Turning points in +	4	4
Final degree	Less than 1°	0.3°

By considering the parameters of Table 12, we tried to estimate the Values for coefficients of friction. The results of simulation for this experiment in absence of the encoder information are shown in Figure 95.

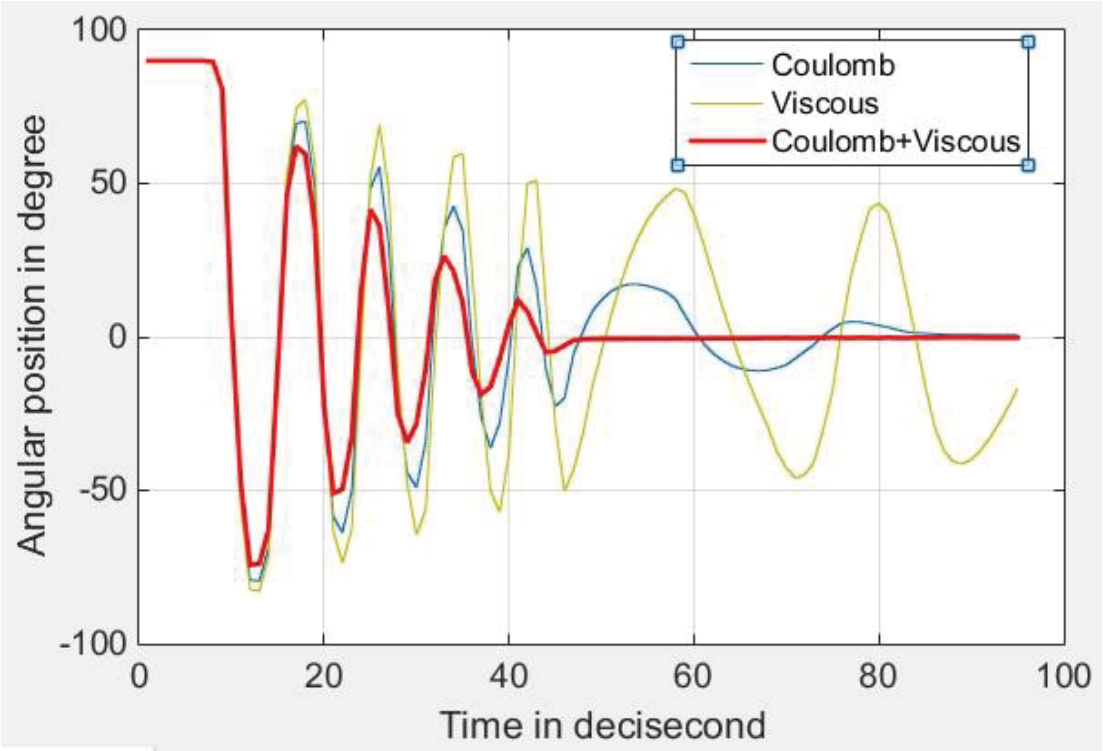


Figure 95: Simulation results of the second experiment

The difference of Figure 93 and Figure 95 is the lack of encoder data in Figure 95. Instead of comparing the simulation results with encoder data a recorded video was used which the results of that experiments came to Table 12.

The values, 0.000043 for viscous friction and 0.0009 for coulomb friction are the nearest values for coefficients of friction between the two limbs.

4.6 Simulating the nonlinear model

For simulating the system we used MATLAB 8.4 (R2014b) due to powerful analysis tools. Also we do not use a fixed step size for the solver, instead we use a variable-step size with relative tolerance 0.001. Our solver is the default of Simulink/MATLAB, ode45 (Dormand-Prince).

We simulate our system by using equations (23), (24) and (25) directly. The Simulink file is shown in Figure 96.

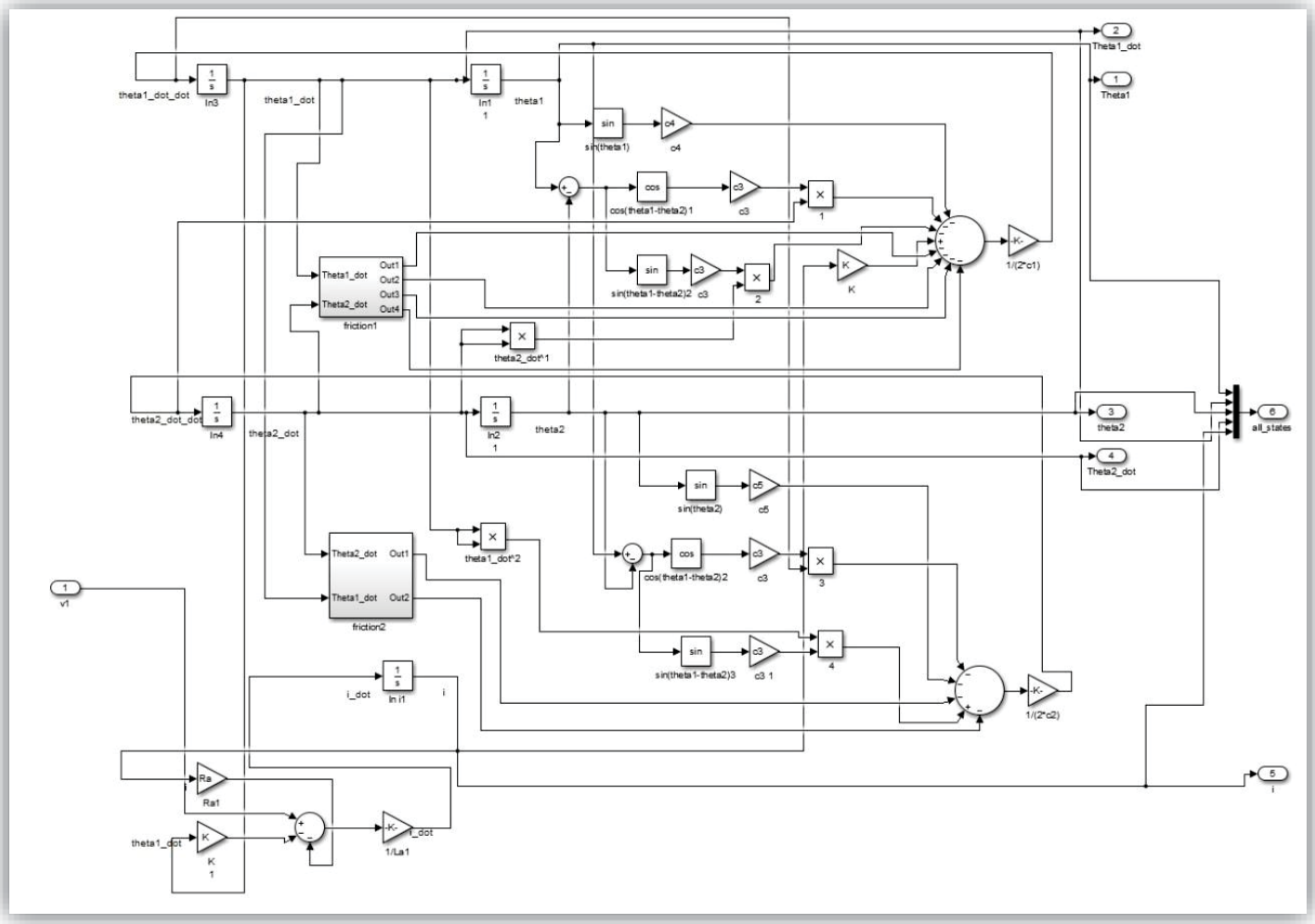


Figure 96: The nonlinear model in Simulink/MATLAB

In Figure 96, 5 integrators are exist for our 5 states, $(\theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2, i)$. The two top integrators for θ_1, Equation (23), the two middle integerators for θ_2, Equation (24) and one down integerator for i, Equation (25). Also for making s impler and readable figure, we put the friction parts of Equations (23) and (24) in two blocks. Inside of the blocks are shown in Figure 97 and Figure 98.

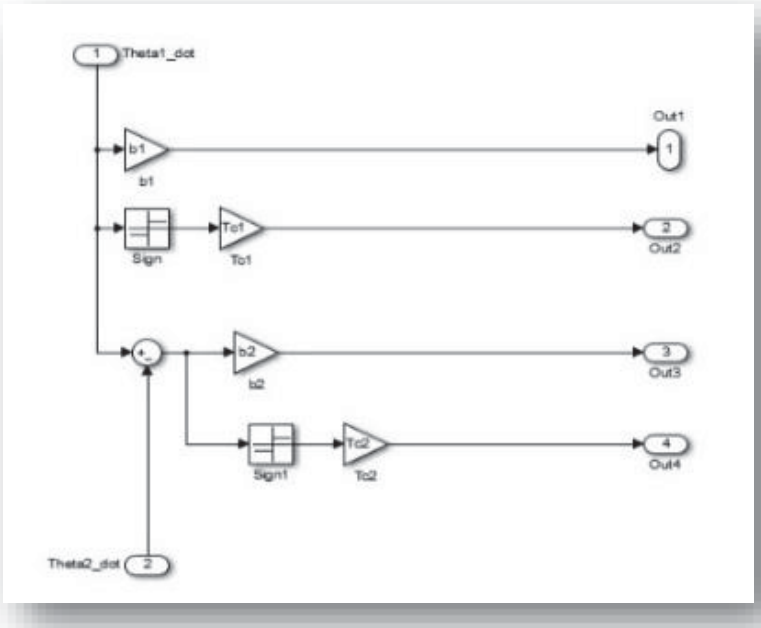


Figure 97: Inside of the subsystem-Friction1 in Figure 96

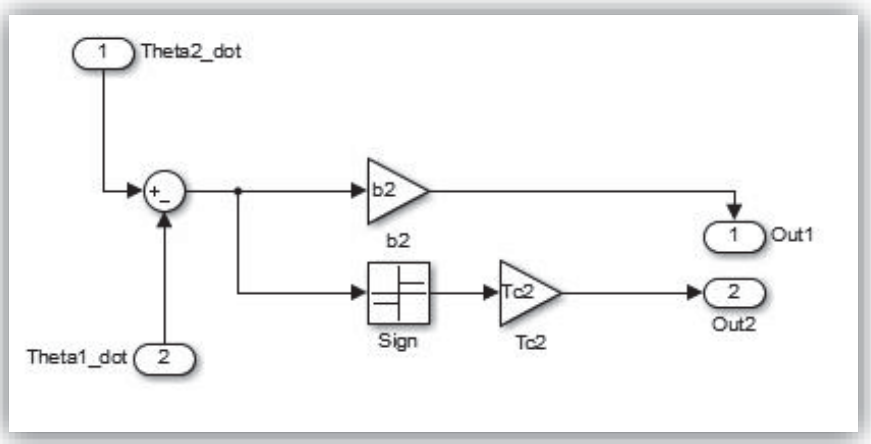


Figure 98: Inside of the subsystem-Friction2 in Figure 96

Figure 97 and Figure 98 show the friction part of the motion equations that discussed in Friction part.

Figure 96 shows the five states $(\theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2, i)$ for our system. If we run this model without the input and also with zero initial conditions nothing should be happen. This can be the first and the simplest test for checking if the model and Simulink blocks work well or not. The result of simulation is depicted on Figure 99.

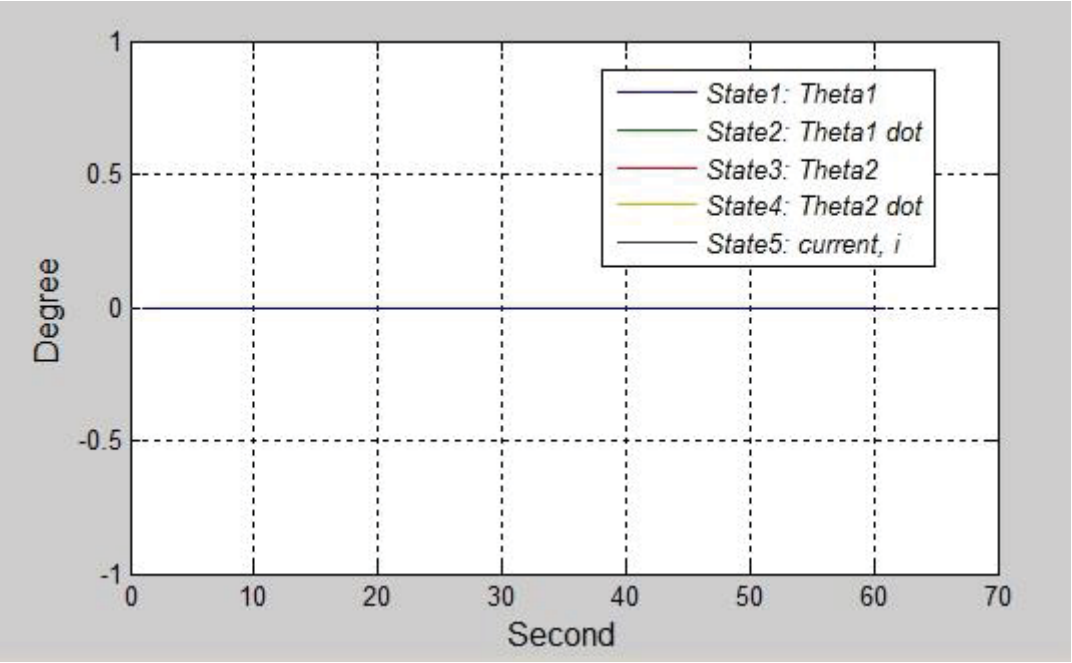


Figure 99: Simulation results of nonlinear model without input and 0 initial conditions

According to Figure 99 the pendulum is constant because state (0,0,0,0,0) is our stable equilibrium point and without any input the pendulum should not move and stay on stable point.

By changing the initial condition and in the case of any input the pendulum should return to the stable equilibrium point. An example for initial condition (-168°,0, -168°,0,0) is showed in Figure 100.

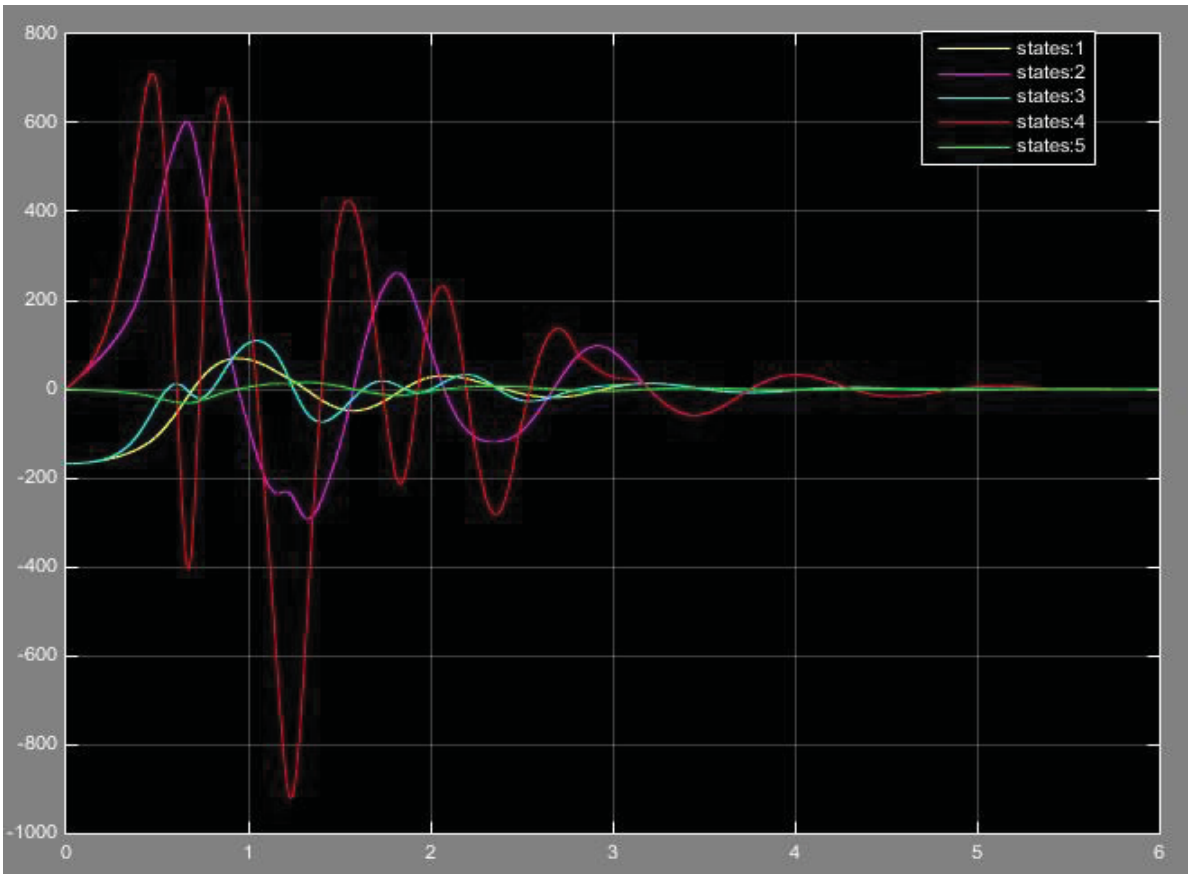


Figure 100: The result of nonlinear model without input and with initial condition (-168°,0, -168°,0,0)
(X-axis is second, Y-axis is degree)

In Figure 100, states 1 till 5 are the system`s state spaces $(\theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2, i)$ respectively. State 1, yellow line and state2, cyan line are the position of the first and the second limb respectively. At second 0 they are in initial condition -168° and immediately after dropping the pendulums are trying to come to stable equilibrium point 0°. After a few oscillation and during around 5 seconds the pendulums are in their stable point.

