

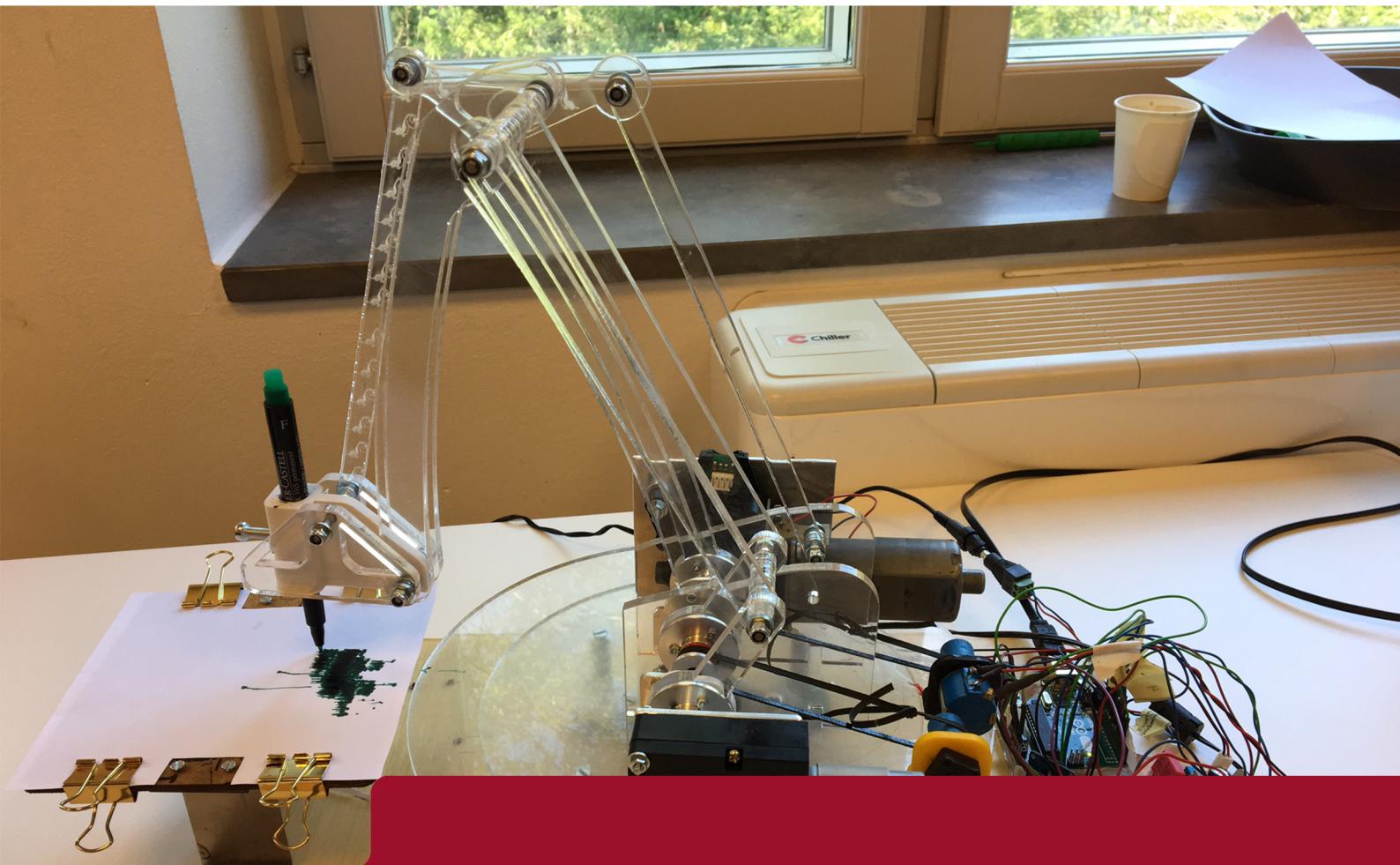


DEGREE PROJECT IN TECHNOLOGY,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2017

Drawing robotic arm

FELICIA GIHL VIEIDER

MARIA MARKOVSKA



KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF INDUSTRIAL ENGINEERING AND MANAGEMENT



Drawing robotic arm

MARIA MARKOVSKA
FELICIA GIHL VIEIDER

Bachelor's Thesis in Mechatronics
Supervisor: Fariba Rahimi
Examiner: Nihad Subasic

TRITA MMK 2017:14 MDAB 632

Abstract

A drawing robot requires precise steering and high accuracy. The purpose with this project is to design a robotic arm with 3-dof that can be fed with a gray scale picture and draw it on a paper. This thesis investigates which drawing technique that fits better for the robot, which controller is the most suitable and what accuracy is achieved with the chosen controller. During the project a drawing robot is created and two different controllers are implemented and tested PD and PID. The implementation shows that PD control is not working for this application. Therefore, PID control is the most suitable. Experiments are set up to measure the drawing accuracy using the PID-controller. A drawing accuracy of $\pm 11mm$ is achieved by the robot. The achieved accuracy is a result depending on several factors, for instance the resolution of the potentiometers.

Referat

Ritande robotarm

En ritande robot kräver en precis styrning och hög noggrannhet. Syftet med detta projekt är att skapa en robot arm med tre frihetsgrader som kan matas med en svartvit bild och rita upp den på ett papper. Projektet undersöker vilken ritteknik som är mest passande för ändamålet, vilken regulator som passar bäst samt vilken noggrannhet som uppnås för den ritade bilden. Under projektets gång skapas en ritande robot och två olika regulatorer testas och implementeras, PD- och PID-regulatorn. Implementeringen visar att en PD-regulator inte fungerar för denna applikation. PID-regulatorn är därför den lämpligaste. Experiment utförs med PID-regulatorn implementerad för att mäta den noggrannhet som roboten ritar med. Roboten uppnår en noggrannhet på $\pm 11\text{mm}$. Den uppnådda noggrannheten är ett resultat beroende av flera faktorer, bland annat upplösningen på potentiometrarna.

Acknowledgements

First of all we would like to thank our course responsible Nihad Subasic for giving us an introduction to the area of mechatronics and the tools and knowledge needed to make this project happen. We would also like to thank our supervisor Fariba Rahimi for guiding us through the project and giving us valuable input along the way. Furthermore we want to thank our teacher assistants Fredrika, Simon, Philip and Marcus for teaching us how to implement our ideas and giving us their support when we needed it. Also a big thanks to Tomas Östberg and Staffan Qvarnström for helping us make the hardware and electronics for the robot and sharing their knowledge with us during the whole project.

Maria Markovska and Felicia Gihl Vieider
Stockholm, May 2017

Contents

1	Introduction	1
1.1	Background	1
1.2	Purpose	1
1.3	Scope	2
1.4	Method	2
1.4.1	Arm construction	2
1.4.2	Controlling the arm	3
1.4.3	Experiments and testing	4
2	Theory	5
2.1	Past research	5
2.2	Control	6
2.2.1	PID-control	6
2.2.2	Modeling the DC motor	7
2.3	Geometrical analysis	8
2.3.1	Newton Raphson method	8
3	Demonstrator	9
3.1	Problem formulation	9
3.2	Electronics	9
3.2.1	Microcontroller	10
3.2.2	Memory	10
3.2.3	Motors	10
3.2.4	Motor drivers	11
3.2.5	Reading the motor angles	11
3.3	Hardware	11
3.3.1	Arm unit	12
3.3.2	Pen holder	12
3.3.3	Rotational plate	12
3.3.4	Drawing plate	12
3.4	Software	12
3.4.1	Image processing	13
3.4.2	Geometrical analysis	14

3.4.3	Control system	15
3.5	Application testing	18
3.6	Results	19
4	Discussion and Conclusions	21
	Bibliography	23
	Appendices	
	A	Program code
	B	Datasheets

List of Figures

1.1	Picture of the robot arm modeled in CAD	3
3.1	Connection diagram of the electronics, made in Fritzing.	10
3.2	The complete robot with numbered parts	11
3.3	The structure of the program illustrated in a flowchart, made Microsoft powerpoint	13
3.4	The image processing step by step	14
3.5	Schematic picture of the arm mechanism drawn in Adobe Illustrator . .	14
3.6	Step responses for the big motor, plotted in Matlab	16
3.7	Step responses for the small motor, plotted in Matlab	16
3.8	The real step responses for each motor using PID-control	17
3.9	A picture of the experimental setup	18
3.10	A test picture fed to the robot	18
3.11	Measurement results with desired R set to 183 mm	19
3.12	Picture of the first test corresponding to the plot in Figure 3.11.	19
3.13	Measurement results with desired R alternating between 165mm and 183mm	20
3.14	Pictures of the circles drawn by the robot	20

List of Tables

3.1	Table over the lengths of the arm parts	14
-----	---	----

List of Abbreviations

Abbreviation	Description
CAD	Computer Aided Design
PID control	Proportional-Integral-Derivative control
3-dof	Three degrees of freedom
DC motor	Direct Current motor
PWM	Pulse Width Modulation
GND	Ground
PLA plastic	Polyactic Acid plastic

Chapter 1

Introduction

1.1 Background

Robotic arms are used in many applications in today's society. Nowadays, the possibilities of controlling and building robotics arms are increasing. Robotic arms are capable of accomplishing human arm tasks more precisely and faster. For this reason they are used in a widely spectra of modern industry, to increase the efficiency of the production in different ways.

A drawing robot, which is an example of a robotic arm, requires precise steering with a controller. There are a lot of different techniques to construct a controller and tuning it to get a desired system behavior. There are also different strategies of how to create a drawing using a robot. This project investigates different drawing and control methods for using the robotic arm in a drawing application.

1.2 Purpose

The purpose with this project is to create a 3-dof robotic arm that can be fed with a picture to draw on a paper. The drawing robot is used for testing two controller methods, PD and PID, and investigate the below research questions:

1. What drawing techniques have been used earlier when designing drawing robots?
 - Which technique fits better for this project?
2. Which controller is the most suitable for the robot, PD or PID?
3. What accuracy is achieved with the chosen controller?
 - How much does the real position differ from the given position of the pen?

1.3 Scope

This project is performed during one semester as a bachelor thesis at Kungliga Tekniska Högskolan. Due to this, there are limitations in both time and budget for making the project. The construction is therefore limited to the available resources. A lot of time during the project is invested in studying the techniques, theory and programming to be able to construct and control a robotic arm.

When modeling the robot arm, the friction, the inductance and external torque is neglected to simplify the control and avoid complicated models.

The size of the drawing is chosen to be $10x10\text{cm}$, due to the reach of the robot arm. The robot uses a black marker pen. Therefore the image can only be drawn in black.

1.4 Method

This part describes the method for answering the research questions.

1.4.1 Arm construction

After studying already existing robotic arms the arm is modeled using CAD Solid Edge ST8 [1], the construction is shown in Figure 1.1. A four axis mechanism design is chosen, which allows a 2-dof movement made by two cooperating motors. This construction is chosen because it allows the motors to be placed in the lower part of the construction. The placement of the motors is keeping the weight of the arm down to reduce the driving torque needed. To give the arm three degrees of freedom, the arm mechanism is mounted on a rotating base plate.

All flat parts are cut out in plexi glass using an Epilog laser cutter, and mounted together using steel shafts and nuts. To be able to draw with the arm, a pen is placed in the moving front part. A holder for the pen is modelled in CAD and printed using an Ultimaker. A hole is drilled and threaded on the front of the pen holder, making it possible to fasten the pen in the holder with a screw.

1.4. METHOD

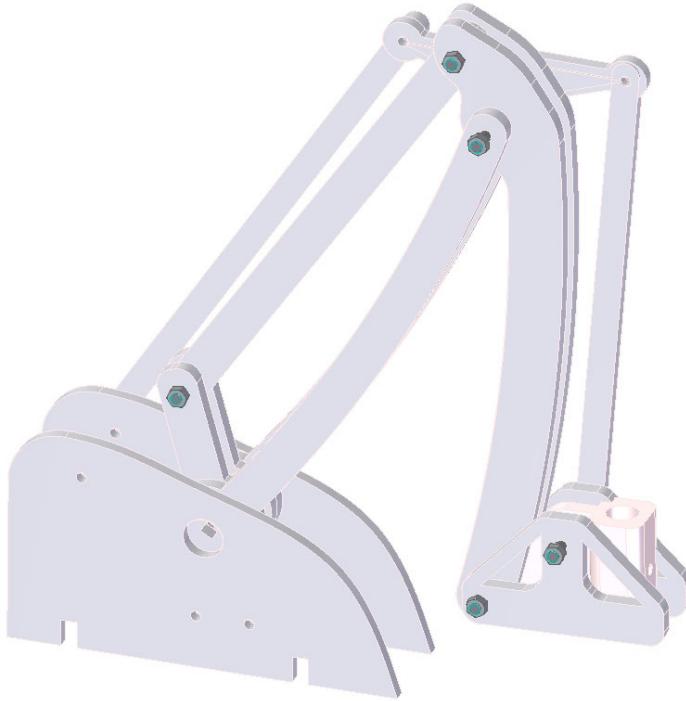


Figure 1.1. Picture of the robot arm modeled in CAD

1.4.2 Controlling the arm

When drawing a picture, the arm is moving the pen between different polar coordinates with the lower part of the mechanism as the origin. Polar coordinates is a system with two degrees described by a radius R and an angle ϕ . The angle corresponds to the position of the rotational plate and the radius is the length from the pen to origin.

To be able to control the arm, the radius has to be converted to the angles of the motor shafts. This is done by setting up an equation system to determine the geometrical relationships of the four axis mechanism. This gives the desired angles of the motor shafts based on a given radius. Two DC motors with transmission are used to move the arm mechanism. To control the arm, the motors are connected to an H-bridge controlled by an Arduino UNO. A potentiometer is connected to each DC motor to read the real angles of the motors. The error between the real and desired angle is used to control the pen position. A stepper motor is driving the rotational plate and is driven by a stepper motor driver connected to the Arduino.

1.4.3 Experiments and testing

To define the project, research questions are formulated in section 1.2. Answering mentioned research questions is the main purpose of the project and demands some experiments and testing. The first research question about which techniques that have been previously used for drawing robots is answered by making literature studies of previous projects. It will also be investigated which technique fits better for this specific project, due to it's purpose and scope. The second question about which controller is appropriate, is answered by implementing both PD and PID controllers for each motor and evaluating the results.

The third question about what accuracy is achieved is answered with the results from physical experiments. The first experimental part is to determine the position error between the real coordinates and the given ones. A simple test is made where the robot is steered to different given positions to draw a dot on a paper. By measuring the distance between the origin and the dot, the real coordinates are determined. The real coordinates are compared to the given ones for every test case to determine the error. Another experiment is to draw a full picture. This means that the robot is fed with a whole set of coordinates making a pattern. The test pattern is a circle.

Chapter 2

Theory

2.1 Past research

There are previous research in the area of designing drawing robots, using different techniques for drawing. Marcus Wallin uses a technique called pointillism to draw the images. In practice this means that the robot uses dots to draw a gray scale picture by varying the dot densities in different areas. Every dot is representing one coordinate on the drawing surface and the picture is drawn by lifting the pen between the coordinates for making the dots [2].

Radu-Mihai Pana-Talpeanu uses line drawing by giving the robots paths to follow without lifting the pen. To follow a path the arm needs to pass every point in correct order. To calculate the paths the thesis investigates different image processing techniques focused on facial recognition [3].

A. Mohammed et al. is discussing the steps needed to calculate a path used when drawing lines. First the image has to be processed to make filter the picture and identify inner and outer countours. The points in each contour must be organized in correct order whereafter the path can be planned using different Algorithms to remove redundant points and finding the optimal path [4].

The method of line drawing requires more calculations which is redundant for pro-cessing and drawing simple sketches. Pointillism is more simple and since the drawing in this project is made for measuring accuracy, the pointillism will make the measurements easier. For this reason the first method fits better for this specific project and is therefore chosen, answering the first research question.

Junyou et al. writes that the traditional way of making a drawing robot is to use inverse kinematics for calculation of the drawing position. This means that the picture is reduced to paths or coordinates which is translated to the pen position of the robot using its geometry. The thesis investigates the use of another method,

that is behaviour based, to produce the drawing. This means the robot movements are divided into different behaviours, for example drawing a straight line, an arc or a vertical bar. Then a picture is translated into a series of behaviours that the robot should go through. This method is supposed to resemble the human way of drawing to make the picture look like it is drawn by an artist. [5]

Hamori et al. builds a 3-dof lego arm to use as a drawing robot. The construction of the lego arm is similar to what this project aims for when making the design and building the robot. To get the position of the drawing tool this thesis uses inverse kinematics [6]. Since the goal with this project is not to make the drawing look human the traditional method with inverse kinematics is used.

2.2 Control

Position control means that the system is given a desired position, which the controller tries to achieve with different strategies depending on the controller type. The difference between the desired position and the real position is defined as the error, and controllers are used for compensating this error in real time to make the position as accurate as possible. There are a lot of methods to construct a controller for a robotic arm and a selection was made to test PD and PID control.

2.2.1 PID-control

A PID controller consists of three different parts that affects the error in different ways. The controller is described by the equation

$$u(t) = K \left(e(t) + \frac{1}{T_I} \int_{t_0}^t e(\tau) d\tau + T_D \frac{de(\tau)}{dt} \right) \quad (2.1)$$

where $u(t)$ is the input signal, K is the gain of the controller and $e(t)$ is the error. T_I and T_D are parameters used for the integrative and derivative parts of the controller. The proportional part (P-part) is adjusting the gain of the system. This means that it is directly affecting the speed of the system. An increased P-part makes the system faster responding. Increasing the P-part too much could give the system overshoot, which means that the desired value is exceeded. It also can reduce the stability of the system. The P-part gives the system a steady state error which can not be eliminated using only the P-part, because of the proportional relationship between the error and the input signal. To minimize the steady state error the integrating part (I-part) is used. Increasing the I-part makes the steady state error smaller, but again at the cost of stability for the system. To ensure stability and reduce the overshoot, the derivative part (D-part) is used. A controller can be built from each of the parts combined as a P, PI, PD or PID controller.

2.2. CONTROL

To find appropriate parameters for the controller, different strategies are used. By modeling the system, pole placement can be used for tuning controller parameters. This means the parameters are chosen by deciding where to place the poles for the closed loop system. The closed loop system G_c is described by the expression

$$G_c = \frac{FG}{1 + FG} \quad (2.2)$$

where F is the controller and G is the modeled open loop system. The n number of poles can then be placed solving the equation

$$(1 + FG) = (1 + pole)^n \quad (2.3)$$

One strategy is to place the poles for the closed loop system as near as possible to the open loop system. This placement will in theory result in an ideal controller for the system.