Now we need to check if the results of the nonlinear model are match with the physical pendulum or not. This process called verification process. For verifying the model we run an experiment and we check the results from the Simulation and the Encoder.

For verifying the model with our physical system we faced two problems.

- **First Problem**, by doing an experiment with the physical system several times, the results are not completely the same as we mentioned in Section 5.5. Then we do not have a constant result to verify our model,
- **Second Problem**, we did an experiment like dropping the pendulum from +90°, in this case the pendulum in the first oscillation could reach to around -40° but for the model it was not possible. Even if we neglect the friction we just could reach to around -23° and it means some value inside of the model is wrong.

The reason of above problems as we mentioned could be a sound from the brushes inside of the motor that can be hear just in one direction. This sound is just for our specific motor and it is not related to other Maxon motors.

Also we found by increasing the electric resistance of the motor from 0.608Ω to 1.73Ω we could verify the model. Electric resistance has a big effect in the result. The differences are shown in Figure 101 for 3 values, 0.608Ω which is the company`s datasheet value, 1.000Ω and 1.730Ω which is seems to be the right value for our motor.

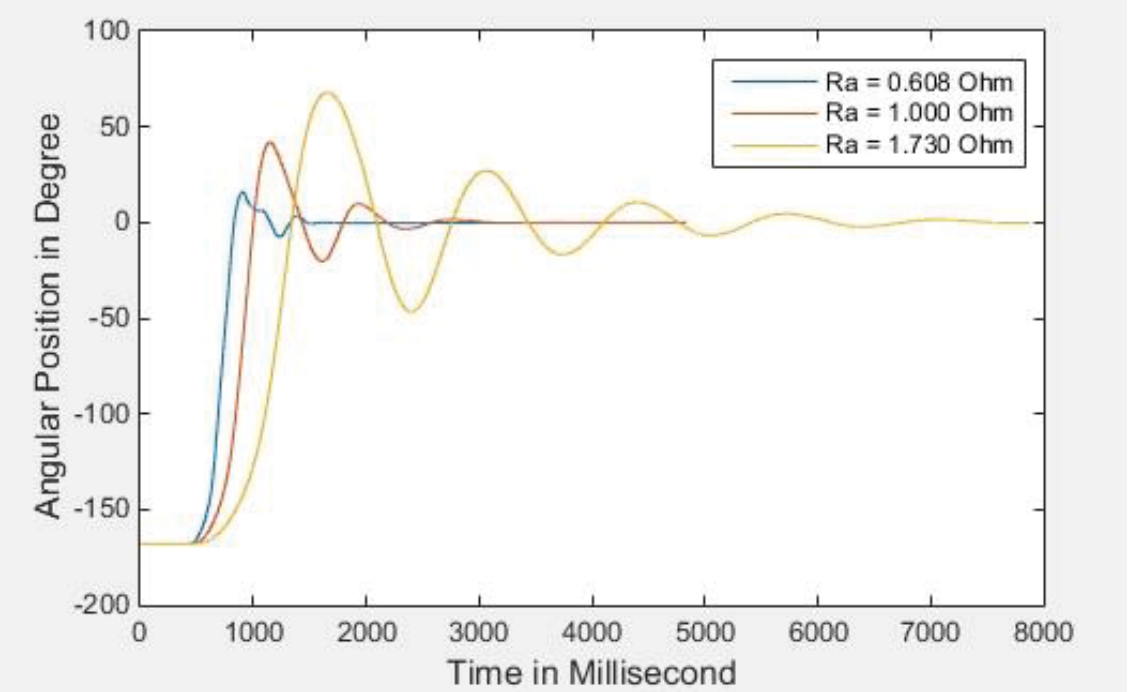


Figure 101: The effect of different values for Electric Resistance

Figure 101 shows the simulation result of nonlinear model for position of the first limb θ_1. By changing the value of motor`s electric resistance the behavior of the pendulum can change dramatically. In our case we put the initial conditions to -180°. For the datasheet`s value the pendulum in the first oscillation reaches to around +15° and for one ohm reaches to around +45° and for 1.730Ω reaches to around +70°.

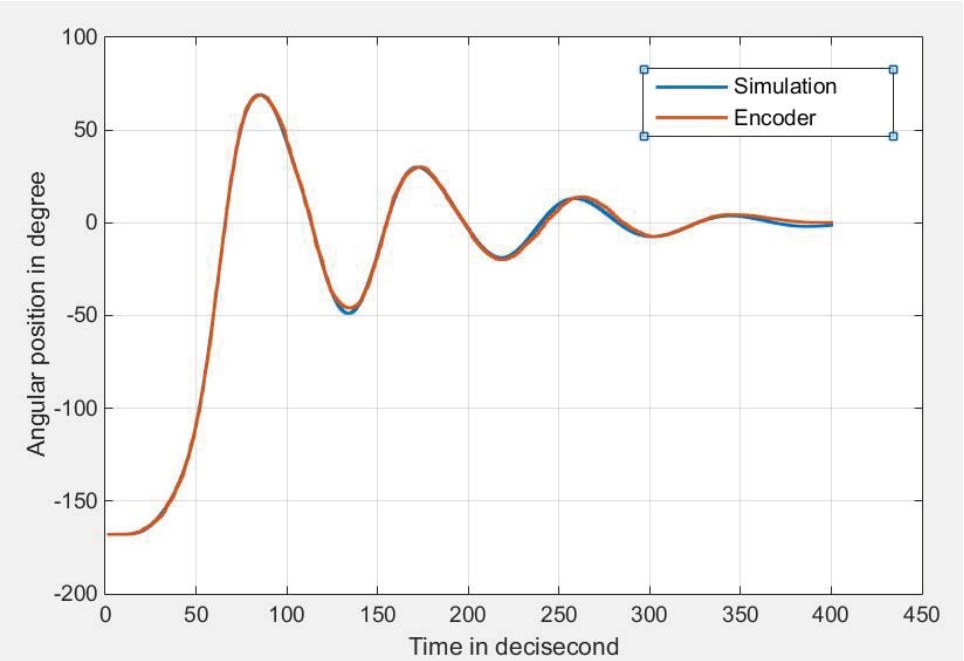


Figure 102: The angular position of the first limb, θ_1 from the encoder and simulation for initial condition (-168°,0, -168°,0,0)

By changing the value of motor`s electric resistance to 1.730Ω, nonlinear model verified with the physical system. The results of verification is shown in Figure 102.

Figure 102 is just contain the simulation and encoder data of the first limb because for the second limb we did not have a reliable sensor. The results of the simulation can follow the encoder data. The difference between the encoder and the simulation results are shown in Figure 103.

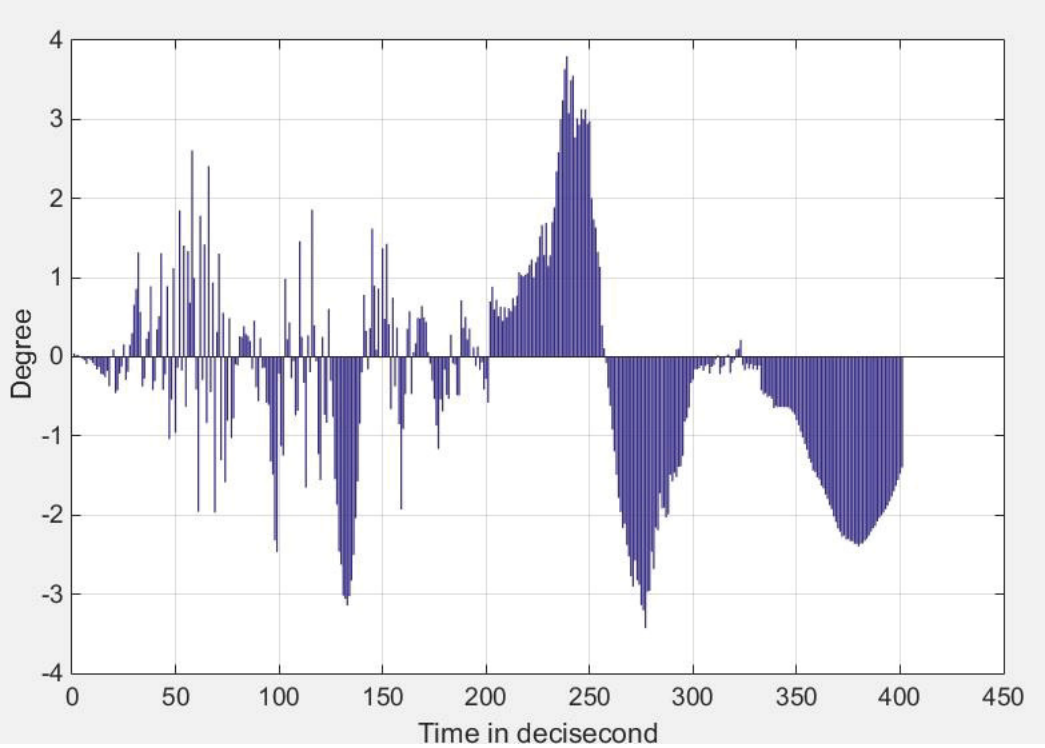


Figure 103: The difference between the encoder and the simulation results

The showed errors in Figure 103 are coming from the listed sources in below.

- The estimated values inside of the model e.g. motor resistance,
- The estimated coefficients of friction,
- The case of the system which connects the motor to the table has a small degree around 1° or 2° with the table,
- The case of the system which is connected to the table needs to be tight enough for preventing of vibration,
- The errors from the DAQ.

4.7 Linearized model

For linearizing our system we will rewrite Equations (26), (27) and (28) somehow $\ddot{\theta_1}$ and $\ddot{\theta_2}$ are in one side and the remains are in another side. The equations are come to the form of Equation (30), (31) and (32).

$$\begin{aligned} 2c_1\ddot{\theta_1} + c_3\cos(\theta_1 - \theta_2)\,\ddot{\theta_2} \\ = -c_4 sin\theta_1 - c_3\dot{\theta_2}^2 sin(\theta_1 - \theta_2) + Ki_a - b_1\dot{\theta}_1 - T_{c1}sign(\dot{\theta}_1) \\ - b_2(\dot{\theta_1} - \dot{\theta_2}) - T_{c2}sign(\dot{\theta_1} - \dot{\theta_2}) \end{aligned} \tag{30}$$

$$\begin{aligned} 2c_2\ddot{\theta_2} + c_3\cos(\theta_1 - \theta_2)\,\ddot{\theta_1} \\ = -c_5 sin\theta_2 + c_3\dot{\theta_1}^2 sin(\theta_1 - \theta_2) - b_2(\dot{\theta_2} - \dot{\theta_1}) - T_{c2}sign(\dot{\theta_2} - \dot{\theta_1}) \end{aligned} \tag{31}$$

$$L_a\dot{i_a} = -R_a i_a - K\dot{\theta}_1 - V \tag{32}$$

Linearization is around the equilibrium points and we have 4 equilibrium points as are shown in Table 13.

Table 13: The equilibrium points

	states	Limb 1	Limb 2
1	(0,0,0,0)	Down	Down
2	$(0,0,\pi,0)$	Down	Up
3	$(\pi,0,0,0)$	Up	Down
4	$(\pi,0,\pi,0)$	Up	Up

Our interested equilibrium points are points 1 (down-down) and 4 (up-up).

4.7.1 Linearizing around down position

For down position, (0,0,0,0) we can do the below simplifications.

- $cos(\theta_1 - \theta_2) = 1$,
- $sin(\theta_1 - \theta_2) = 0$,
- $sin\theta_1 = \theta_1$,
- $sign(\theta) = 0$.

Since $sign(\theta)$, the signum function is discontinues (Figure 104) we cannot linearize it but this function around 0 is similar to $arctan$ function and in some cases for a limited range we can replace it with $arctan$ (Figure 105).

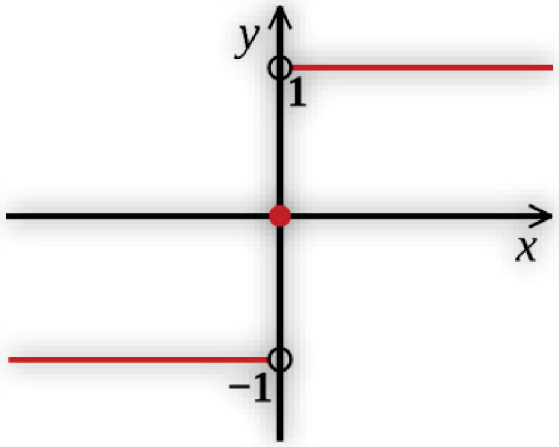


Figure 104: The Signum function

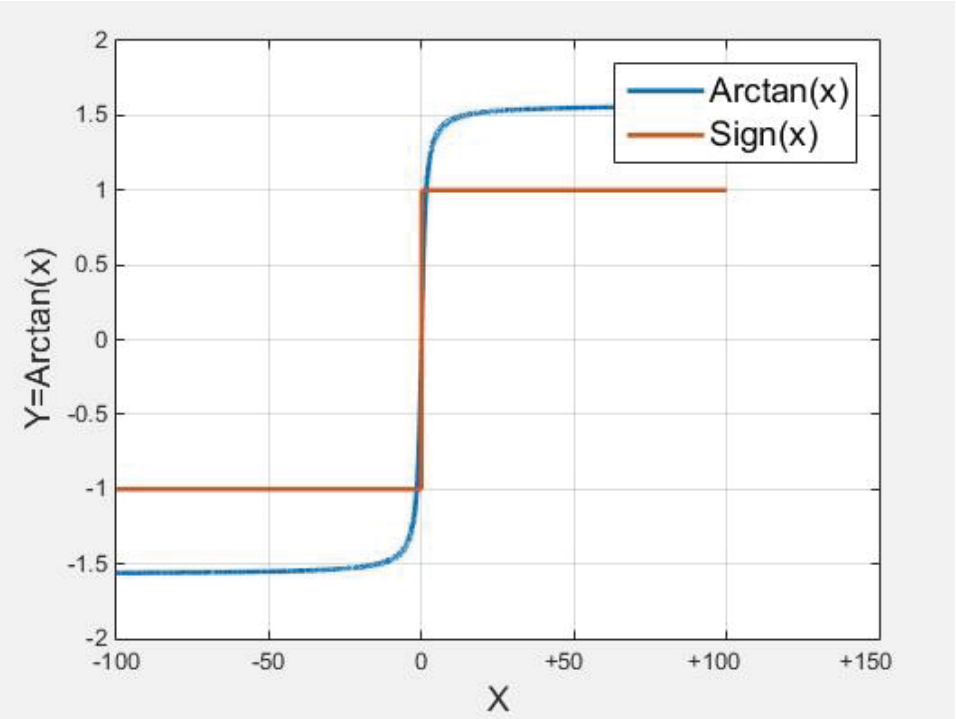


Figure 105: The arctan and sign function

For linearizing *arctan* we can use the Taylor-series expansion which yields the below result.

$$arctan(x) = tan^{-1}x_0 + \frac{d(tan^{-1}x)}{dx} + \cdots = 0 + x + 0 - \frac{x^3}{3}$$

If we consider the first three parts, it will be just x. As for small values around 0, $arctan(x)$ and $sin(x)$ are similar (See Figure 106), then x can be roughly replaced with $sin(x)$.

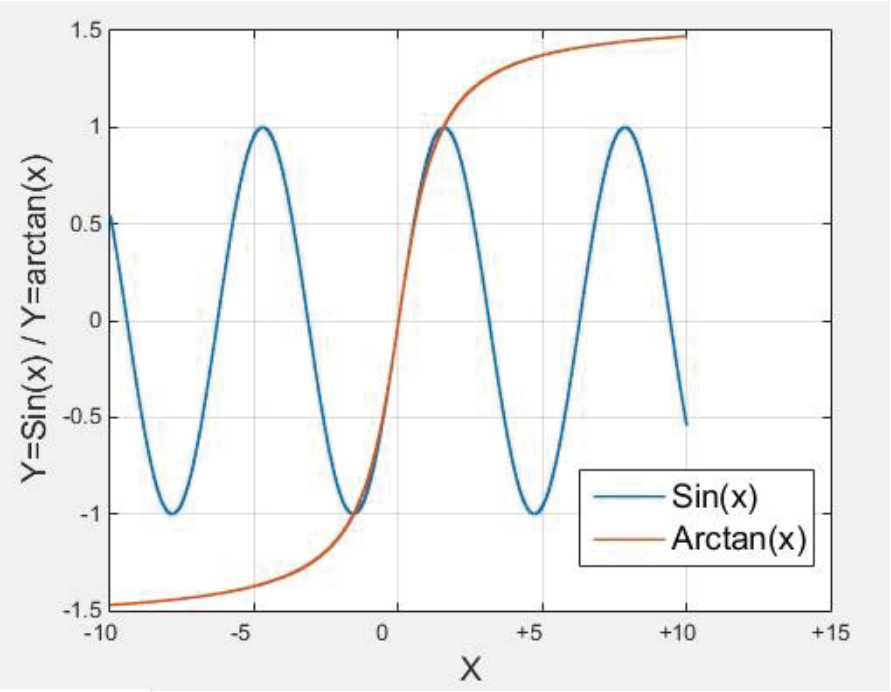


Figure 106: The plot of arctan(x) and sin(x)

In our case we prefer to neglect the signum function. Our linearized model for our stable equilibrium point will be Equation (33).

$$\begin{pmatrix} 2c_1 & c_3 & 0 \\ c_3 & 2c_2 & 0 \\ 0 & 0 & L_a \end{pmatrix}\begin{pmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \dot{i} \end{pmatrix} = \begin{pmatrix} -c_4 & -b_1 - b_2 & 0 & b_2 & K \\ 0 & b_2 & -c_5 & -b_2 & 0 \\ 0 & -K & 0 & 0 & -R_a \end{pmatrix}\begin{pmatrix} \theta_1 \\ \dot{\theta}_1 \\ \theta_2 \\ \dot{\theta}_2 \\ i \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} v \qquad (33)$$

If we name matrix $\begin{pmatrix} 2c_1 & c_3 & 0 \\ c_3 & 2c_2 & 0 \\ 0 & 0 & L_a \end{pmatrix} = E_d$ then the Equation 33 will change to Equation 34.

$$\begin{pmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \dot{i} \end{pmatrix} = {E_d}^{-1}\begin{pmatrix} -c_4 & -b_1 - b_2 & 0 & b_2 & K \\ 0 & b_2 & -c_5 & -b_2 & 0 \\ 0 & -K & 0 & 0 & -R_a \end{pmatrix}\begin{pmatrix} \theta_1 \\ \dot{\theta}_1 \\ \theta_2 \\ \dot{\theta}_2 \\ i \end{pmatrix} + {E_d}^{-1}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} v \qquad (34)$$

By adding two other states to the left hand side we will have our final form of our linearized model in Abcd from as it shown in Equation (35).

$$\begin{pmatrix} \dot{\theta}_1 \\ \ddot{\theta}_1 \\ \dot{\theta}_2 \\ \ddot{\theta}_2 \\ \dot{i} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ -c_4/E_d & (-b_1 - b_2)/E_d & 0 & b_2/E_d & K/E_d \\ 0 & 0 & 0 & 1 & 0 \\ 0 & b_2/E_d & -c_5/E_d & -b_2/E_d & 0 \\ 0 & -K/E_d & 0 & 0 & -R_a/E_d \end{pmatrix}\begin{pmatrix} \theta_1 \\ \dot{\theta}_1 \\ \theta_2 \\ \dot{\theta}_2 \\ i \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1/E_d \end{pmatrix} v \qquad (35)$$

Through Equation (35) the parameters and states are easy to read from the Abcd from,

$$\dot{X} = AX + bu$$

Where X is our states $\begin{pmatrix} \theta_1 \\ \dot{\theta}_1 \\ \theta_2 \\ \dot{\theta}_2 \\ i \end{pmatrix}$ and u in our case is v.

By checking the eigenvalues of matrix A which are 1.0e+03 * $\begin{bmatrix} -2.8339 + 0.0000\text{i} \\ -0.0012 + 0.0058\text{i} \\ -0.0012 - 0.0058\text{i} \\ -0.0006 + 0.0112\text{i} \\ -0.0006 - 0.0112\text{i} \end{bmatrix}$, we can understand that our linearized system in the down position is asymptotically stable because all of the eigenvalues are located in the left hand side of the complex plane.

4.7.2 Linearizing around up position

For the up position$(\mp\pi, 0, \mp\pi, 0)$, we can do the below simplifications.

- $cos(\theta_1 - \theta_2) = -1$
- $sin(\theta_1 - \theta_2) = 0$
- $sin\theta_1 = \pi - \theta_1$
- $sin\theta_2 = \pi - \theta_2$
- $sign\theta_1 = 0$
- $sign(x) = 0$

Then our linearized model for our unstable equilibrium point (up position) will be Equation (36)

$$\begin{pmatrix} 2c_1 & -c_3 & 0 \\ -c_3 & 2c_2 & 0 \\ 0 & 0 & L_a \end{pmatrix} \begin{pmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \dot{i} \end{pmatrix} = \begin{pmatrix} c_4 & -b_1 - b_2 & 0 & b_2 & K \\ 0 & b_2 & c_5 & -b_2 & 0 \\ 0 & -K & 0 & 0 & -R_a \end{pmatrix} \begin{pmatrix} \theta_1 - \pi \\ \dot{\theta}_1 \\ \theta_2 - \pi \\ \dot{\theta}_2 \\ i \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} v \qquad (36)$$

Again if we name matrix $\begin{pmatrix} 2c_1 & -c_3 & 0 \\ -c_3 & 2c_2 & 0 \\ 0 & 0 & L_a \end{pmatrix} = E_u$ then the Equation 36 will change to Equation 37.

$$\begin{pmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \dot{i} \end{pmatrix} = {E_u}^{-1} \begin{pmatrix} c_4 & -b_1 - b_2 & 0 & b_2 & K \\ 0 & b_2 & c_5 & -b_2 & 0 \\ 0 & -K & 0 & 0 & -R_a \end{pmatrix} \begin{pmatrix} \theta_1 - \pi \\ \dot{\theta}_1 \\ \theta_2 - \pi \\ \dot{\theta}_2 \\ i \end{pmatrix} + {E_u}^{-1} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} v \qquad (37)$$

And by adding the 2 other states to the left hand side we will have our final form of our linearized model in Abcd from as it shown in Equation (38).

$$\begin{pmatrix} \dot{\theta}_1 \\ \ddot{\theta}_1 \\ \dot{\theta}_2 \\ \ddot{\theta}_2 \\ \dot{i} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ c_4/E_u & (-b_1 - b_2)/E_u & 0 & b_2/E_u & K/E_u \\ 0 & 0 & 0 & 1 & 0 \\ 0 & b_2/E_u & c_5/E_u & -b_2/E_u & 0 \\ 0 & -K/E_u & 0 & 0 & -R_a/E_u \end{pmatrix} \begin{pmatrix} \theta_1 - \pi \\ \dot{\theta}_1 \\ \theta_2 - \pi \\ \dot{\theta}_2 \\ i \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1/E_u \end{pmatrix} v \qquad (38)$$

For checking the stability of this point we take a look on the eigenvalues of matrix A which are 1.0e+03 * $\begin{bmatrix} -2.8339 \\ -0.0120 \\ -0.0071 \\ 0.0048 \\ 0.0109 \end{bmatrix}$. Two eigenvalues are located in the right hand side of complex plane and this means instability for the up position of the pendulum.

4.8 Control law

Our strategy for designing the feedback controller is pole placement gain using Ackermann's formula.

Consider a linear dynamic system in the state space form

$$\dot{X} = AX + bu$$

$$\mathrm{Y} = c^T X + du$$

In some cases one is able to achieve the goal (e.g. stabilizing the system or improving its transient response) by using the full state feedback, which represents a linear combination of the state variables, that is

$$\mathrm{u} = -FX$$

So that the closed-loop system, given by

$$\dot{X} = (A - bF)X$$

$$\mathrm{Y} = c^T X$$

Has the desired specifications.

The main role of state feedback control is to stabilize a given system so that all closed-loop eigenvalues are placed in the left half of the complex plane. Then the place of desire eigenvalues are our choice that define the speed of our controller.

The codes for MATLAB are shown in below.

```
% Control law
P1(1:4) = -12; %desired eigenvalues
[kdown] = acker(Adown, bdown, P1)
[kup] = acker(Aup, bup, P1)
```

Where

- `Adown` is the matrix A of our linearized model in state space form,
- `b` is the matrix b of our linearized model in state space form,
- `acker` is a MATLAB function

Instead of using `acker` function we can write a function with ourselves. We named this function `sscontroller1` and in below the codes are shown.

```
function [k, T] = sscontroller1(A, b, EW)
 n=length(A);
Cu=ctrb(A,b);
 if rank(Cu)== n ;
% system is controlaable
k=[];
en=[zeros(n-1,1);1]; % (fliplr(eye(1,n)))'
```

```
t1=(en'*inv(Cu))';
T=[];
T= (ctrb(A',t1))' ;
P=poly(EW);
a=poly(A);
Ks= fliplr(P-a);
Ks(n+1)=[];
k=(Ks*T)' ;
else
    % system is not controlaable
    disp('system is not controlaable')
end
```

Figure 107 shows the two linearized models of our system and also the controller in Simulink/MATLAB.

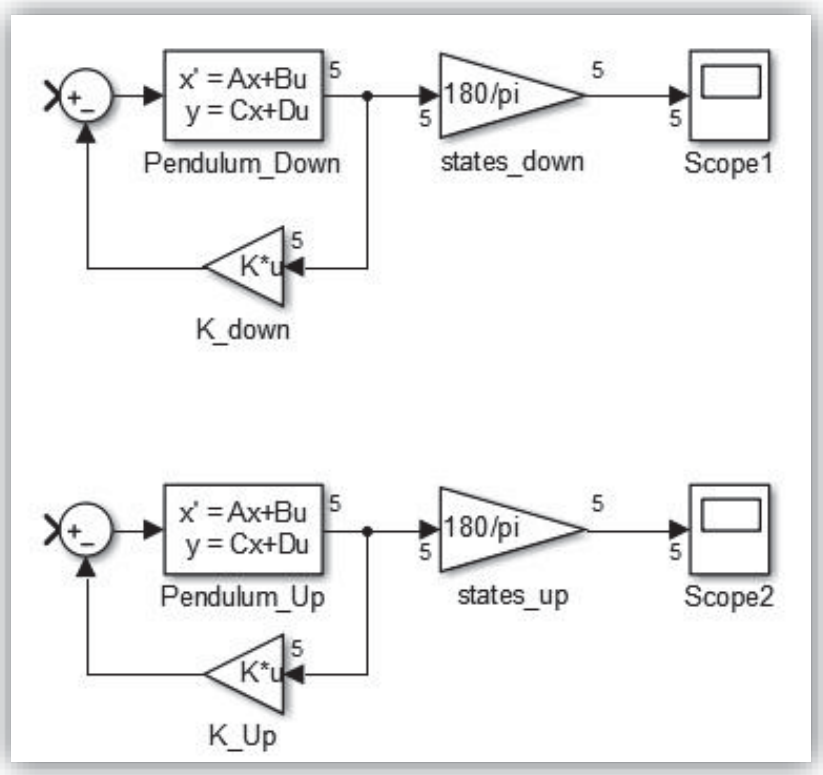


Figure 107: Feedback Controller

By changing the Eigenvalues we can change the speed of the controller and we will see different behavior from the system.

For two linearized models the controllers return the pendulum from the initial condition [90° 0 90° 0 0] to the equilibrium points as it shown in Figure 108 for the down position and in Figure 109 for the up position.

The Eigenvalues which we selected for the controller are [-12 -12 -12 -12 -12].

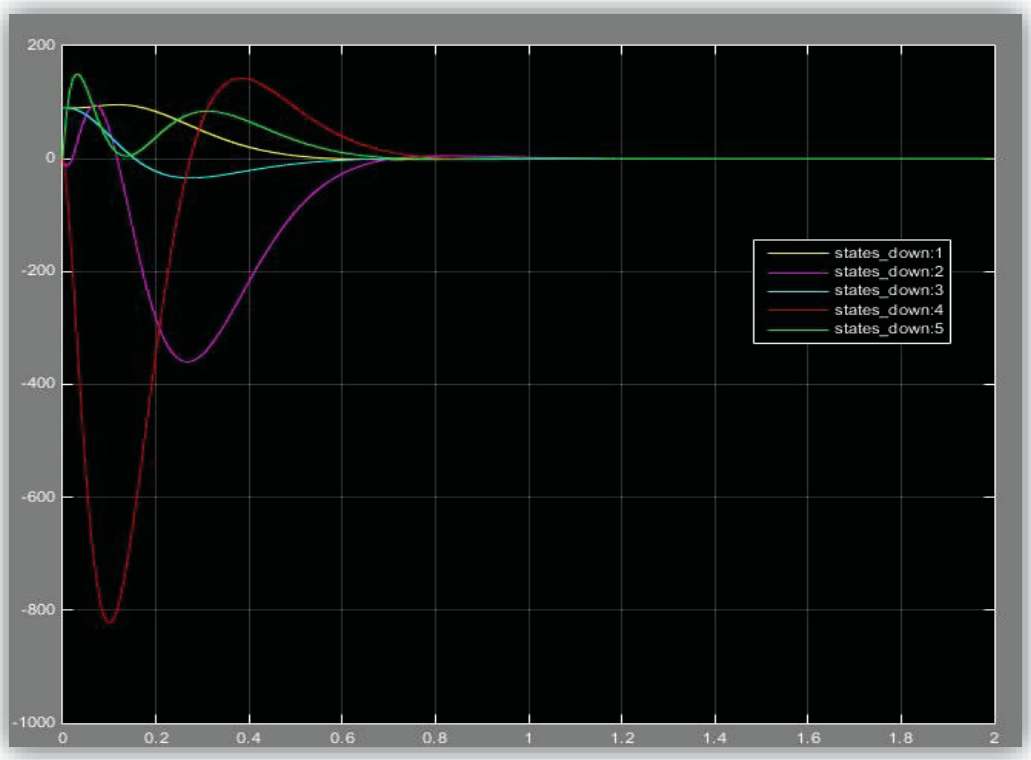


Figure 108: controller bring the pendulum to the down position with the designed controller signal
(X-axis is second, Y-axis is angular position in degree)

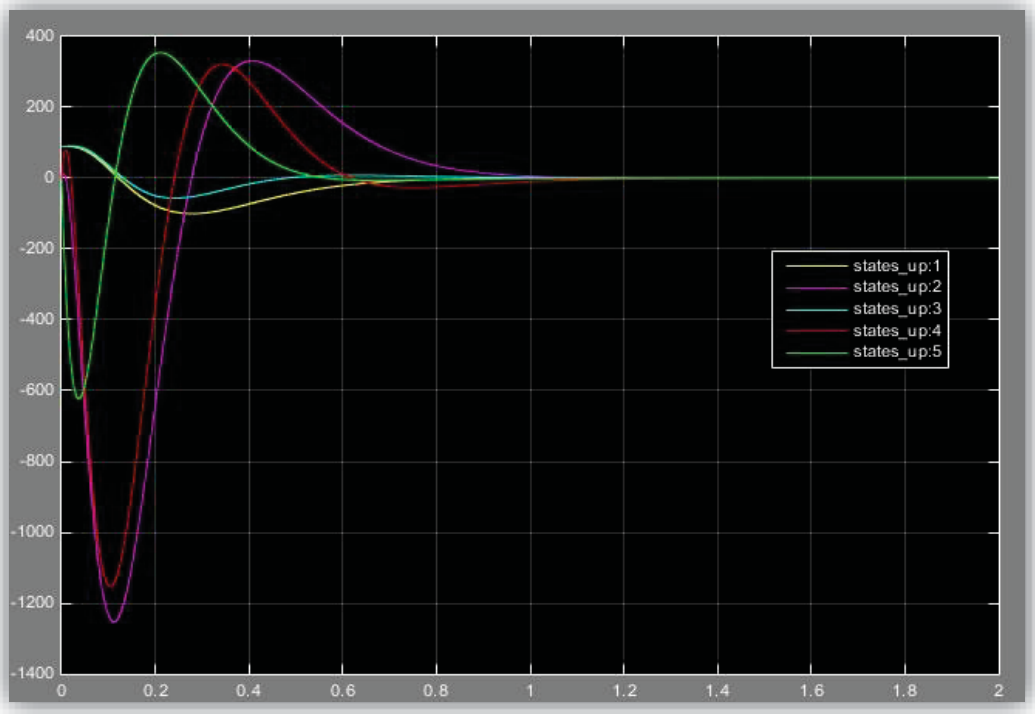


Figure 109: controller bring the pendulum to the up position with the designed controller signal
(X-axis is second, Y-axis is angular position in degree)

Also the interfaces of the controller in LabVIEW are shown in Figure 110 and Figure 111 and the codes are shown in Figure 112 and Figure 113. The ready interface from LabVIEW designed somehow that by inputting the state space matrices and also desired eigenvalues the K values for control law will appear.