If the system is difficult to model mathematically, a tuning method called Ziegler Nichols can be used. This means that first T_I is set very big (as near to infinity as possible) and T_D is set to zero. This disconnects the I and D part. Then the P-part is increased until the system is oscillating with a constant amplitude, where the period time is measured. The period time is then used to tune the parameters by using standard rules for the Ziegler Nichols method. [7]

2.2.2 Modeling the DC motor

The transfer functions for DC-motors are derived from the differential equation

$$J\ddot{\theta} + \left(f + \frac{k_a k_w}{R_a} \right) \dot{\theta} = \frac{k_a}{R_a} u - \nu \quad (2.4)$$

where J is the rotor inertia [kgm^2], k_a is the torque constant [Nm/A], k_w is the voltage constant [V/rpm], R_a is the internal resistance, θ is the motor angle, u is the applied voltage [V], f is the friction and ν is the applied torque [Nm].

The friction and external torque are neglected, to simplify the equation. Taking the Laplace transform of equation (2.5) gives following transfer function $G(s)$ which is relating the input signal (applied voltage) to the output signal (motor angle)

$$G(s) = \frac{\frac{k_a}{k_a k_w}}{s \left(\frac{JR_a}{k_a k_w} s + 1 \right)} \quad (2.5)$$

and gives a simple model of the DC-motor systems that can be used to design a controller.

2.3 Geometrical analysis

The internal geometrical relationships in a four axis mechanism can be analyzed by setting up closed geometrical loops for the system, involving the unknown parameters that need to be determined. This is done by calculating the local vectors from a starting point in the system to every axis point in the mechanism and adding them up until reaching the start point of the loop again. This will result in an equation system where altogether the loop equations equals zero, since they start and end in the same point. The resulting equation system is non-linear and therefore a numerical method is needed to solve it.

2.3.1 Newton Raphson method

Non-linear equation systems can be solved using numerical method Newton Raphson. This is a method for finding the roots for equations in an approximate way. By giving a starting guess of the root that is somewhere in the near range of the real root, the method creates a tangent and checks the intersection with zero axis to get an approximate root. This is done stepwise using the following formula

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad (2.6)$$

where n is every iteration, and f is the function for which the root is approximated. The method keeps iterating with equation (2.6) to create more and more exact root approximations until a the chosen tolerance of deviation is achieved and thereby the root of the function is approximately determined. [8]

Chapter 3

Demonstrator

3.1 Problem formulation

The goal of this project is to design a drawing robotic arm. To reach the goal, the following problems are handled and further presented in this chapter:

- Designing and building a 3-dof drawing robot arm
- Moving the arm to a desired position
- Implementing two different feedback controllers
- Defining algorithms for processing a picture to feed the robot

3.2 Electronics

Electronics used for driving and controlling the robotic arm are a microcontroller, two DC motors, one stepper motor, two potentiometers, one H-bridge and one steppermotor driver. The schematics of the electronics are shown in Figure 3.1

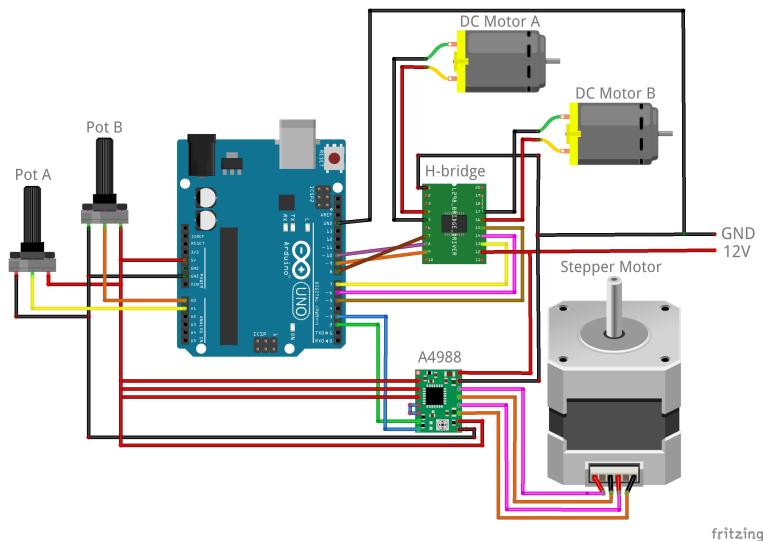


Figure 3.1. Connection diagram of the electronics, made in Fritzing.

3.2.1 Microcontroller

For controlling and driving the construction a microcontroller Arduino UNO is used. The way the Arduino is connected is shown in Figure 3.1. The Arduino takes input from the potentiometers through the analog pins A1 and A0. To control the other components the digital output pins are used.

3.2.2 Memory

To enable storage for larger sets of coordinates to feed the robot, a MicroSD card is used. The MicroSD card is plugged in the Arduino with a MicroSD card reader from Luxorparts.

3.2.3 Motors

The construction is driven by two DC motors. A worm gear motor Bosch AHC 24V is used to control the α angle and is referred to as the large motor. The β angle is driven by a SERVO DMN29BA-002 24V, this motor is referred to as the small motor. The main benefit with using a worm gear motor is that it is self-locking. This means that it remains locked in position when turned off, which is needed for accurate position control of the arm. The rotational plate is driven by a 2-Phase Hybrid Stepping Motor with 200 steps per revolution.

3.3. HARDWARE

3.2.4 Motor drivers

A H-bridge is used to control the speed of the DC-motors separately using a PWM signal from the Arduino. The H-bridge is built using standard components compound by soldering on a circuit board. The design of the H-bridge circuit is similar to a 2A Dual L298 H-bridge. The stepper motor is controlled by a A4988 Stepper Motor Driver Carrier and enables a selection from five step resolutions, up to a sixteenth step. This makes it possible to control the rotational degree with high precision.

3.2.5 Reading the motor angles

To keep track of the angles of the two DC motors, potentiometers with resistance 5kOhm and 10kOhm are used. The potentiometers follow the rotation of the shafts and give a voltage output to the Arduino, for it to continuously calculate the current position of the arm. For the stepper motor the angle position is calculated in the software, counting every time it has taken a step.

3.3 Hardware

The mechanism is described in four sections; arm part, pen holder, rotational plate and drawing plate. A view of the complete hardware with all the main parts numbered is presented in Figure 3.2.

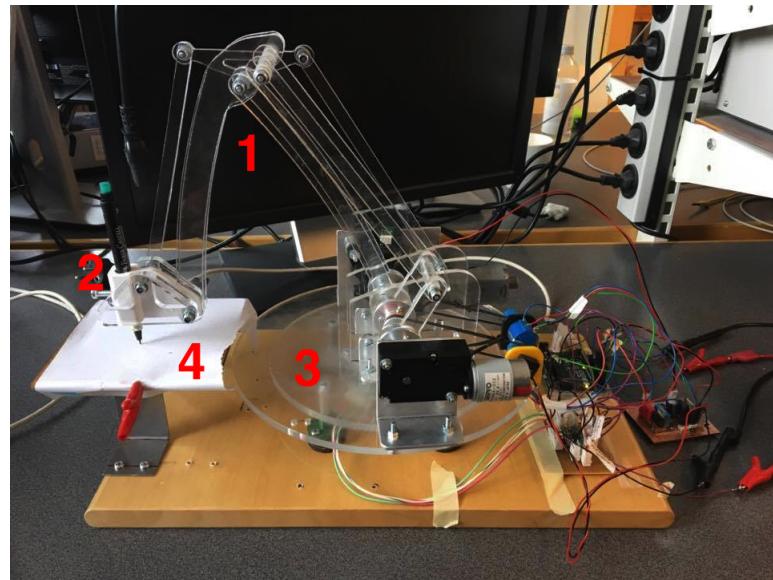


Figure 3.2. The complete robot with numbered parts

3.3.1 Arm unit

The arm construction (part 1 in Figure 3.2) has two degrees of freedom which makes it possible to access a limited area around it. The limitations of the reaching area are defined due to the chosen lengths of the arm links, which are presented in Section 3.4.2. The arm is made as a four axis mechanism which is driven by two motors at the lower part of the construction. The arm links are made in plexi glass using laser cutting. The different couplings in the mechanism are made in PLA plastic using 3D-printing or from lathering aluminum. All the arm joints are made by cutting and threading a 4 millimeter aluminum bar. Other small parts are standard components.

3.3.2 Pen holder

On the front of the construction there is a pen holder used for the drawing, see part 2 in Figure 3.2. The construction is simple, with a horizontal screw to attach the pen. The pen holder is 3D printed in PLA plastic using an Ultimaker.

3.3.3 Rotational plate

The third degree of freedom is made by the rotational plate at the bottom of the construction, this is shown in Figure 3.2 as part 3. The plates are made from plexi glass with laser cutting and assembled with screws and nuts. The stepper motor axis is connected to the upper plate with a screw coupling. The upper plate rotates directly on a fixed plate. No bearings are used because the friction between the plates is low enough for the rotation to manage smoothly.

3.3.4 Drawing plate

The drawing plate is placed in front of the robot to draw pictures on, see part 4 in Figure 3.2. The distance between the origin and the drawing plate is measured and used in the software when giving coordinates to the robot. The drawing plate is laser cut in wood and fastened with bent aluminum plates.