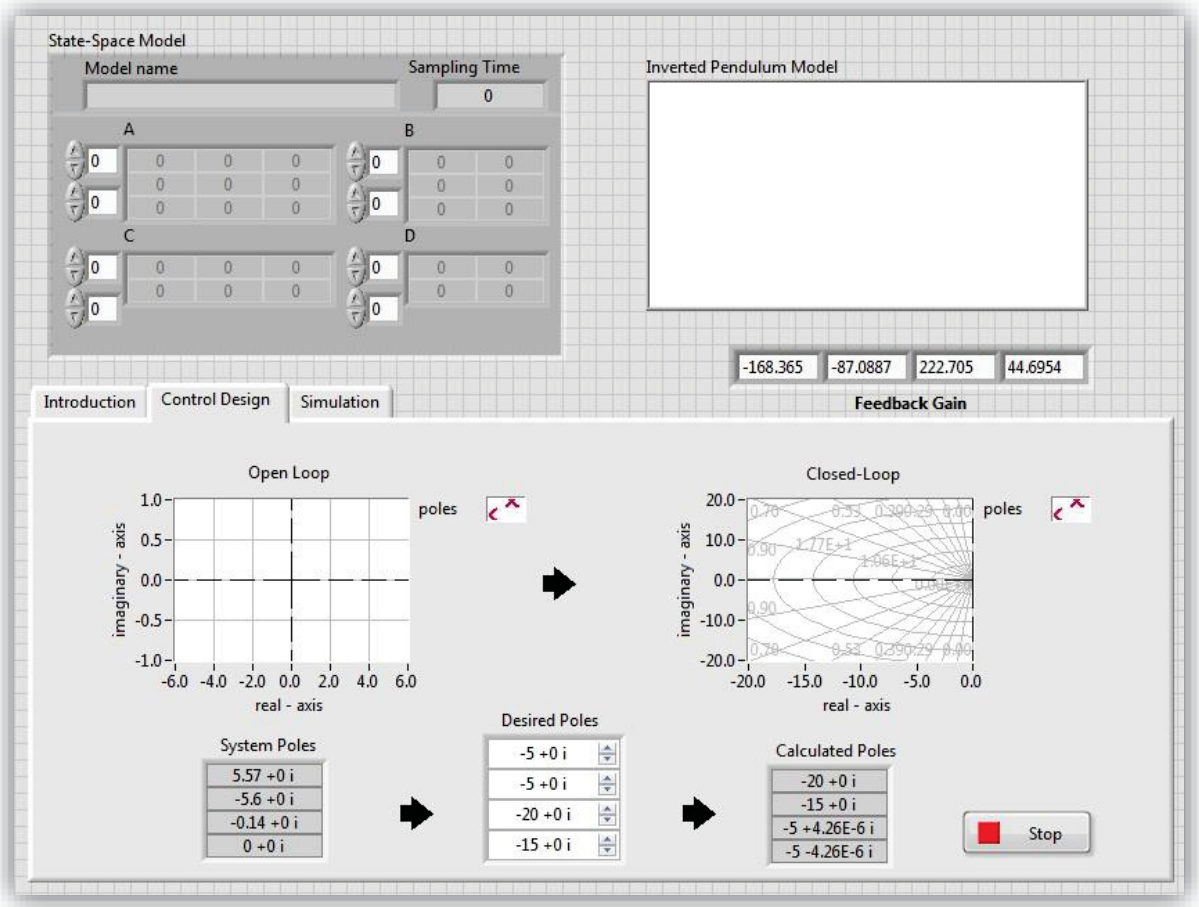


Figure 110: Control design interface of the controller in LabVIEW

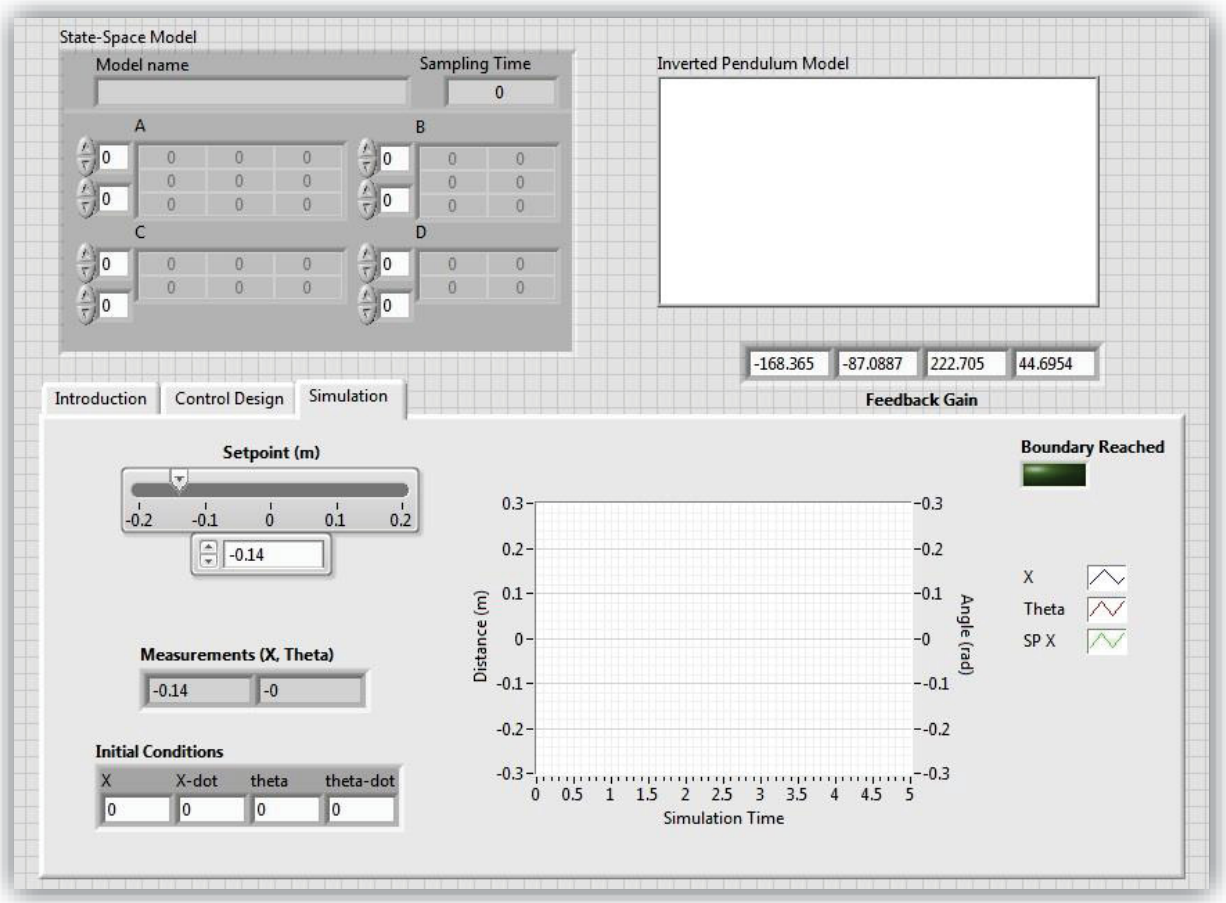


Figure 111: Simulation interface of the controller in LabVIEW

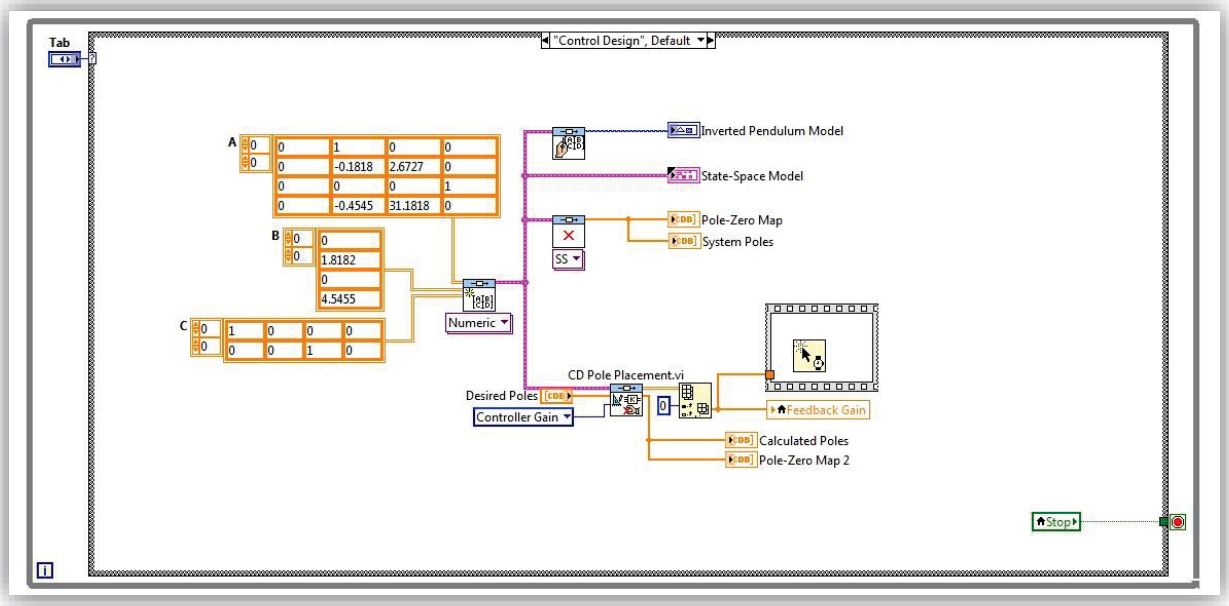


Figure 112: blocks of Lab View's interface for controller part

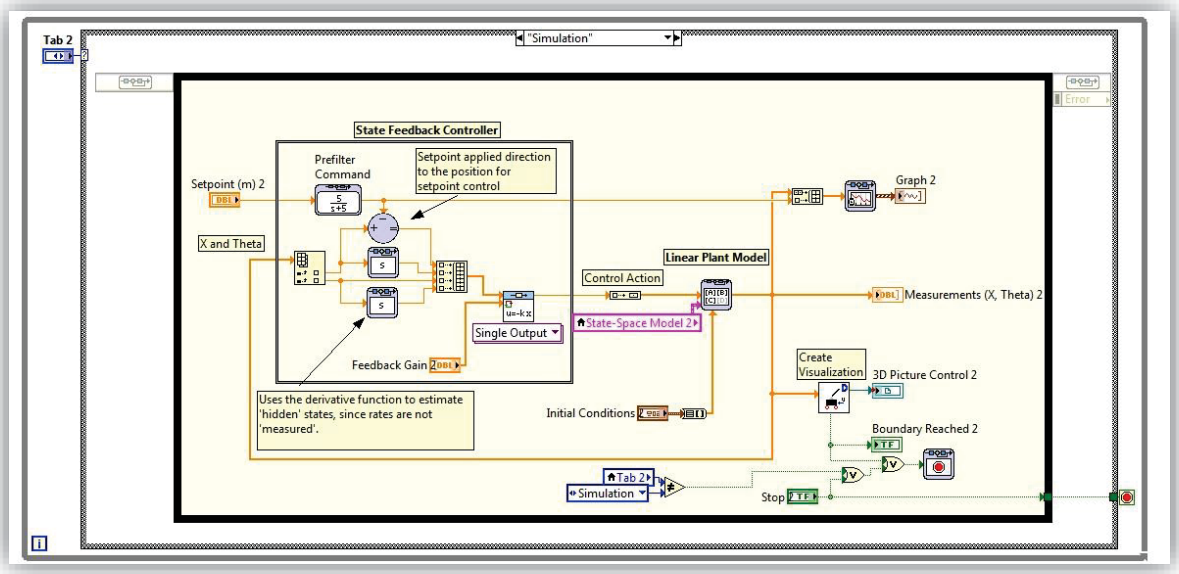


Figure 113: blocks of Lab View's interface for controller part

5 Interface of LabVIEW

5.1 NI MYRIO

NI MyRIO is an embedded device, created by National Instrument. This board is integrate a Xilinx FPGA and a dual-core ARM Cortex-A9 and can be programming with LabVIEW environment. This device has a lot of input/output that can be logical, analog, pwm generator, and some channel are specializing to treat some specific signal like quadrature signal. This board can control a camera by an USB connector.

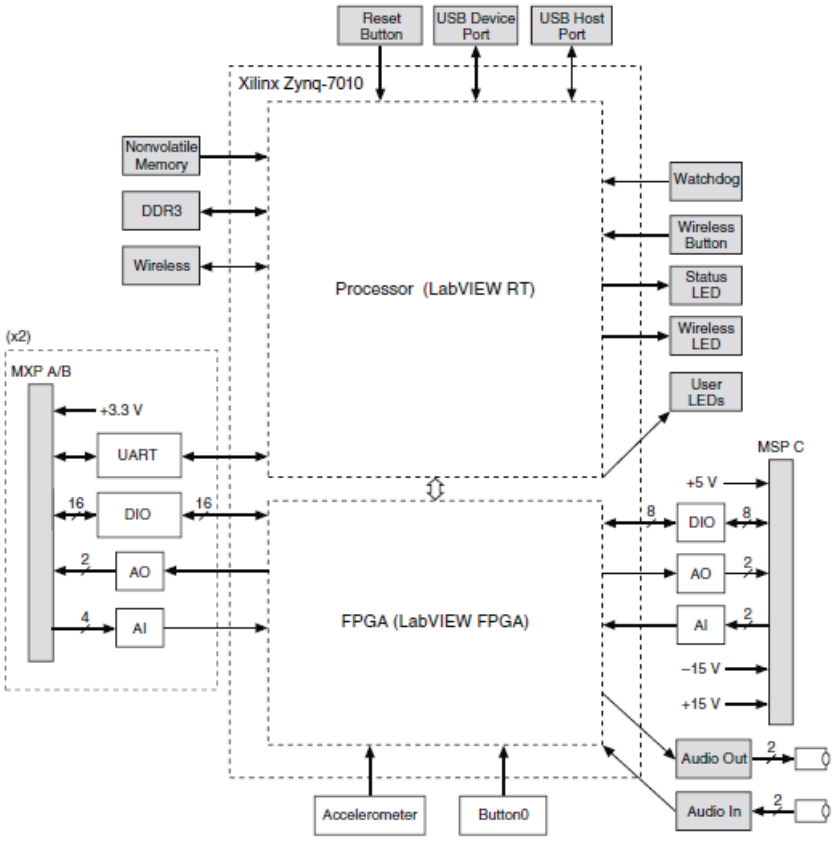


Figure 114: Hardware block diagram from the NI MyRIO datasheet

To program Ni MYRIO from LabVIEW, there are some rules we should follow in order to improve the system performance:

- Have a good management of the memory. If we want to save data from a sensor continuously and to stock all these information (for example, an array which his size will increase continuously for each data acquisition), at one point the memory block associated with the array data will be full, and to save another data another memory block will be used. Changing a memory block will be automatically done but it will take a lot of time and can slow down the system performance. To avoid that, we can define and fix the dimension of each memory block associated for each data. It is especially true when we need to save the image from the camera.
- Do not use Matlab file: .m
- Do not use complex mathematical operations (can take a lot of time)
- Priority for the FPGA function to the Real Time function

5.2 Survey

Ni MyRio is a young embedded device. To help the user to configure MyRIO for their project, a guide was written by NI National Instrument, available at [16].

For the image processing implemented in MyRIO, project [17] contains a vision system for grabbing an image from a USB webcam. Their acquisition is done by using the NI Vision Assistant Express VI, but their system is configured to grab an image each one second (too low for our system).

To control a double inverted pendulum, the company Quanser proposes to students some equipment provided from National Instrument to control inverted pendulum: (from a brochure in this address « download.ni.com/pub/gdc/tut/quanser_ni_brochure_2014.pdf ».

But at this moment there is no work has been done or indexed.

5.3 Structure of the code on LabVIEW

The structure of the code follow the conception rules of the multitask systems (Figure 115). The program can be divided in many tasks that each of them has a specific role (Table 14**Error! Reference source not found.**).

Table 14 : Description of each task of the code on LabVIEW

Task	Priority level	Periodicity	Description
Acquisition Data	1	200 us	- Calculate and refresh the angular displacement of the first limb - refresh the current motor value
Calculation of the speed and the acceleration	2	400 us	- calculate and refresh the speed and the acceleration of the first limb
Control motor	3	500 us	- refresh the voltage motor command value
Waveform generator	4	500 us	- generate a specific waveform which will command the motor.
Feedback Control	5	1ms	- same case as the waveform generator, but the voltage command come from to a Feedback control
Image Processing	6	Variable	- Responsible of the image acquisition and his treatment and analysis in order to get the angular position of the second limb
Interface	7	100 ms	- refresh the LabVIEW interface

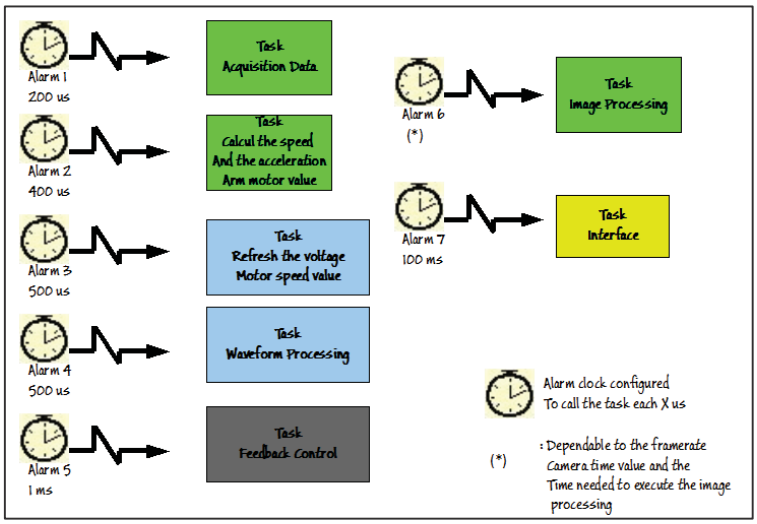


Figure 115: General view of the LabVIEW code

A task in the LabVIEW environment is represented by a "time structure" which his parameters can be configurable by the user (priority, periodicity, clock associated with this structure etc...).

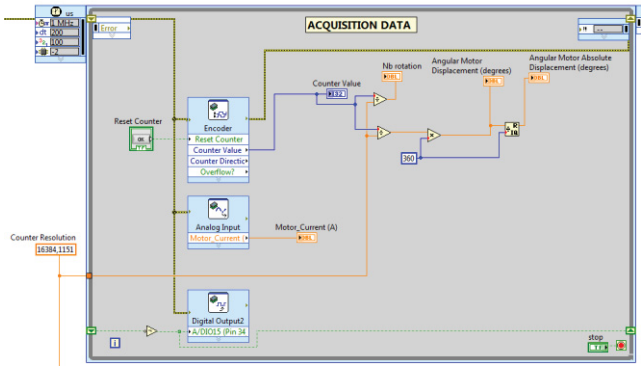


Figure 116: Block LabVIEW: Task Acquisition Data

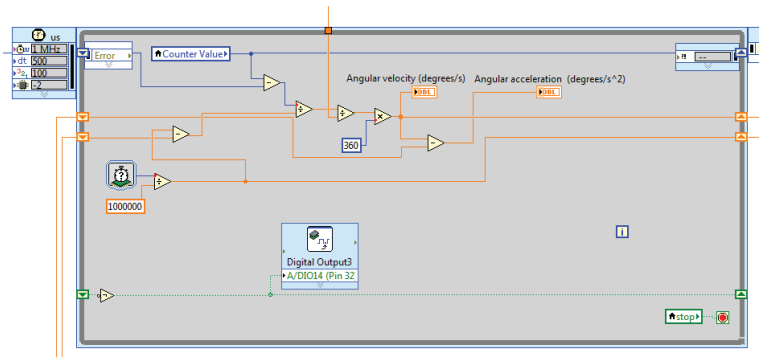


Figure 117: Block LabVIEW: Task Calculation of the speed and the acceleration

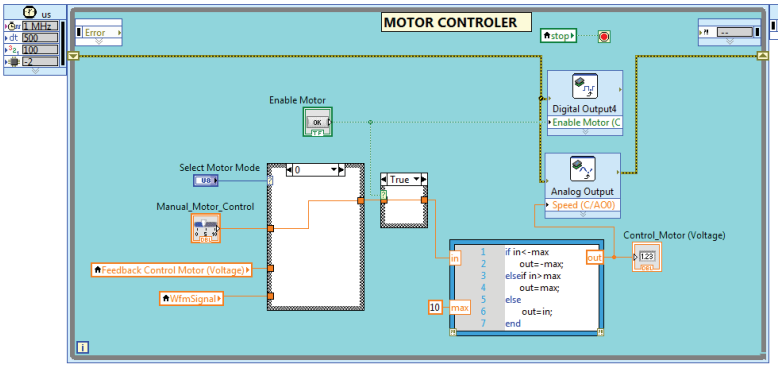


Figure 118: Block LabVIEW: Task Motor Controller

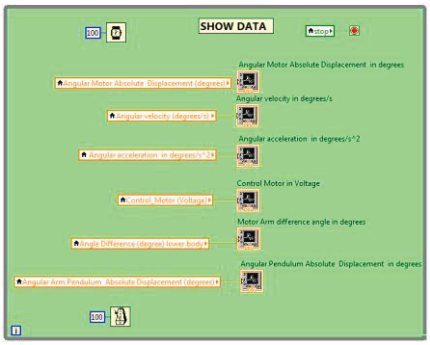


Figure 119: Block LabVIEW: Task Interface

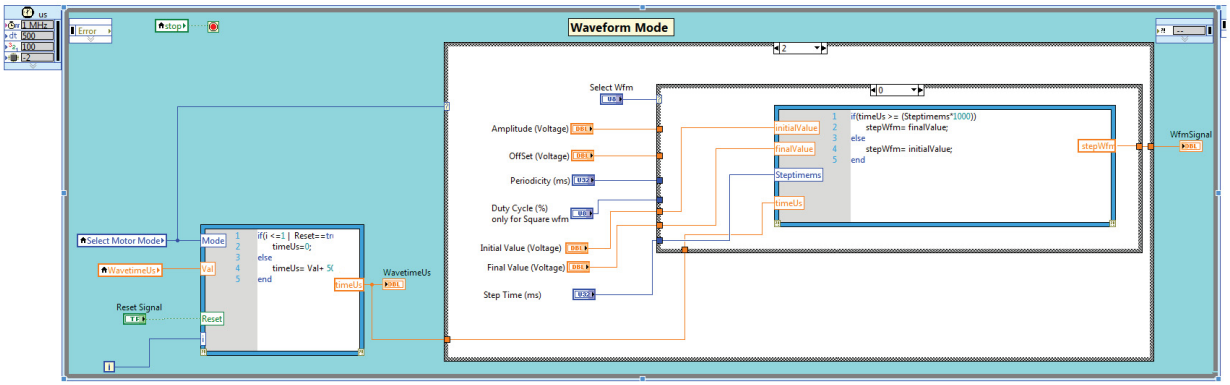


Figure 120: Block LabVIEW: Task Waveform Generator

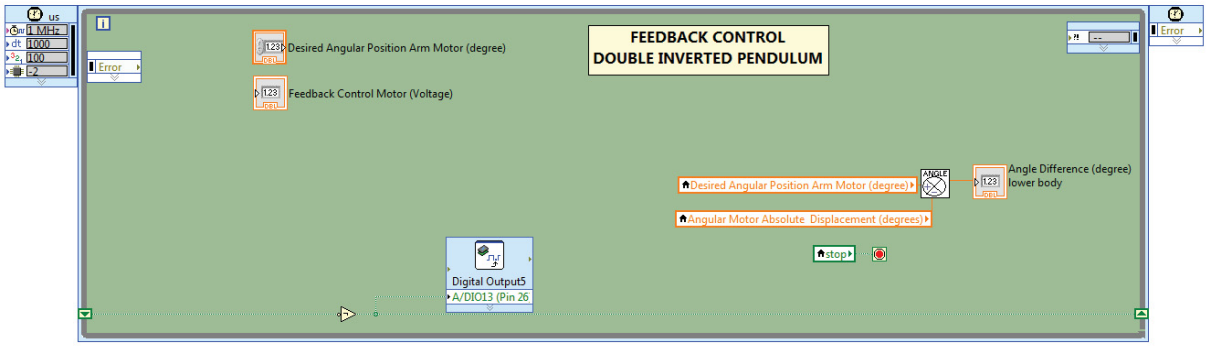


Figure 121: Block LabVIEW: Task Feedback Control (where the control system should be implemented)

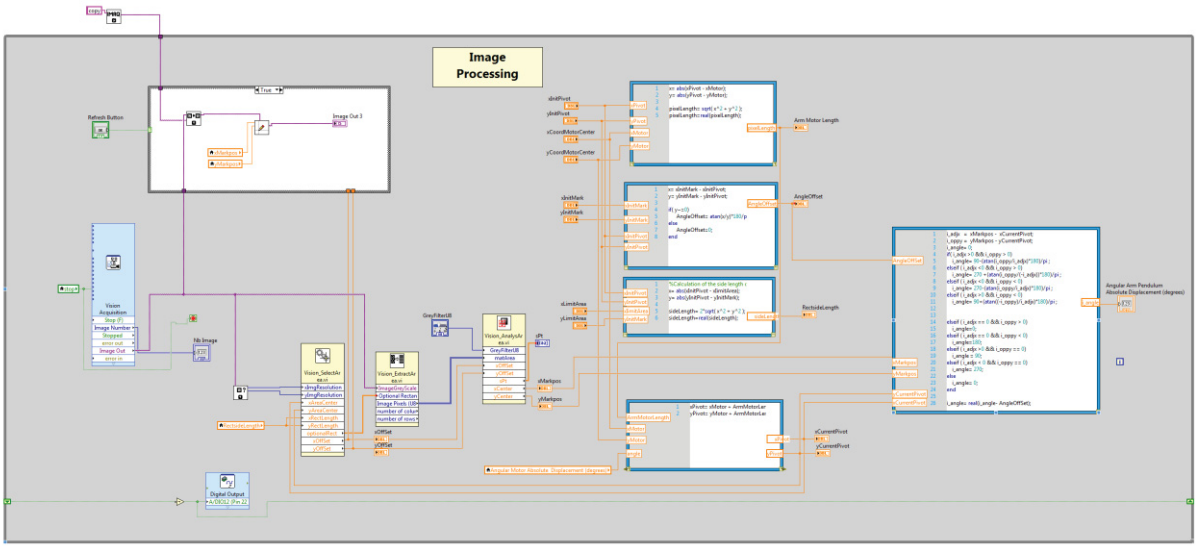
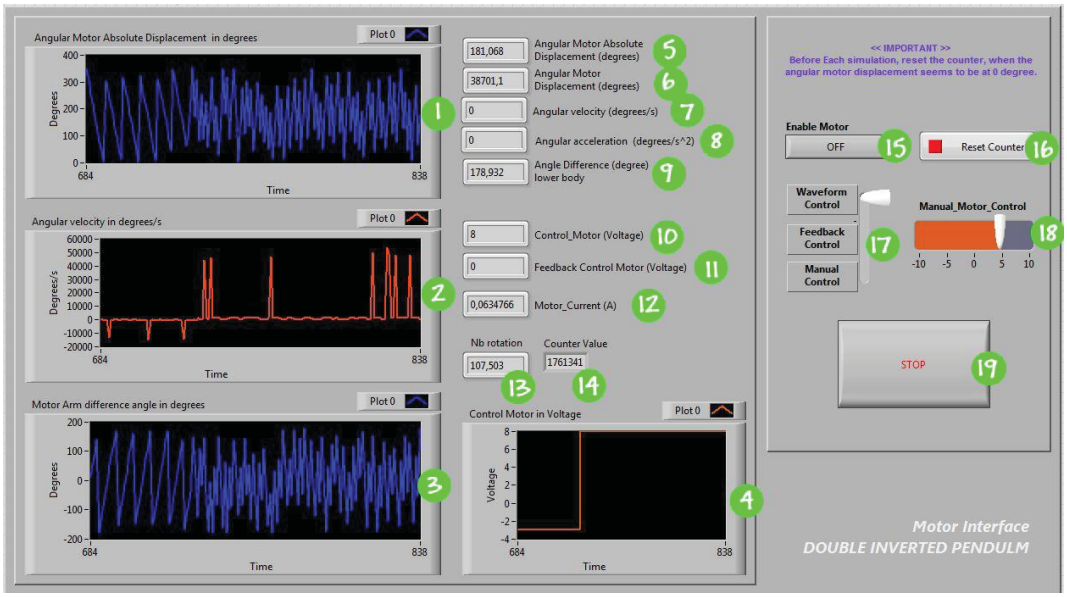


Figure 122: Block LabVIEW: Task Image Processing

5.4 LabVIEW Interface

The LabVIEW interface created for the user in order to control and parameterize the system are:



(1) First limb angular absolute position waveform (degrees)

(2) First limb angular velocity waveform (degree. s^{-1})

(3) difference first limb angular position compared to the desired value (waveform degrees)

(5) First limb angular displacement value (degrees)

(7) First limb angular velocity value (degree. s^{-1})

(8) First limb angular acceleration value (degree. s^{-1})

(11) Feedback Control Motor value (voltage)

(12) motor current (ampere)

(13) number of rotation

(16) reset the encoder counter value

(17) select the motor controller mode

(18) gear motor for the manual mode (voltage)

(4) voltage motor controller waveform (voltage)

(5) First limb angular absolute displacement value (degrees)

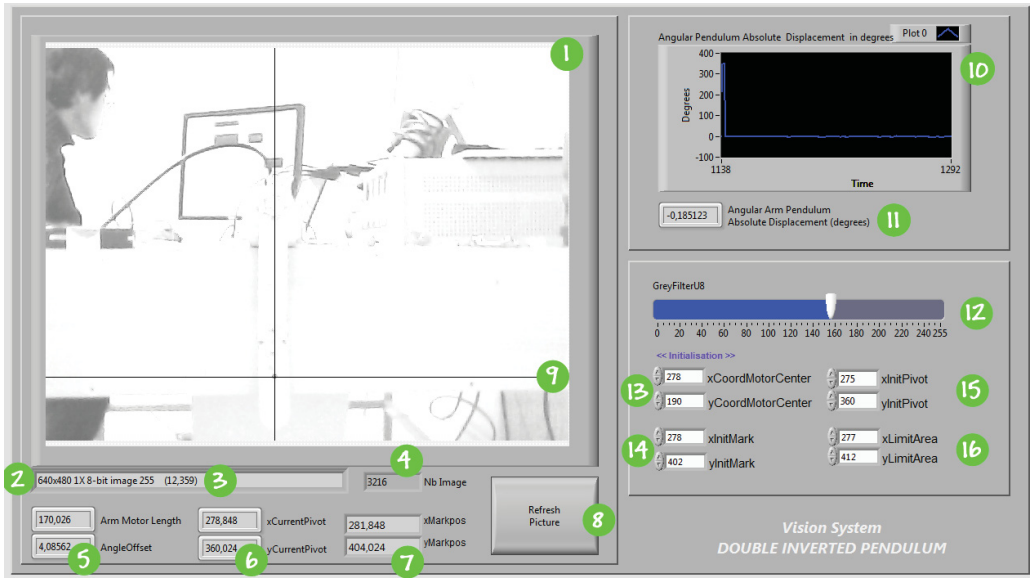
(9) difference first limb angular position compared to the desired value (degrees)