3.4 Software

Arduino IDE [9] is used to program the Arduino board. This software allows the user to write programs and upload them to the board with a USB cable. The programming language used is C/C++. The written code is presented in Appendix A. A PID library [10] is used for the controller and a sorting library [11] is used for filtering the potentiometer values.

The program reads coordinates from the MicroSD card. The potentiometer values are read to determine the current angles of the DC motors and the controller calculates the voltage output to the motors. The voltage output is controlled by

3.4. SOFTWARE

sending PWM signals to the H- bridge. A PWM signal is a square wave signal varying between a low and a high voltage (0-12V in this case). The time is distributed between low and high modes to output a certain voltage between 0 and 12V to the motors. A flow chart illustrating the logic of the used program is presented in Figure 3.3.

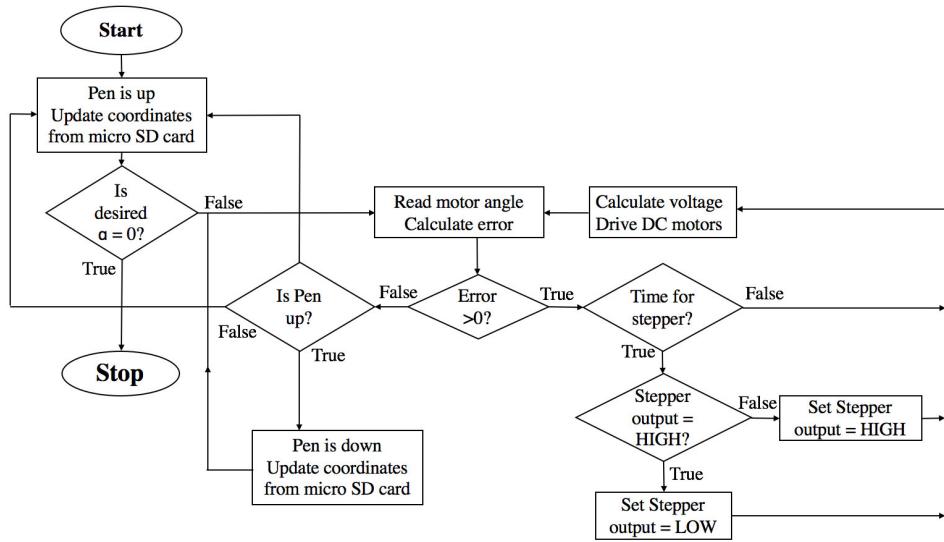


Figure 3.3. The structure of the program illustrated in a flowchart, made Microsoft powerpoint

3.4.1 Image processing

The input for the drawing robot is a gray scale picture. The input picture is processed into suitable information, which in this case is the coordinates for every dot. To do this the picture is translated into pixels that are assigned with any of the three different values white, gray and black. The image processing algorithm is reading the picture pixel by pixel and the amount of blackness in the pixel, decides which shade will be represented. Every pixel is set to contain the area of a chosen number dots. The black pixels are drawn with 100 percent dots in the pixel area, the gray with 50 percent and the white with 0 percent. In this way the robot will generate a gray scale image with three different shades of brightness of the pixels. The image processing of an example picture step by step is shown in Figure 3.4.

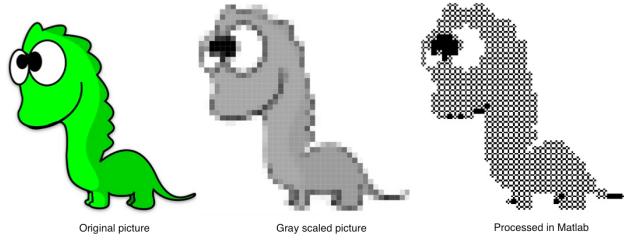


Figure 3.4. The image processing step by step

3.4.2 Geometrical analysis

An analysis of the geometry of the four bar mechanism was made to determine the relationship between the four axis geometry and the position of the pen. A schematic picture of the mechanism in a two dimensional view is presented below in Figure 3.5. Parameters for the arm parts of the mechanism are presented in Table 3.4.2.

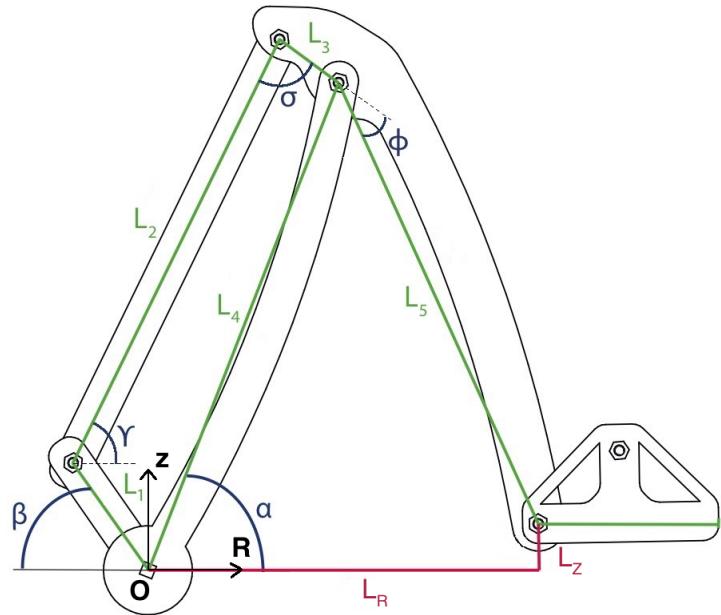


Figure 3.5. Schematic picture of the arm mechanism drawn in Adobe Illustrator

L	1	2	3	4	5
mm	50	180	28	200	185

Table 3.1. Table over the lengths of the arm parts

3.4. SOFTWARE

The unknown parameters to be determined are the angles, their dependency of each other and a given pen position. The analysis is made by determining closed geometrical loops of the arm geometry, as explained in section 2.3. This gives the equation system

$$0 = L_1 \cos \beta - L_2 \cos \gamma - L_3 \cos \sigma + L_4 \cos \alpha \quad (3.1)$$

$$0 = L_1 \sin \beta + L_2 \sin \gamma - L_3 \sin \sigma - L_4 \sin \alpha \quad (3.2)$$

$$0 = -L_4 \cos \alpha - L_5 \cos \sigma + \phi \quad (3.3)$$

$$0 = L_4 \sin \alpha - L_5 \sin \sigma + \phi \quad (3.4)$$

The system is solved with the numerical method Newton Raphson to require full information about the angles dependency on each other and on the given position. The result from this analysis is implemented in the software to relate position of the motor angles to the position of the drawing tool.

3.4.3 Control system

The transfer functions for the DC motors are given in section 2.2.2. The parameters for each motor are available in or calculated from the data sheets, Appendix B. The external torque is neglected, assuming it is not affecting the behavior of the system significantly. The inertia of the robot arm is calculated approximately by taking the inertia in the center of mass for all the arm parts separately in CAD. The axis of the inertia of each part is then moved to the origin with Steiner's parallel axis theorem

$$I_O = I_G + md^2 \quad (3.5)$$

where I_O is the origin inertia, I_G is the mass center inertia, m is the mass of the body and d is the distance to origin. The inertia for each arm are summarized into one total inertia. Since the total inertia of the arm mechanism is dependent on the angle position, it is calculated in Matlab [12] for some different angles. The maximum evaluated inertia is chosen for the system model. The motor inertia is assumed to be much smaller than the inertia of the robot arm and is neglected.

The parameters for the PD and PID control are calculated using pole placement. The step responses for the closed loop systems using the calculated parameters for PD and PID are presented in Figure 3.6 for the large DC motor and Figure 3.7 for the smaller one.

Implementing the PD-controller, the robotic arm does not move at all. The reason for this is that a minimum voltage is required for the motors to be able to rotate. The minimum voltage is never exceeded when using only a PD-controller, since it does not take previous time into account (because the integral part is missing). The conclusion is that a PD-controller is not suitable to use in this application, since the I-part is necessary for the robot to move. This answers the second research question, concluding that PID-control is the most suitable for this system.