(10) voltage motor controller value (voltage)

(14) encoder counter value

(15) enable supply voltage motor

(19) stop the experimentation



(1) image acquired by the camera

(2) image size

(3) position of the mouse cursor (if it is in this area 1)

(4) number of image caught

(5) Initial angle inclination of the camera

(6) position of the pivot pendulum

(7) position of the pivot pendulum

(8) Refresh the screen picture

(9) indicator of the position of the marker

(10) Angular Second limb absolute position waveform (degree)

(11) Angular Second limb absolute position value (degree)

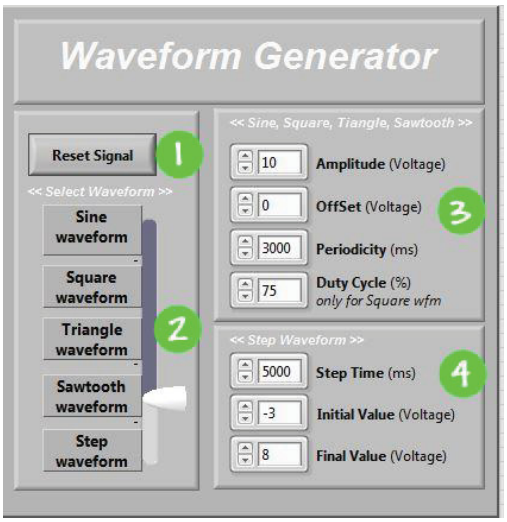
(12) Filter Greyscale value

(13) initial center position of the shaft motor (to be calibrated)

(14) initial center position of the marker (to be calibrated)

(15) initial center position of the pivot Pendulum (to be calibrated)

(16) limited Area to (to be calibrated)



(1) reset the signal

(2) choice of the waveform

(3) Parameters for sine, square, triangle and saw tooth waveform

(4) Parameters for step waveform

Conclusion

The embedded device MyRIO is adaptable for the implementation of control system and to acquire different type of signals (logical, analogic, quadrature). This device is suitable for the incremental encoder but in the other hand, MyRIO is not completely suitable to save continuously measurement data. It is necessary to manage the memory correctly (especially for saving the measurement data continuously). One solution is to define and fix the dimension of each memory block associated for each data measurement. It is especially true when we need to save an image from the camera.

The electronic scheme for the current measurement was entirely designed and almost tested (only the part "Shunt Resistor + tension Divider Bridge" was not tested). The amplifier with a galvanic isolation was tested and it is functioning. The only part that was not experimentally tested is the shunt resistor + Voltage Divider Bridge. By using this scheme, it will return to us a better result than using the" maxon servo controller", because this electrical circuit has better bandwidth and the output value signal is not quantified.

The method to get the angular position of the second limb by the Hough Transform is very accurate, but his description with LabVIEW block may take a lot of time. Because using Matlab .m file in LabVIEW is not efficient because of the compiler C from MATLAB (not optimized and take a lot of resource). The method by pattern recognition can be enough fast and suitable for our case.

Experimentations, to get the angular position of the second limb, were made with a low-cost USB webcam camera, which adds a lot of noise. It is necessary for the final experimentation, and to valid this part, to test with the chosen camera. We insist on the results of "specification of the vision system" report (look at appendix 1). According to this report the vision system needs a camera with at least 72 fps and a fine suggestion is around 144 fps. This camera must have below features.

- more than 72 fps,
- Higher shutter speed,
- To be able to do pretreatment during taking a picture (filter, edge detector etc…) in order to gain time,
- To have an usb connection (to be able to connect to MYRIO).

REFERENCES

[1] M. Stuflesser and M. Brandner, "Vision-Based Control of an Inverted Pendulum using Cascaded Particle Filters," in *IEEE Instrumentation and Measurement Technology Conference Proceedings, 2008. IMTC 2008*, 2008, pp. 2097–2102.

[2] H. Wang, A. Chamroo, C. Vasseur, and V. Koncar, "Hybrid control for vision based Cart-Inverted Pendulum system," in *American Control Conference, 2008*, 2008, pp. 3845–3850.

[3] J. Krishen and V. M. Becerra, "Efficient fuzzy control of a rotary inverted pendulum based on LQR mapping," in 2006 IEEE Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 2006, pp. 2701–2706.

[4] S. Yasunobu and M. Mori, "Swing up fuzzy controller for inverted pendulum based on a human control strategy," in *, Proceedings of the Sixth IEEE International Conference on Fuzzy Systems, 1997*, 1997, vol. 3, pp. 1621–1625 vol.3.

[5] M. Maggiore and L. Consolini, "Virtual Holonomic Constraints for Euler #x2013;Lagrange Systems," *IEEE Trans. Autom. Control*, vol. 58, no. 4, pp. 1001–1008, Apr. 2013.

[6] "Studying Kalman filtering — A laboratory experiment : A. Hultgren, pp 749–752," *Control Eng. Pract.*, vol. 3, no. 1, p. 133, Jan. 1995.

[7] M. Akhtaruzzaman and A. A. Shafie, "Modeling and control of a rotary inverted pendulum using various methods, comparative assessment and result analysis," in *2010 International Conference on Mechatronics and Automation (ICMA)*, 2010, pp. 1342–1347.

[8] "Euler–Lagrange equation," *Wikipedia, the free encyclopedia*. 09-Jan-2015.

[9] "GRIN - Mathematical Model Analysis and Control Algoritms Design based on State Feedback Method of Rotary Inverted Pendulum." [Online]. Available: http://www.grin.com/en/e-book/232209/mathematical-model-analysis-and-control-algoritms-design-based-on-state. [Accessed: 10-Jan-2015].

[10] "Gravity of Earth," *Wikipedia, the free encyclopedia*. 02-Jan-2015.

[11] "The Value of g." [Online]. Available: http://www.physicsclassroom.com/class/circles/Lesson-3/The-Value-of-g. [Accessed: 10-Jan-2015].

[12] "Permanent Magnet DC Motors." [Online]. Available: http://www.ohioelectricmotors.com/permanent-magnet-dc-motors-649#fn-649-5. [Accessed: 10-Jan-2015].

[13] "Control Tutorials for MATLAB and Simulink - Motor Speed: System Modeling." [Online]. Available: http://ctms.engin.umich.edu/CTMS/index.php?example=MotorSpeed&section=SystemModeling. [Accessed: 10-Jan-2015].

[14] S. Brock, "Identification of the parameters in inverted pendulum model [DC motor control]," in *7th International Workshop on Advanced Motion Control, 2002*, 2002, pp. 316–321.

[15] D. Park, D. Chwa, and S.-K. Hong, "An Estimation and Compensation of the Friction in an Inverted Pendulum," in *SICE-ICASE, 2006. International Joint Conference*, 2006, pp. 779–783.

[16] "NI myRIO Project Essentials Guide - National Instruments." [Online]. Available: http://www.ni.com/tutorial/14621/en/. [Accessed: 27-Jan-2015].

[17] "Waterloo Labs," *Waterloo Labs*. [Online]. Available: http://www.waterloolabs.com/. [Accessed: 27-Jan-2015].

[18] "A review on Hough transform", *Henri MAITRE,* [Online]: Available: http://documents.irevues.inist.fr/bitstream/handle/2042/2334/02.PDF+TEXTE.pdf?sequence=3

TABLE OF FIGURES

Figure 1: picture of the double inverted pendulum 8
Figure 2 : Our real double pendulum connected to a DC motor shaft..... 8
Figure 3 : the shunt regulator DSR 70/30, Taken from a web page 9
Figure 4: ESCON 50/5 Servo Controller, taken from a web page 9
Figure 5: Minimal External Wiring from the Company datasheet 10
Figure 6: Wiring DC motor to the ESCON 50/5, Company datasheet 11
Figure 7: DC Motor equivalent circuit model, where U: the terminal voltage of the DC motor, I: the armature current, L: the armature inductance, r: the armature resistance, E: the back efm).....12
Figure 8: PWM waveform scheme principle. Where Vmoy: average voltage, the: time high state, T: Time periodicity, Vcc: Maximum voltage.....13
Figure 9: Management of the Supply voltage for the motor on LabVIEW (from the LabVIEW Program of our project).....13
Figure 10: Part of the LabVIEW interface (from the LabVIEW Program of our project) .14
Figure 11: part of the LabVIEW Interface (from the LabVIEW Program of our project) .14
Figure 12: Part of the LabVIEW interface (from the LabVIEW Program of our project) .14
Figure 13: Waveform generator interface (from the LabVIEW Program of our project) ..15
Figure 14: Sine waveform generated (amplitude: 5V, offset: 3V, periodicity: 3000 ms) (from the LabVIEW Program of our project) 15
Figure 15: Square waveform generated: (amplitude: 5V, offset: -2V, periodicity: 3000 ms, duty cycle: 75%) (from the LabVIEW Program of our project) 15
Figure 16: Step waveform generated (Step Time: 5s, initial value: -3V, final value: 8V (from the LabVIEW Program of our project) 16
Figure 17: Saw tooth waveform generated: (amplitude: 5V, offset: 0V, periodicity: 3000 ms) (from the LabVIEW Program of our project).....16
Figure 18: Hardware connections.....17
Figure 19: Encode SCH50F, Taken from a web page 18
Figure 20: Scheme of the functioning of the encoder 18
Figure 21: Scheme of the functioning of the encoder 18
Figure 22: Scheme of the functioning of the output signals from the encoder.....19
Figure 23: Encoder LabVIEW Block (from the LabVIEW Program of our project) 19
Figure 24: Configuration of the encoder block (from the LabVIEW Program of our project) 20
Figure 25: Block acquisition data responsible to refresh the angle position value from the encoder (from the LabVIEW Program of our project) 20
Figure 26: General view of the processing chain for the current measurement in the motor 21
Figure 27; First part of the electrical scheme for the current motor measurement 21
Figure28: Second part of the electrical circuit scheme: amplifier with galvanic isolation22
Figure 29: How to place the shunt resistor 22
Figure 30 : First part of the electrical scheme for the current motor measurement 23
Figure 31: PSpice Simulation V2= Vout = V1/100 =Vshunt/100 23
Figure 32: Taken from a web page 23
Figure 33: scheme from the IL300 datasheet.....24

Figure 34: study scheme for the IL300 25
Figure 35: simulation result from PSPICE software 26
Figure 36: Final schema for the galvanic isolation 27
Figure 37: Offset and attenuation stage 27
Figure 38: Threshold figure from the TLC2272 datasheet 27
Figure 39: scheme of the simulation in PSPICE 29
Figure 40: simulation result from PSPICE 29
Figure 41: servo and amplifier current electrical scheme 30
Figure 42: current ***If*** result with an input signal (offset +1V and an alternative part equal to 0.25V) 30
Figure 43: current ***If*** result with an input signal (offset +0,75V and an alternative part equal to 0.25V) 31
Figure 44: current ***If*** result with an input signal (offset +0,6V and an alternative part equal to 0.1V) 31
Figure 45: Output Stage 32
Figure 46: PSPICE simulation result (test by using two different grounds references) 33
Figure 47: PSPICE result Bod diagram 33
Figure 48: PSPICE result *time domain: instable signal output* 33
Figure 49: suppression of the instability by adding C2, R8, and C4 components (filter function) 33
Figure 50: Gain and stability study after adding the capacitor C4 34
Figure 51: Gain and stability study after adding the capacitor and the resistor C2 and R8 34
Figure 52: PSPICE result: THD Analysis for 1 KHZ and 10 KHZ 35
Figure 53: Straight line in Cartesian axes 38
Figure 54: Straight line in Cartesian axes 38
Figure 55: Straight line transposed to the Hough Plane 38
Figure 56: inverted pendulum with red and blue marker added with Paint: picture jpg (400x300) 39
Figure 57: Hough processing chain 39
Figure 58: Hough processing chain 39
Figure 59: inverted pendulum with red and blue marker added with Paint: picture jpg (400x300) 40
Figure 60: result after the pretreatment (color filter + greyscale conversion + edge detector) (from our Matlab project) 40
Figure 61: Accumulator result from the Hough Transform (we only keep the maximal value of the accumulator for each angle). The Hough operation returns us an angle result about the angular position of the second limb equal to -76,5° (with 0,5° degree of accuracy). By knowing the position of the red and the blue marker, we can calculate the true angle value: 76,5 degree (with 0,5 degree of accuracy) 40
Figure 62: Accumulator result from the Hough Transform "1 to M" 41
Figure 63: Hough processing chain for LabVIEW implementation 42
Figure 64 : Test of the implementation of the Hough transform 43
Figure 65: Accuracy for the Hough Transform for each true angle value that we should found (resolution choose for he Hough operation: 0,1 degree). The average error value for each angle value is 0.0303 degree. The maximal error value is 0,4 degree 43

Figure 66: Accuracy for the Hough Transform for each true angle value that we should found (resolution choose for the Hough operation: 0,5 degree). The average error value for each angle value is 0.1214 degree. The maximal error value is 0,5 degree.....44
Figure 67: double inverted pendulum 45
Figure 68: general operating processing chain to get the angle position with the marker detection.....45
Figure 69: calculation of the pivot pendulum position (showed by a cross). The limited region around the pivot pendulum is represented by the square. The picture is obtained by a low cost usb camera for webcam application.....46
Figure 70: Detection of the black mark center in the analyzing region. The picture is obtained by a low cost usb camera for webcam application.....46
Figure 71: Vision System interface in LabVIEW.....47
Figure 72: camera usb 38 4754, taken from the company web site.....47
Figure 73 : processing times for the vision system with marker detection.....47
Figure 74: A way to measure the time needed for each part.....49
Figure 75: The strategy for finding the angle of the second limb.....51
Figure 76: The Simulink figure of the vision system.....52
Figure 77: The output of MATLAB function in Figure 76 which shows the detected circles, drawn lines and the angle between them. 53
Figure 78: The results of the encoder and the vision system for an experiment 54
Figure 79: the difference between the results of the encoder and the vision system.....55
Figure 80: The neglected part of the pendulum for modeling 58
Figure 81: A rigid rod with uniform mass distribution hangs from a pivot point.....58
Figure 82: Moment of inertia in a physical pendulum.....59
Figure 83: Moment of inertia in a flat plate.....60
Figure 84: value of g if we become far away from the Earth's center [11] 61
Figure 85: Karlskrona city in Sweden, the location of the laboratory [11] 62
Figure 86: Results from the gravity calculator based on the Karlskrona city [11] 63
Figure 87: Sketch of double pendulum.....64
Figure 88: Degree of freedom in double pendulum 64
Figure 89: Degree of freedom in double pendulum [13].....68
Figure 90: Sketch of a simple pendulum.....71
Figure 91: The difference between the results of 5 experiments with physical pendulum 72
Figure 92: Simulink file of equation (29), a pendulum with 2 types of friction 73
Figure 93: The results of Simulation and Encoder for the first experiment 74
Figure 94: A scene from the recorded video for the second experiment 75
Figure 95: Simulation results of the second experiment 76
Figure 96: The nonlinear model in Simulink/MATLAB 77
Figure 97: Inside of the subsystem-Friction1 in Figure 96.....78
Figure 98: Inside of the subsystem-Friction2 in Figure 96.....78
Figure 99: Simulation results of nonlinear model without input and 0 initial conditions79
Figure 100: The result of nonlinear model without input and with initial condition (-168°,0, -168°,0,0) (X-axis is second, Y-axis is degree).....80
Figure 101: The effect of different values for Electric Resistance 81

Figure 102: The angular position of the first limb, $\boldsymbol{\theta 1}$ from the encoder and simulation for initial condition (-168°,0, -168°,0,0) 82
Figure 103: The difference between the encoder and the simulation results 83
Figure 104: The Signum function 84
Figure 105: The arctan and sign function 85
Figure 106: The plot of arctan(x) and sin(x) 85
Figure 107: Feedback Controller 89
Figure 108: controller bring the pendulum to the down position with the designed controller signal (X-axis is second, Y-axis is angular position in degree) 90
Figure 109: controller bring the pendulum to the up position with the designed controller signal (X-axis is second, Y-axis is angular position in degree) 90
Figure 110: Control design interface of the controller in LabVIEW 91
Figure 111: Simulation interface of the controller in LabVIEW 92
Figure 112: blocks of Lab View's interface for controller part 92
Figure 113: blocks of Lab View's interface for controller part 93
Figure 114: Hardware block diagram from the NI MyRIO datasheet 94
Figure 115: General view of the LabVIEW code 96
Figure 116: Block LabVIEW: Task Acquisition Data 96
Figure 117: Block LabVIEW: Task Calculation of the speed and the acceleration 96
Figure 118: Block LabVIEW: Task Motor Controller 97
Figure 119: Block LabVIEW: Task Interface 97
Figure 120: Block LabVIEW: Task Waveform Generator 97
Figure 121: Block LabVIEW: Task Feedback Control (where the control system should be implemented) 97
Figure 122: Block LabVIEW: Task Image Processing 98

Appendix 1 - Specifications of the camera

In this appendix the specification of the camera for the vision system discussed. We model the system and according to the information of the model and based on the different controller we tried to find the specification of a suitable camera for the vision system.