CHAPTER 3. DEMONSTRATOR

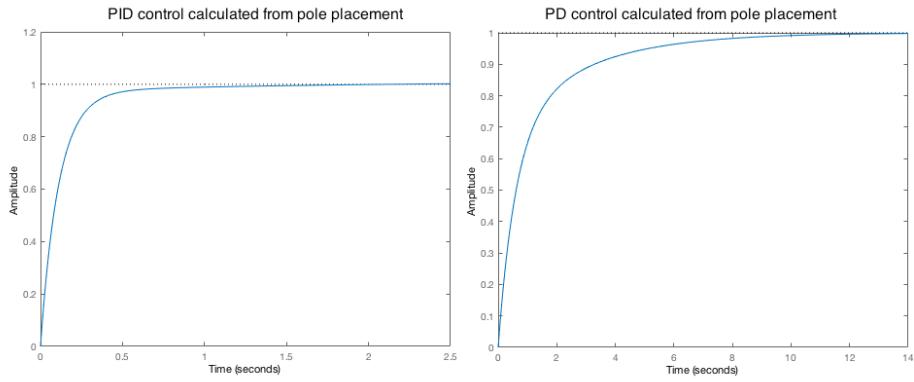


Figure 3.6. Step responses for the big motor, plotted in Matlab

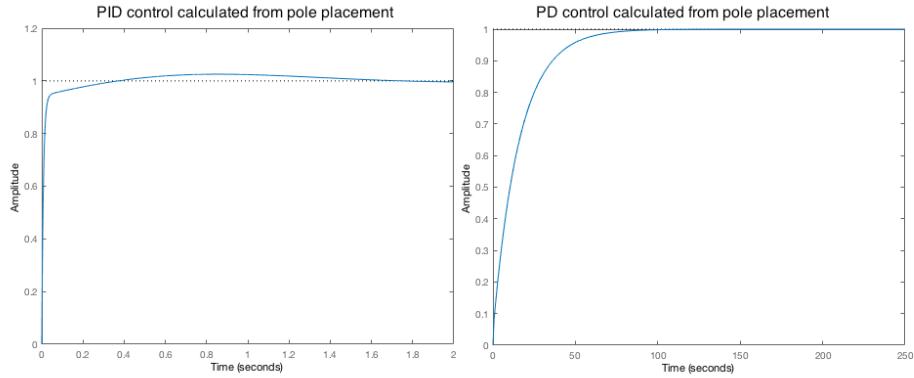


Figure 3.7. Step responses for the small motor, plotted in Matlab

Implementing the PID-controller, the parameters from pole placement does not give the robot a desired movement. The small motor is oscillating a lot and the large motor is working but not giving an accurate position. Since the parameters for the small motor are not working at all, Ziegler Nichols method is used to get new parameters for the small motor. Implementing the Ziegler Nichols parameters on the small motor gives a working movement but not an accurate position. The parameters for both motors are therefore adjusted experimentally after the implementation of the controller. This is done by drawing and measuring accuracy while changing the parameters a bit at a time, until the parameters giving the best accuracy are found.

For the large motor the parameters are separated in two sets, aggressive and conservative parameters, meaning that the aggressive parameters are tuned to correct

3.4. SOFTWARE

the error faster and with less damping than the conservative parameters. This is because of a problem occurring when the error is too small, the voltage gets to low and therefore it needs more aggressive controlling. When the error is bigger the aggressive parameters are too fast and gives the robot an undesired overshoot, therefore more conservative parameters are implemented when the error is larger. These adjustments are not needed for the small motor, since the angle range for it's movement is much smaller. A plot of the real step response using the chosen PID-parameters is presented in Figure 3.8.

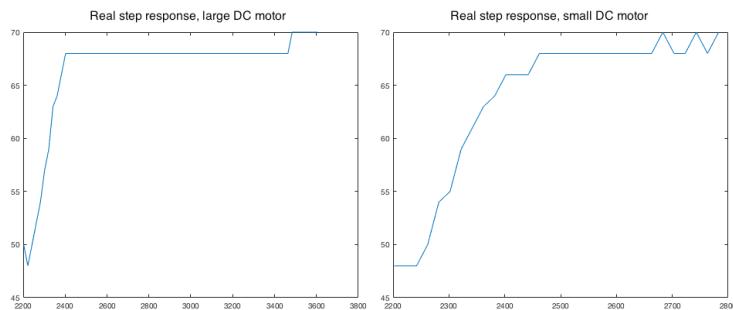


Figure 3.8. The real step responses for each motor using PID-control

3.5 Application testing

The testing is divided into different experiment setups. The results from the experiments are presented in section 3.6. A picture of the setup for making the experiments is presented in Figure 3.9.

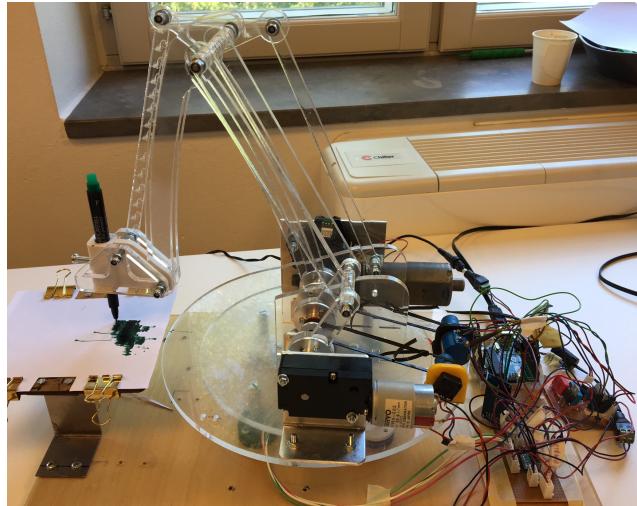


Figure 3.9. A picture of the experimental setup

One experiment is to make the robot draw dots on chosen coordinates on a paper. The drawn dots distance to the origin is measured and the measured values are compared with the given . The next experiment is to alternate between drawing two different coordinates, to evaluate how this affects the accuracy. The last experiment is to make the robot draw a full picture. The test picture fed to the robot is seen in Figure 3.10.

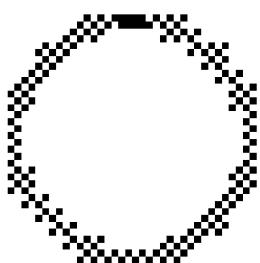


Figure 3.10. A test picture fed to the robot

3.6. RESULTS

3.6 Results

The first experiment is to give the robot the same coordinate several times. This was done twice, changing the potentiometer value corresponding to α with a one degree step in between the tests. Results for both tests are presented in Figure 3.11.

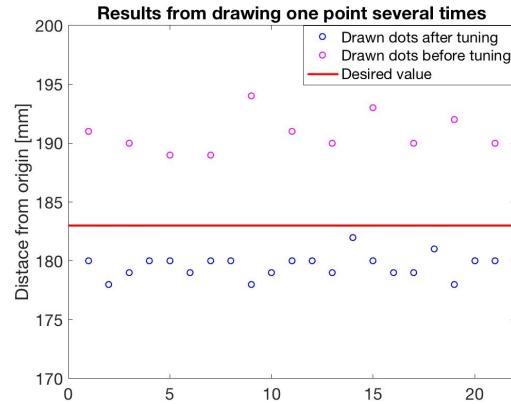


Figure 3.11. Measurement results with desired R set to 183 mm

The first test is represented with dark blue dots and the second with magenta dots. The desired value is represented with a red line. In the first test the real coordinate values differ between 178mm and 182mm. After tuning the potentiometer value the second test was made, resulting in values differing between 189mm and 194mm. Both tests give around the same deviation from the desired value, but in different directions. This shows that a change with one degree in α results in a change of 11mm in position of the pen. A picture of the test is seen in Figure 3.12.



Figure 3.12. Picture of the first test corresponding to the plot in Figure 3.11.

CHAPTER 3. DEMONSTRATOR

The other experiment is to make the robot alternate between two different coordinates. The result from this test is shown in Figure 3.13.

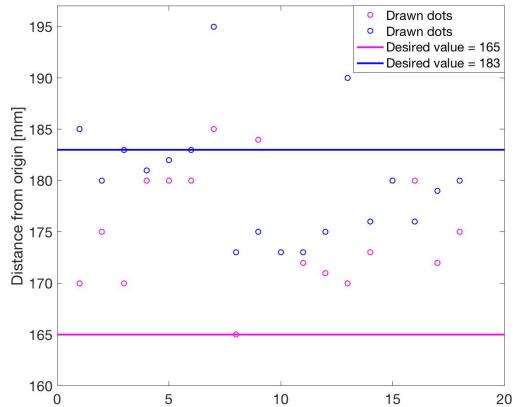


Figure 3.13. Measurement results with desired R alternating between 165mm and 183mm

The results with desired value of 183mm are represented with dark blue color, and the results with desired value 165mm are in magenta. The desired coordinates are chosen so that α differs with two degrees in between them. The test results show that when altering between the two desired coordinates, the robot draws spread dots in between them. The tests show that the accuracy is $\pm 11\text{mm}$.

The last experiment is to give the robot a full set of coordinates forming a circle, see Figure 3.10 in section 3.5. This test was made several times while tuning the PID-parameters and the potentiometer reading the α angle. Some of the results are presented in Figure 3.14. The test shows that a positive error in α gives a more rectangular shaped picture, which can be observed in the rightmost circle in Figure 3.14.



Figure 3.14. Pictures of the circles drawn by the robot

Chapter 4

Discussion and Conclusions

The research questions have been answered and a summary of the answers is presented as follows. Investigating *what drawing techniques have been used earlier*, we looked at two different drawing techniques used in previous projects, choosing one of them to implement on the robot. The choice was based on simplicity and measurability, comparing the two methods. When investigating *which controller is the most suitable for the robot*, we implemented PD and PID controllers, concluding that PD-controller did not work at all for our application. *What accuracy is achieved*, was answered by making drawing experiments to measure the accuracy. The result was an accuracy of $\pm 11\text{mm}$. While, the result is not as accurate as it was expected to be, the possible error sources are investigated and discussed below.

There are some error sources that depends on other factors than the controllers. For example a small error is caused by deviations from the modeled construction when mounting the arm. Some length used in calculations was measured by hand. This could have made the real mechanism differ a little bit from the model we used in our calculations. The desired angles are approximated with numerical methods and therefore the calculated position will have a small error. But since we set a tolerance of 10^{-9} for the approximation this is not significant.

The biggest error source turned out to be the resolution of the potentiometers. Because of the low resolution, the error tolerance had to be set to one degree to cover all angles. The results showed that a small angle error gives a large impact on the position and therefore this has affected the accuracy significantly. This error can be reduced by using a higher resolution in the potentiometers, and can be a future improvement of the project. Since we mounted and calibrated them by manual tuning some offset between the real angles and the sampled ones may have occurred. The potentiometers were calibrated continuously during the project to avoid this type of offset.

CHAPTER 4. DISCUSSION AND CONCLUSIONS

Using pole placement for tuning the controllers did not work out well. The reason is that the theoretical model was different from the real system. Some of the motor parameters were approximated and the geometry was not taken into account. The geometry of the mechanism is not linear and therefore hard to model. Since this was neglected the model does not include the impact of the motors on each other when rotating. Taking this into account would make a more accurate model to use when designing the controllers. The system model could have been made more complex, but that will require more advanced calculations. Also, the system feedback could have been made differently using a sensor to detect the pen position instead of calculating it geometrically. Some problems occurred with getting the height of the pen correct for all the different coordinates in one picture using only geometrical calculations. This potentially could have been solved using a pressure sensor to adjust the height with feedback.

The drawing implementation in this project, was used to measure accuracy for the position of a robotic arm. The purpose with the drawing part was to visualize the position in an easy and measurable way. If the focus would have been on making a robot that is optimized for drawing, another design would have been more appropriate. For example linear actuators could have been used to steer it. This would have been more similar to a regular printer construction, which probably is a more effective way to draw a picture.

The construction was made in many iterations, improving it in small steps at a time. One lesson learned is the importance of setting up the electronics in a stable way. A lot of problems occurred with loose connections, especially for the H-bridges and the potentiometers. Also a lot has been learnt about making a full construction and creating all the parts from scratch. Redoing this a lot of times has given us a good insight of what is needed to create a complete and sustainable construction.

Bibliography

- [1] *Solid Edge ST8*, <https://www.plm.automation.siemens.com/>, Date accessed: 2017-05-16.
- [2] M. Wallin, “Robotic illustration,” Kth, Tech. Rep., 2013 . Date accessed: 2017-04-18. [Online]. Available: <http://www.diva-portal.org/smash/get/diva2:697863/FULLTEXT01.pdf>
- [3] Radu-Mihai and Pana-Talpeanu, “Trajectory extraction for automatic face sketching,” KTH, Tech. Rep., 2013 . Date accessed: 2017-04-18. [Online]. Available: <http://www.diva-portal.org/smash/get/diva2:700505/FULLTEXT01.pdf>
- [4] M.Abdullah, W.Lihui, and Rx.Gao, “Integrated image processing and path planning for robotic sketching,” Tech. Rep., 2013 . Date accessed: 2017-06-05. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S2212827113006768?via%3Dhub>
- [5] Y. Junyou, Q. Guilin, M. Le, B. Dianchun, and H. Xu, “Behavior-based control of brush drawing robot,” 2011 . Date accessed: 2017-06-05. [Online]. Available: <http://ieeexplore.ieee.org.focus.lib.kth.se/document/6199408/metrics?part=1>
- [6] Á. Hámori, J. Lengyel, and B. Reskó, “3dof drawing robot using lego-nxt,” Alba Regia University, Center Óbuda University, Hungary, Tech. Rep., 2011 . Date accessed: 2017-06-05. [Online]. Available: <http://ieeexplore.ieee.org.focus.lib.kth.se/document/5954761/?part=1>
- [7] Glad and Ljung, *Reglerteknik - Grundläggande teori*, 2006.
- [8] T. Sauer, *Numerical Analysis 2nd edition*, 2012.
- [9] *ARDUINO IDE 1.8.2*, <https://www.arduino.cc/en/main/software>, Date accessed: 2017-04-18.
- [10] *PID Library Arduino*, <https://github.com/br3ttb/Arduino-PID-Library>, Date accessed: 2017-05-16.
- [11] *Sorting Library Arduino*, <https://github.com/emilv/ArduinoSort>, . Date accessed: 2017-05-16.

BIBLIOGRAPHY

[12] *Matlab r2016a*, <https://se.mathworks.com>, Date accessed: 2017-05-16.

Appendix A

Program code

This appendix contains the code used for the robot.

```

// Main program for drawing robot
// Bachelor's Thesis in Mechatronics, KTH
// Felicia Gihl Vieider & Maria Markovska
// 2017-05-16

#include <ArduinoSort.h>
#include <Readingpotentiometers.h>
#include <Drive_DC.h>
#include <PID_v1.h>
#include <SD.h>

// Making objects for reading potentiometers (pin,mapping_start,mapping_range)
Define_pot blue(1,-26,1834);
Define_pot black(0,-5,242);

// Making objects for driving motor (motor_pin,in1,in2,max_volt)
Drive_DC stormotor_A(10,8,9,12);
Drive_DC litenmotor_B(5,6,7,12);

// Define variables for Setpoint (desired angle), Input (potentiometer angle), Output (voltage)
double Setpoint_A, Input_A;
double Output_A;
double Setpoint_B, Input_B;
double Output_B;
double Setpoint_step;

// Define home state coordinates
double Home_state_A = 70;
double Home_state_B= 45;
double Home_state_stepper= 90;

// Counter for reading through the coordinate vectors
int state_count = 0;

// State variable turn is true while the picture is not done yet
int turn = true;

// State variables to shift z state (pen height)
int z = 0;
int y = 0;

//Specify the links and initial tuning parameters
//double aggKp_A=1.5, aggKi_A=1.4, aggKd_A=2; // Aggresive tuning
double consKp_A=1, consKi_A=1.1, consKd_A=1.1; // Conservative tuning
double Kp_B=1, Ki_B=1.2, Kd_B=1;

// Making PID objects
PID A_PID(&Input_A, &Output_A, &Setpoint_A, consKp_A, consKi_A, consKd_A, DIRECT);
PID B_PID(&Input_B, &Output_B, &Setpoint_B, Kp_B, Ki_B, Kd_B, DIRECT);

// Define global variables for filter function, filter_input
int i = 0;
int A_vektor[5];
int B_vektor[5];
int Error_A;
int Error_B;

// Define stepper variables
unsigned long steptime = 20; // Sample time for stepper: Decides stepper speed
double steps = 0; // Count how long the stepper has moved
double desired_steps; // Calculates how long the stepper needs to move to reach setpoint
unsigned long currentMillis; // Current time
unsigned long previousMillis = 0; // Previous time
int stepState = LOW;
int dirState = LOW;
double Error_step;

// Define start angle for stepper
double step_angle = 90;

// Define global variables for reading from SD card
File file;

```

```

double Val_vec1[] = {1,1,1,1,1,1};
int Setpoint_A_1 = Home_state_A;
int Setpoint_B_1 = Home_state_B;
int Setpoint_A_0 = Home_state_A;
int Setpoint_B_0 = Home_state_B;
double Setpoint_step_new = Home_state_stepper;
bool row_is_correct = false;

// Function to read file from SD card
void read_file(int state_count, int i){
    file = SD.open("cirkel.txt");

    while (file.available()) {
        Val_vec1[0] = file.parseInt();
        Val_vec1[1] = file.parseInt();
        Val_vec1[2] = file.parseInt();
        Val_vec1[3] = file.parseInt();
        Val_vec1[4] = file.parseFloat();
        Val_vec1[5] = file.parseInt();

        i = i+1;

        if(i == state_count){
            row_is_correct = true;
        }
        else {row_is_correct = false;}

        if(row_is_correct) {break;}
    }
    file.close();
    return;
}

void setup() {
    pinMode(10, OUTPUT);
    pinMode(8, OUTPUT);
    pinMode(9, OUTPUT);
    pinMode(7, OUTPUT);
    pinMode(5, OUTPUT);
    pinMode(6, OUTPUT);
    pinMode(2,OUTPUT); // Step
    pinMode(3,OUTPUT); // Dir
    pinMode(1,OUTPUT);
    pinMode(0,OUTPUT);
    pinMode(13,OUTPUT);
    digitalWrite(1, HIGH);
    digitalWrite(0, HIGH);
    digitalWrite(13, HIGH);
    delay(100);
    Serial.begin(9600);
    SD.begin(4);

    // Initialize the variables we're linked to
    Input_A = blue.read_pot();
    Input_B = black.read_pot();

    // Turn the PID on
    A_PID.SetMode(AUTOMATIC);
    B_PID.SetMode(AUTOMATIC);

    // Create vectors used in filter function
    int num = blue.read_pot();
    A_vektor[0] = num;
    A_vektor[1] = num;
    A_vektor[2] = num;
    A_vektor[3] = num;
    A_vektor[4] = num;

    int num_2 = black.read_pot();
    B_vektor[0] = num_2;
    B_vektor[1] = num_2;
}