This appendix covered 3 important factors of a camera, frame rate, shutter speed and resolution.

1- Frame rate

Introduction

By considering the linearized models of the system around the two of important equilibrium points, down (both pendulums in down position) and up (both pendulums in up position) and by assuming zero friction and also by using the Ackermann`s formula (Pole placement gain selection), and this feedback law:

$u = -Kx$

the results from Simulink of MATLAB (Figure 1) show the behavior of the system in 4 different cases.

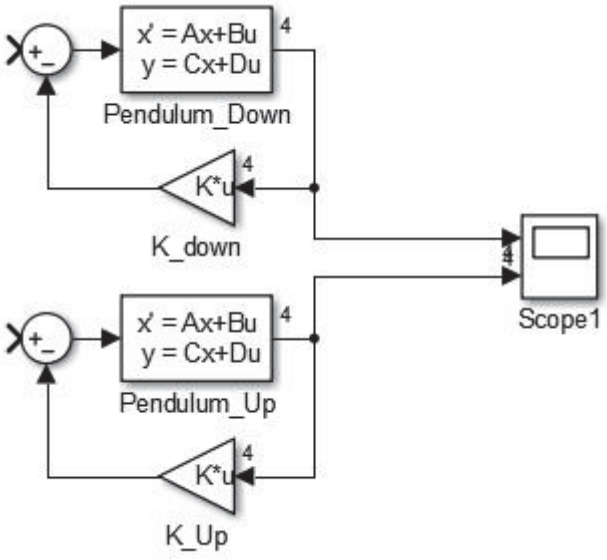


Figure 1. The schematic of the Simulink file

Case 1: Eigenvalue -3

Initial conditions: [q1 , q1_dot , q2 , q2_dot] = [35 0 35 0]

Eigenvalues: [-3 -3 -3 -3]

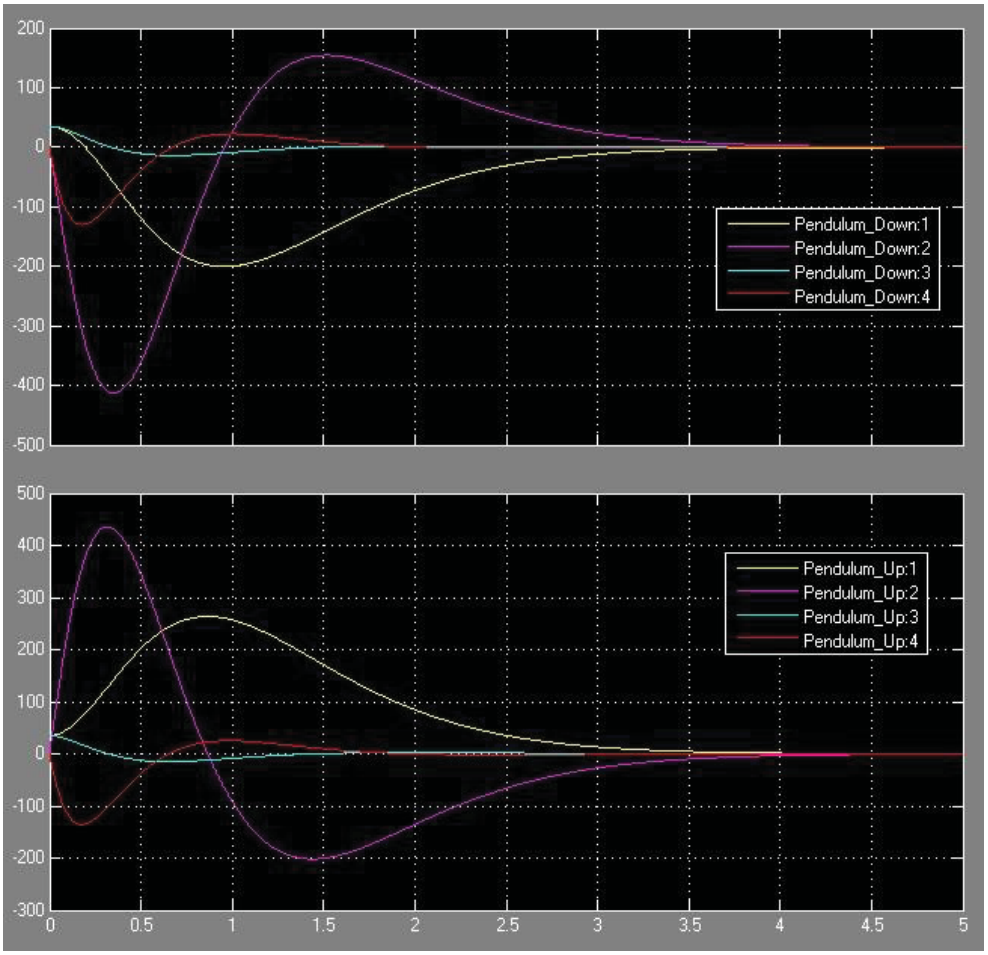


Figure 2. X Axis: Time (in second) / Y Axis: Position of bars (in degree)

Yellow line: position (q1), Magenta line: velocity (q1_dot)

cyan line: position (q2), Red line: velocity (q2_dot)

According to Figure 2, the controller takes around 4.5 seconds to returning the pendulum from Initial conditions, +35° to 0°. For us the position of q2 is important, the cyan color. In this case, the quickest movement of the pendulum is 50° from +35 to -15 in around 0.7 seconds which it means 72° in 1 second.

Case 2: Eigenvalue -6

Initial conditions: [q1 , q1_dot , q2 , q2_dot] = [35 0 35 0]

Eigenvalues : [-6 -6 -6 -6]

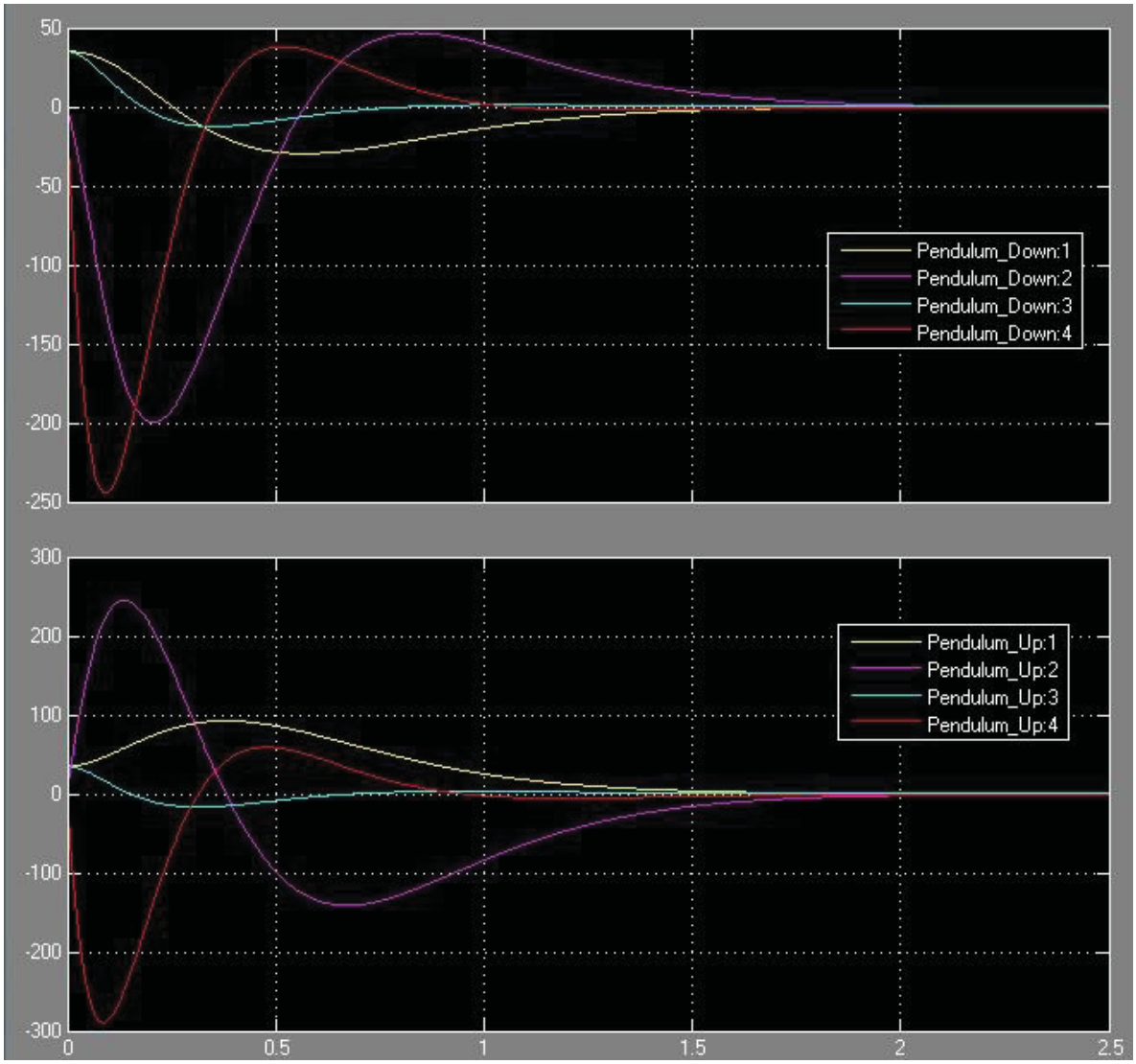


Figure 3. X Axis: Time (in second) / Y Axis: Position of bars (in degree)

Yellow line: position (q1), Magenta line: velocity (q1_dot)

cyan line: position (q2), Red line: velocity (q2_dot)

In this case (Figure 3), there are 50° changes in 0.3 seconds which it means 167° in 1 second.

Case 3: Eigenvalue -9

Initial conditions: [q1 , q1_dot , q2 , q2_dot] = [35 0 35 0]

Eigenvalues : [-9 -9 -9 -9]

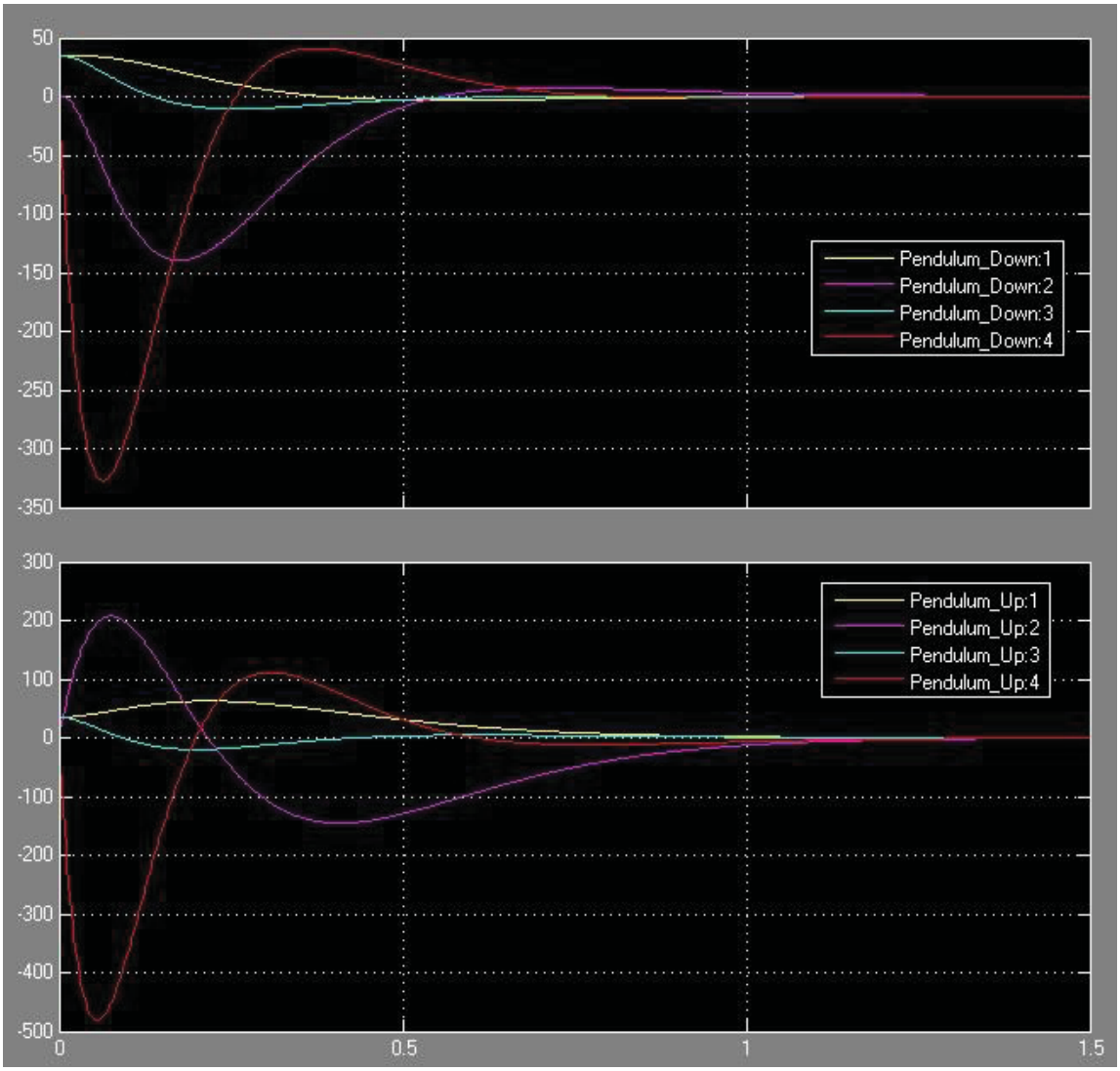


Figure 4. X Axis: Time (in second) / Y Axis: Position of bars (in degree)

Yellow line: position (q1), Magenta line: velocity (q1_dot)

cyan line: position (q2), Red line: velocity (q2_dot)

In the third case (Figure 4), it can be seen, 55° changes in 0.2 seconds which it means 275° in 1 second.