```

```

B_vektor[2] = num_2;
B_vektor[3] = num_2;
B_vektor[4] = num_2;

Setpoint_A = Home_state_A;
Setpoint_B = Home_state_B;
Setpoint_step = Home_state_stepper;

// Calculate desired steps
desired_steps = Setpoint_step-step_angle;

Error_A = abs(Input_A-Setpoint_A);
Error_B = abs(Input_B-Setpoint_B);
Error_step = abs(steps-desired_steps);

}

//Filter function to filter the input from the potentiometers
int filter_input(Define_pot *color,int vektor[]){
    vektor[i] = color->read_pot();
    //Serial.println(vektor[i]);
    i++;
    if(i== 5){i = 0; }

    int resultat[5];

    resultat[0] = vektor[0];
    resultat[1] = vektor[1];
    resultat[2] = vektor[2];
    resultat[3] = vektor[3];
    resultat[4] = vektor[4];
    sortArray(resultat,5);

    return resultat[2];
}

void loop() {

// Check error in Stepper
if (abs(steps-desired_steps)>0.2){
    // Set step direction
    if (0>desired_steps){dirState = LOW;}
    else{dirState=HIGH;}

    currentMillis = millis(); // Save current time
    if (currentMillis - previousMillis >= steptime) {
        // Save the last time the stepper made a step
        previousMillis = currentMillis;

        // If last step was HIGH turn it to LOW and vice-versa:
        if (stepState == LOW) {stepState = HIGH;}
        else {stepState = LOW; }

        // Set the output to stepper:
        digitalWrite(3, dirState);
        digitalWrite(2, stepState);

        // Add the step angle to know where the stepper is
        if (dirState==HIGH){steps = steps+1.8/16;}
        else{steps = steps-1.8/16;}
        //Serial.println(steps);
        //Serial.println(currentMillis);
    }
}

// No stepper needed, set stepper LOW
else{digitalWrite(2, LOW);}

// Read potentiometer angles with a filter
Input_A = filter_input(&blue,A_vektor);

```

```

Input_B = filter_input(&black,B_vektor);

int gap = abs(Setpoint_A-Input_A); // Check distance away from setpoint

//Use adaptive tuning:
//if (gap < 4)
// { //we're close to setpoint, use conservative tuning parameters
//   A_PID.SetTunings(aggKp_A, aggKi_A, aggKd_A);
// }
//else
//{ //we're far from setpoint, use aggressive tuning parameters
//   A_PID.SetTunings(consKp_A, consKi_A, consKd_A);
// }

// Calculate Dc motor output
A_PID.Compute();
B_PID.Compute();

// Drive DC motors
if(abs(Input_A-Setpoint_A)<3){Output_A =0;}
litenmotor_B.drive_motor(Output_B);
stormotor_A.drive_motor(Output_A);

// Check error for all motors
Error_A = abs(Input_A-Setpoint_A);
Error_B = abs(Input_B-Setpoint_B);
Error_step = abs(steps-desired_steps);

// Compare error with tolerance
if ( Error_B<2 && Error_A<2 && Error_step < 0.2){

// Check if there is more coordinates in the vectors
if(Setpoint_A_1 == 0){ // If the vectors are finished, go to home state
  if(turn == true){
    Setpoint_A = Home_state_A;
    Setpoint_B = Home_state_B;
    Setpoint_step = Home_state_stepper;
    step_angle = step_angle + steps;
    steps = 0;
    desired_steps = Setpoint_step-step_angle;
    A_PID.Compute();
    B_PID.Compute();
    Serial.println("END");
    turn = false;
  }
}

if(turn == true){
  if (z == 0){
    if ( Error_B<2 ){
      z = 1;
      Setpoint_A = Setpoint_A_1;
      Setpoint_B = Setpoint_B_1-6;
      Setpoint_step = Setpoint_step_new;
      step_angle = step_angle + steps;
      desired_steps = Setpoint_step-step_angle;
      steps = 0;
      A_PID.Compute();
      B_PID.Compute();
    }
  }
}

else if(z==1 && y == 0){
  if ( Error_B<2 && Error_A<2 ){
    Setpoint_A = Setpoint_A_0;
    Setpoint_B = Setpoint_B-3;
    y = 1;
    desired_steps = 0;
    step_angle = step_angle + steps;
    steps = 0;
    A_PID.Compute();
  }
}

```



```

// Library for driving DC motors
// Bachelor's Thesis in Mechatronics, KTH
// Felicia Gihl Vieider & Maria Markovska
// 2017-05-16

#include <Arduino.h>
#include "Drive_DC.h"

Drive_DC::Drive_DC(int motor, int in1, int in2, int maxvoltage)
{
    _motor = motor;
    _in1 = in1;
    _in2 = in2;
    _maxvoltage = maxvoltage;
    pinMode(motor, OUTPUT);
    pinMode(in1, OUTPUT);
    pinMode(in2, OUTPUT);
}

void Drive_DC::counter_clockwise(int PWM)
{
    digitalWrite(_in2, LOW);
    digitalWrite(_in1, HIGH);
    analogWrite(_motor, (PWM));
}

void Drive_DC::clockwise(int PWM)
{
    digitalWrite(_in2, HIGH);
    digitalWrite(_in1, LOW);
    analogWrite(_motor, PWM);
}

void Drive_DC::drive_motor(float PWM)
{
    if (PWM > _maxvoltage) { PWM=_maxvoltage; }
    if (PWM<- _maxvoltage) { PWM=-_maxvoltage; }

    if (PWM <= 0)
    {
        Drive_DC::counter_clockwise(map(abs(PWM), 0, _maxvoltage, 0, 255));
    }
    else if (PWM > 0)
    {
        Drive_DC::clockwise(map(PWM, 0, _maxvoltage, 0, 255));
        //Serial.println(map(PWM, 0, _maxvoltage, 0, 255));
    }
}

```

```
// Library for reading potentiometers
// Bachelor's Thesis in Mechatronics, KTH
// Felicia Gihl Vieider & Maria Markovska
// 2017-05-16

#include <Arduino.h>
#include "Readingpotentiometers.h"

Define_pot::Define_pot(int pin,int mappingstart,int mappingrange)
{
    _pin = pin;
    _mappingstart = mappingstart;
    _mappingrange = mappingrange;
}

int Define_pot :: read_pot()
{
    potVal = analogRead(_pin);
    pot_angle = map(potVal, 0 ,1023 , _mappingstart, _mappingrange);

    return pot_angle;
}
```

Appendix B

Datasheets

This appendix contains the data sheets for all the motors.

D.C. motors with transmission

AHC and AHC 2 | 24 V 17,3 W

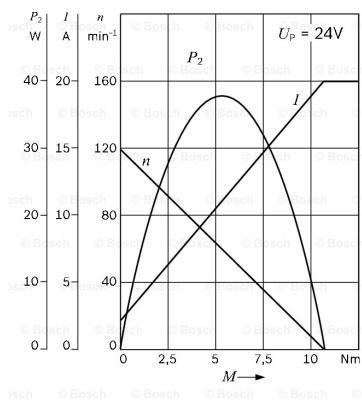
optional Hall sensor available



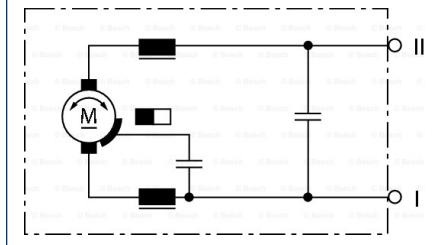
Technical data

Part number	0 390 203 386
Nominal voltage	U_N 24 V
Nominal power	P_N 15,6 W
Nominal current	I_N ≤ 4,8 A
Maximum current	I_{max} 20 A
Nominal speed	n_N 99,5 min ⁻¹
Nominal torque	M_N 1,65 Nm
Breakaway torque	M_A ≥ 11 Nm
Reduction	i 50 : 1
Direction of rotation	Anti-clockwise
rotation	
Type of duty	short-time duty
Degree of protection	IP 50
Weight	approx. 0,44 kg
Clockwise	I to (-), II to (+)
Anti-clockwise	I to (+), II to (-)
	160 mm
	50 mm
	5 min
	Connector Housing
	TE AMP 968182-1 / Connector Locker TE
	AMP 0-968183-1 / Connector Terminals TE
	AMP Junior-Power-Timer 0-927768-1 (x2)