Case 4: Eigenvalue -15

Initial conditions: [q1 , q1_dot , q2 , q2_dot] = [35 0 35 0]

Eigenvalues : [-15 -15 -15 -15]

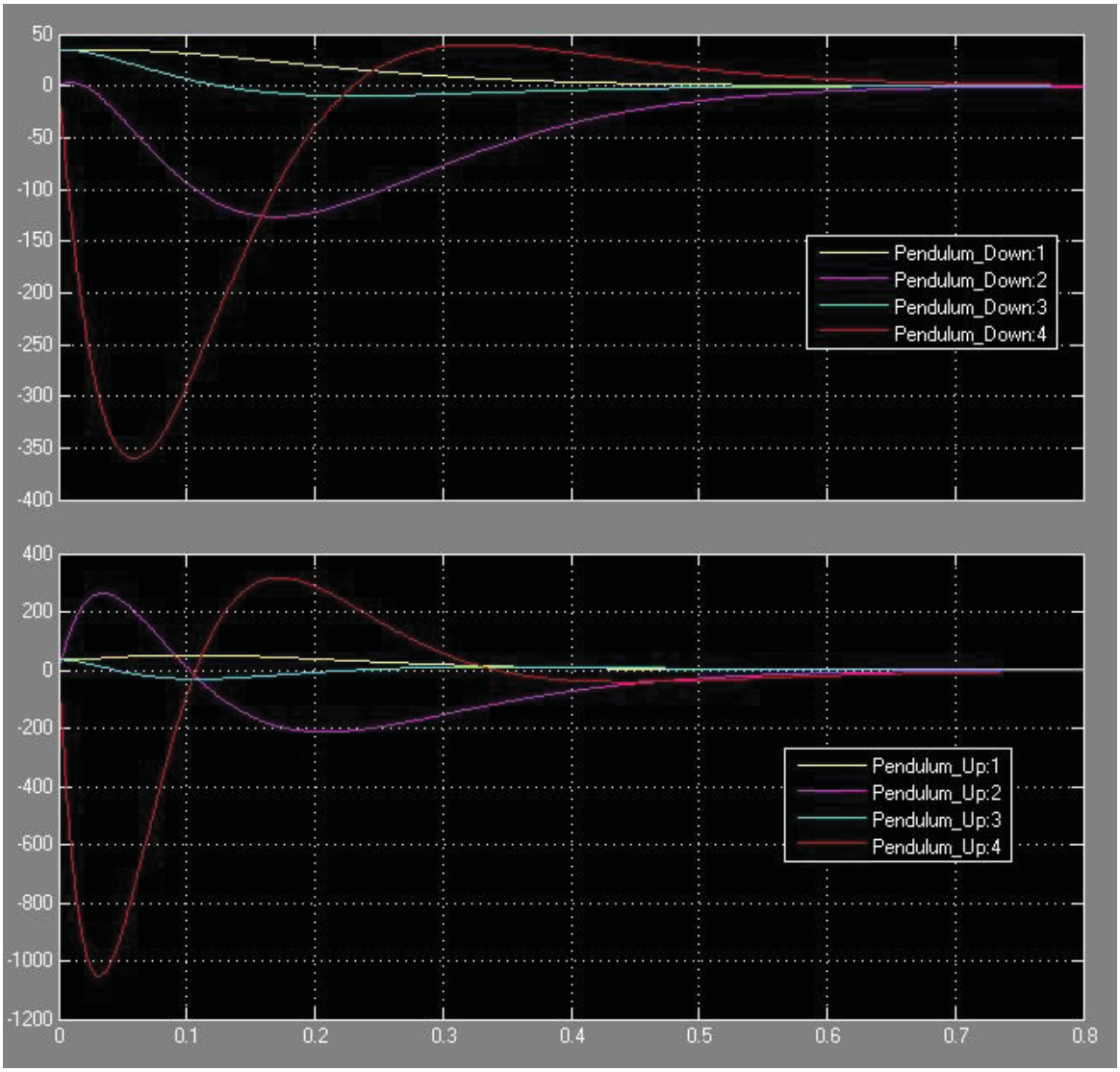


Figure 5. X Axis: Time (in second) / Y Axis: Position of bars (in degree)

Yellow line: position (q1), Magenta line: velocity (q1_dot)

cyan line: position (q2), Red line: velocity (q2_dot)

And finally for the last case (Figure 5), 65° changes in 0.1 second which it means 650° in 1 second.

Now I should come to this question, which sensitivity we need? I mean, sensing 1 degree is enough precise? Or it should be less than that? According to [1] a similar successful work was done with a sensitivity of 0.25 degrees.

Conclusion

Then we summarize the results in Table 1.

Table 1. Summarizing of the results for four cases

sensitivity	1°	0.5°	0.25°	0.1°
Case 1- Eigen -3	72 fps	144 fps	288 fps	720 fps
Case 2- Eigen -6	167 fps	334 fps	668 fps	1670 fps
Case 3- Eigen -9	275 fps	550 fps	1100 fps	2750 fps
Case 4- Eigen -15	650 fps	1300 fps	2600 fps	6500 fps

In the topic of frame rate, maybe I can come to this conclusion which a camera with a frame rate between 600 to 1000 fps is suitable for our work but it seems a one with1000 fps is quite enough good for our work.

According to [2] a normal motion picture is filmed and played back at 24 frames per second, while television uses 25 frames/s (PAL) or 29.97 frames/s (NTSC). This is also based on brain`s human ability that can process only 25 frames/s [1].

Also the maximum available frame rate is 1,400,000 fps, for Phantom v711 camera.

2- Shutter speed

Shutter speed is a key factor for reducing the error during the angle calculation process. Slow shutter speed can achieve a motion blur for moving objects. In this case and especially in high speed movement of the pendulum the error will increase dramatically.

In below the effect of shutter speed are shown in 2 different speeds.

*Fast shutter speed**

*Slow shutter speed**

**the images are token from Nikon website.*

Nowadays in most of the new digital cameras the shutter speed is around 1/2000 seconds, it means the shutter will be open just for 1/2000 = 0.0005 seconds that is enough good. The conclusion of this topic is clear, a higher shutter speed leads to a lower error.

3- Resolution

To know the necessary resolution to calculate the angular position of the second limb, we have to calculate the resolution needed of this part.

If we consider that one pixel difference is enough to get a X precision (like 0,5 degree precision), we need to know how many pixel should describe the length of the pendulum arm. And knowing that the camera will be fixed (centered on the motor axis), and get a picture that can catch all the position of the second limb, we can know the strict minimum resolution needed.

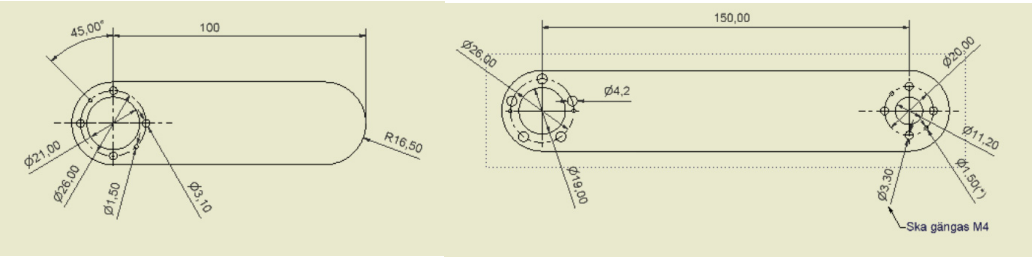


Figure 6. Pendulum arm and Motor arm dimension in mm

By using the dimension of each arms of the double inverted pendulum, we have this equation that described the resolution needed in function of the precision we fixed:

$$\text{Resolution} = 2 * \frac{(L1 + L2)}{L2} * \frac{1}{\tan(X)}$$

L1= length of the motor arm (150mm)

L2= length of the Pendulum arm (100mm)

X= precision needed (like 0,25 degree precision)

Table 2. Summarizing of the results

degree precision	Minimum pixel resolution
0,1	2865x2865
0,25	1145x1145
0,5	572x572
1,0	287x287

References:

[1] M. E. Magana and F. Holzapfel, "Fuzzy-logic control of an inverted pendulum with vision feedback," *IEEE Trans. Educ.*, vol. 41, no. 2, pp. 165–170, May 1998.
[2] "High-speed camera," *Wikipedia, the free encyclopedia*. 27-Oct-2014.

Appendix 2 – MATLAB Codes for Hough Transformation

```
%% MAIN.m FILE
% load a picture
p_file= '..\..\picture\raw13.jpg';
mat_raw_img= imread(p_file);
vect_img= imgMat2Vect(mat_raw_img);
i_sizeimg=size(mat_raw_img);
i_yimg= i_sizeimg(1);
i_ximg= i_sizeimg(2);

%% Hough Pretreatment
[mat_pretreatment_img,
sRedpart,sBluepart,matEdgePt]=
HoughPretreatment(vect_img,i_ximg,i_yimg,
1,i_yimg, 1, i_ximg);

%% Hough Transform
[matHough, vectTheta] =
HoughAnalysOne2M(matEdgePt,1,
length(matEdgePt), 0.1, 0.5);

%% Hough Accumulator
vectAccu = HoughAccumulator( matHough, vectTheta,
1, length(matHough));

%% Find Hough Peak
i_HoughAngle = HoughPeak(vectAccu,vectTheta);

%% Calcul Blue part position & the Red part position
sRedpart(1)= sRedpart(1)/sRedpart(3);
sRedpart(2)= sRedpart(2)/sRedpart(3);
sBluepart(1)= sBluepart(1)/sBluepart(3);
sBluepart(2)= sBluepart(2)/sBluepart(3);

%% Calcul the real angle
i_realAngle =
angleHough2Real(i_HoughAngle,sRedpart,sBluepart);
```

```
%% coordMat2Vect.m FILE
%input:
   % i_ximg: size x length of the analyzing picture
   % ix: position coordinate x
   % iy: position coordinate y
%output
   % i_rng: position in the vector

function i_rng = coordMat2Vect(ix, iy, i_ximg)
   i_rng= (iy-1)*(i_ximg*3)+(ix-1)*3;
end
```

```
%% imgMat2Vect.m FILE

% In labview when we extract the pixel data from a
jpeg file, we get a vector instead a matrice in 2
dimensions like in Matlab. This function is just to
convert the picture data from matlab to labview

function vect_img= imgMat2Vect(mat_img)
   i_size=size(mat_img); %getting the size of the
picture
   vect_img= uint8(zeros(1, i_size(1)*i_size(2)*3));

   for(iy=1:i_size(1))
      for(ix=1:i_size(2))
         rng= (iy-1)*(i_size(2)*3)+(ix-1)*3;
         vect_img(rng+1) = mat_img(iy,ix,1);
         vect_img(rng+2) = mat_img(iy,ix,2);
         vect_img(rng+3) = mat_img(iy,ix,3);
      end
   end
end
```

```
%% angleHough2Real.m FILE

%Calculation of the real angle from the hough angle
found
function i_realAngle =
angleHough2Real(i_houghAngle,sRedpart,sBluepart)
   if(i_houghAngle<=0)
      i_realAngle= abs(i_houghAngle);
   else
      i_realAngle=180-i_houghAngle;
   end
   % blue position added
   if( i_realAngle < 45) % x < 45
      if ((sBluepart(2)) > (sRedpart(2)))
         i_realAngle = i_realAngle+180;
      end
   elseif( i_realAngle <135 ) % 45 <= x < 135
      if ((sBluepart(1)) > (sRedpart(1)) )
         i_realAngle=i_realAngle+180;
      end
   elseif( i_realAngle <180 ) % 135 <= x <180
      if ((sBluepart(2))  < (sRedpart(2)))
         i_realAngle=i_realAngle+180;
      end
   end
end
```

```
%% HoughAccumulator.m FILE
function vectAccu = HoughAccumulator( matHough, vectTheta, i_start, i_end)
   vectAccu=zeros(1, length(vectTheta));
   i_size= size(matHough);
   i_length=i_size(2);
   i_nbElmt= i_size(1);

   if(i_end> i_length)
      i_end = i_length;
   end
   matTemp=sort(matHough(:, i_start:i_end),1);
   for i=i_start:i_end
      counterMax=0;
      currentCounter=0;
      currentValue=  matTemp(1,i-i_start+1);  %new value

      for j=1: i_nbElmt
         if( currentValue == matTemp(j,i-i_start+1))
            currentCounter=currentCounter+1; %increment the
                                             current counter
         else
            % compare the actual counter with the max counter
            if(currentCounter> counterMax)
               counterMax= currentCounter;
            end
            currentCounter= 1;
            currentValue  =  matTemp(j,i-i_start+1); %new value
         end
         %if it s the last element of the column
         if( j == i_nbElmt)
            % compare the actual counter with the max counter
            if(currentCounter> counterMax)
               counterMax= currentCounter;
            end
         end
      end
      %save the max value for this angle
      vectAccu(i)= counterMax;
   end
end
```

```
%% HoughPeak.m FILE

function i_angle = HoughPeak(vectAccu,vectTheta)
   [A B]= max(vectAccu);
   i_angle=vectTheta(B);
end
```

```
% Function adapted for the labview environment
% Read color filter
% get the blue and read position
% convert to binary conversion

function [mat_rgb2grey_filter_redblue,sRedpart,sBluepart, matEdgePt] = ...
   HoughPretreatment(vect_imgRGB, ...
                     i_ximg, ...
                     i_yimg, ...
                     i_linestart, ...
                     i_lineend, ...
                     i_columnstart, ...
                     i_columnend )
   %% Variables and initializing
   i_value=true;
   sBluepart=[0 0 0];  %sBluepart=struct('i_xavrg','i_yavrg','i_nbpt');
   sRedpart= [0 0 0];  %sRedpart =struct('i_xavrg','i_yavrg','i_nbpt');
   mat_rgb2grey_filter_redblue= false(i_yimg, i_ximg);
   matEdgePt=[]; %matrice that contain the position for each edge pt of the picture

   %% Color Filter
   i_maxred_threshold  = 40;
   i_maxgreen_threshold= 40;
   i_maxblue_threshold = 40;
   i_minred_threshold  = 180;
   i_minblue_threshold = 180;
   %%
   for iy=i_linestart: i_lineend  %vertically direction y

      for ix=i_columnstart : i_columnend %horizontally direction x
          rng= coordMat2Vect(ix, iy, i_ximg); %calculation of the range of the pixel
in the vect_imgRGB

          if(vect_imgRGB(rng +1 )< i_maxred_threshold &&   ... % blue filter
            vect_imgRGB(rng +2 )< i_maxgreen_threshold && ...
            vect_imgRGB(rng +3 )> i_minblue_threshold)

             %finding sBluepart avrg
             sBluepart(1)=sBluepart(1)+ix; %add the x value position of the blue pt
             sBluepart(2)=sBluepart(2)+iy; %add the y value position of the blue pt
             sBluepart(3)=sBluepart(3)+1;  %increment the number of blue pt

          elseif( vect_imgRGB(rng +1 )> i_minred_threshold &&   ... % red filter (1)
                vect_imgRGB(rng +2 )< i_maxgreen_threshold && ...
                vect_imgRGB(rng +3 )< i_maxblue_threshold)

             %Extract Edge (should be done here manually to increase the accuracy
of the pretreatment because a standard
              Extract edge algorithm like the canny algo, take a lot of time
             if(iy==1 | iy== i_yimg | ix==1 | ix== i_ximg)
                mat_rgb2grey_filter_redblue(iy,ix)=i_value;
             else
                %Edge detector
                rng1= coordMat2Vect(ix, iy-1, i_ximg);  % y-1,x
                rng2= coordMat2Vect(ix, iy+1, i_ximg);  % y+1,x
```

```
                rng3= coordMat2Vect(ix-1, iy, i_ximg);  %  y ,x-1
                rng4= coordMat2Vect(ix+1, iy, i_ximg);  %  y ,x+1

                if(~((detectRedPixel(vect_imgRGB(rng1+1:rng1+3)
,i_minred_threshold, i_maxgreen_threshold,
                     i_maxblue_threshold)==1) && ...
                    (detectRedPixel(vect_imgRGB(rng2+1:rng2+3)
,i_minred_threshold, i_maxgreen_threshold,
                     i_maxblue_threshold)==1) &&...
                    (detectRedPixel(vect_imgRGB(rng3+1:rng3+3)
,i_minred_threshold, i_maxgreen_threshold,
                     i_maxblue_threshold)==1) && ...
                    (detectRedPixel(vect_imgRGB(rng4+1:rng4+3)
,i_minred_threshold, i_maxgreen_threshold,
                     i_maxblue_threshold)==1)))
                   mat_rgb2grey_filter_redblue(iy,ix)=i_value;
                   matEdgePt=[matEdgePt [ix;iy]]; %saving the pt localization for the
hough analysis
                end
             end
             %finding sRedpart
             sRedpart(1)=sRedpart(1)+ix; %add the x value position of the red pt
             sRedpart(2)=sRedpart(2)+iy; %add the y value position of the red pt
             sRedpart(3)=sRedpart(3)+1; %increment the number of red pt
          end
       end
    end
end
```

```
%% HoughAnalysM2one..m FILE
function                    [matHough,                    vectTheta]                    =
HoughAnalysOne2M(matEdgePt,i_ptStart,i_ptEnd, i_thetastep, i_rhostep)
    vectTheta=[];
    matHough=[];

    if(~isempty(matEdgePt))
        vectTheta=[-90:i_thetastep:90];
        if(i_ptEnd> length(matEdgePt))
            i_ptEnd=length(matEdgePt);
        end
        matHough=zeros(i_ptEnd-i_ptStart+1    , length(vectTheta) );
        %Calcul each pt to the Hough plan
        for i=i_ptStart:i_ptEnd
          matHough(i-i_ptStart+1,:)=                                                  round((
matEdgePt(1,i)*cos(vectTheta.*pi/180) + ...
                                  matEdgePt(2,i)*sin(vectTheta.*pi/180))/i_rhostep  );
            end
    end
end
```