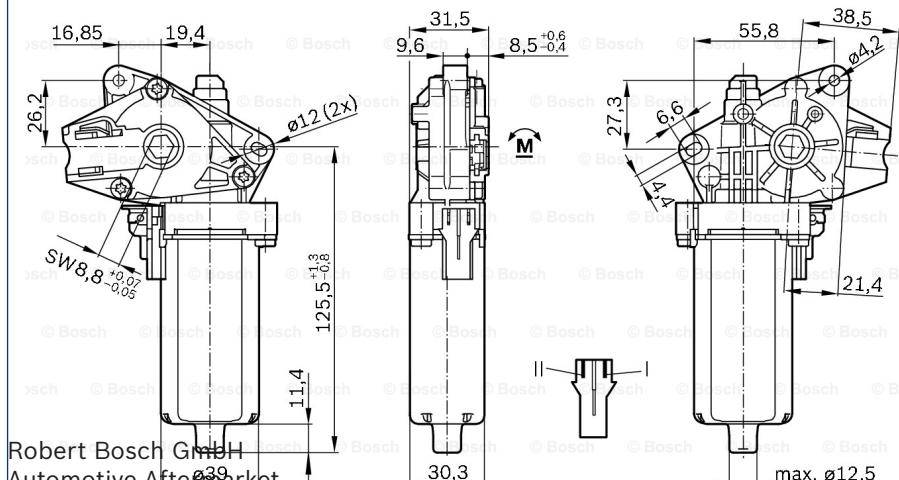
Characteristic curve



Connection diagram



Dimensional drawing



Robert Bosch GmbH
Automotive Aftermarket
Postfach 410960
76225 Karlsruhe
Germany

www.bosch-elektromotoren.de



BOSCH
Invented for life

DMN29 Series



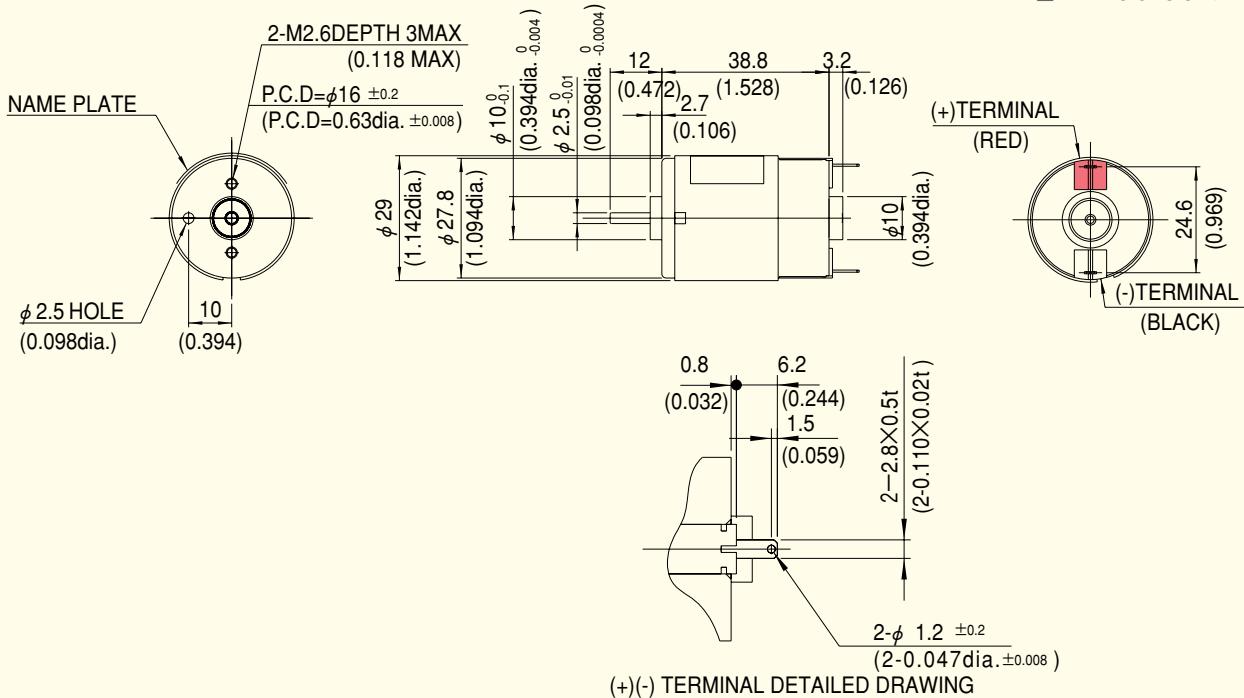
Specification

TYPE	RATED					NO LOAD		STALL		WEIGHT		
	OUT PUT	VOLTAGE	TORQUE		CURRENT	SPEED	CURRENT	SPEED	TORQUE			
			W	V					mN·m	oz·in	g	lb
DMN29BA	3.0	12	7.8	1.11	0.42	3700	0.07	5000	30	4.17	90	0.20
DMN29BB	3.0	24	7.8	1.11	0.21	3700	0.05	5000	30	4.17	90	0.20

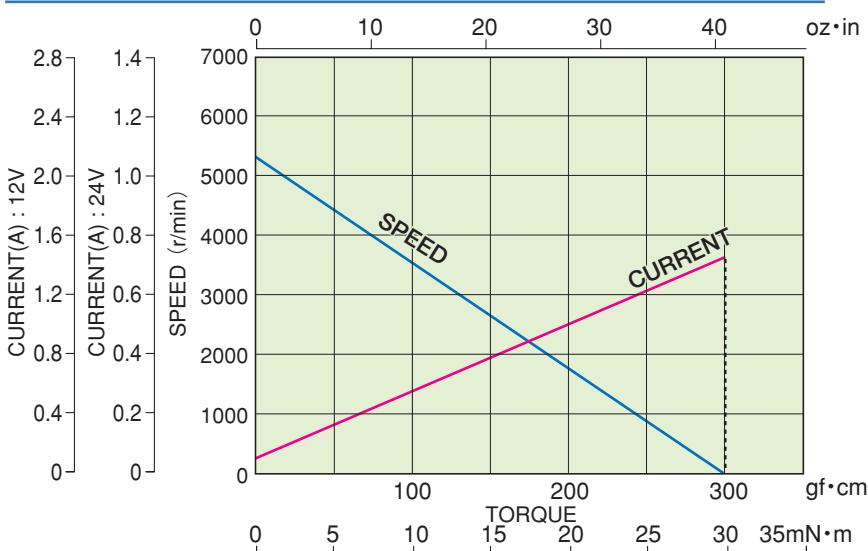
Outline

DMN29

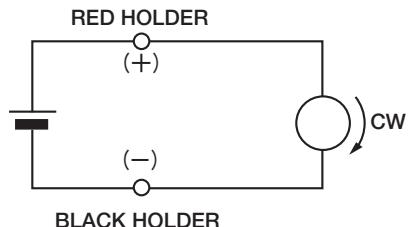
DIMENSIONS Unit mm(inch)



CURRENT, SPEED-TORQUE CURVE



CONNECTION



A

Intermittent Operation

**Specification**

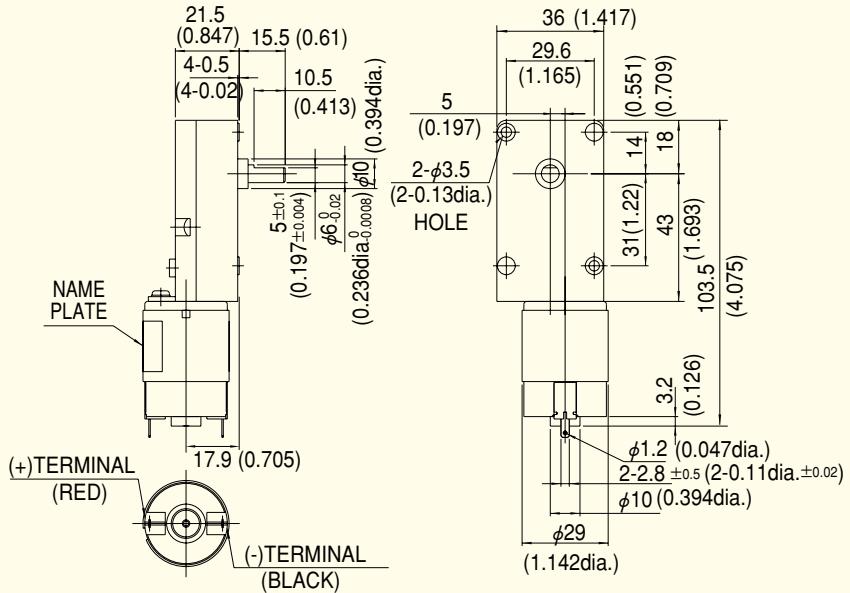
GEAR RATIO	DMN29BA-002 DMN29BA-003		
	RATED TORQUE		SPEED
	N·m	oz·in	r/min
*	78.9	0.190	27.8 56

※Rotation of gearbox shaft is in reverse of rotation of motor.

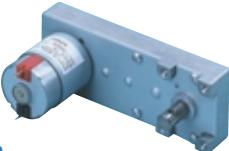
Outline

DMN29BA-002, 003 WEIGHT:140g 0.31 lb

DIMENSIONS Unit mm(inch)

**L**

Intermittent Operation

**Specification**

GEAR RATIO	DMN29BL□◇		
	RATED TORQUE		SPEED
	N·m	oz·in	r/min
30	0.14	19.5	123
50	0.23	32.0	74.0
120	0.56	77.9	30.8
150	0.69	90.8	24.7
200	0.92	131	18.5
255	0.98	139	15.3

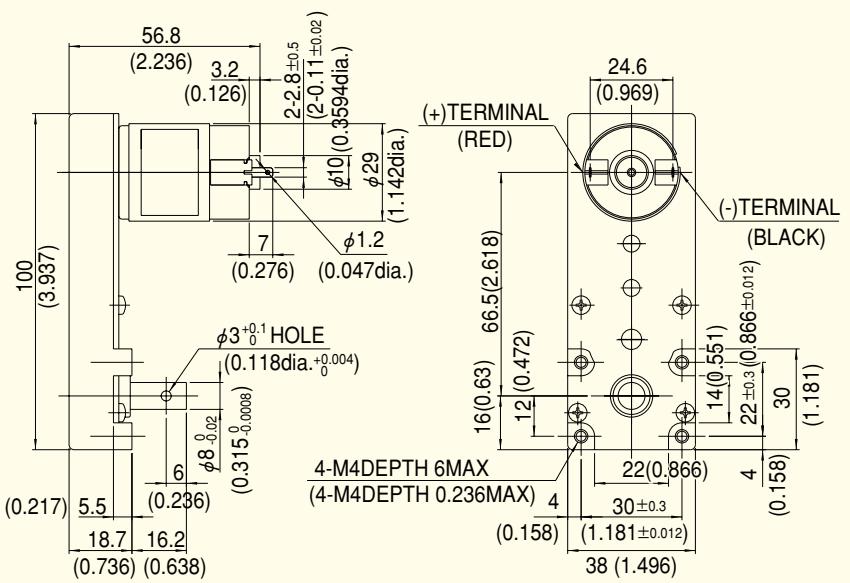
※Enter the required reduction ratio in the □.

※Enter the required voltage A or B in the ◇.

Outline

DMN29BL□◇ WEIGHT:250g 0.55 lb

DIMENSIONS Unit mm(inch)



2-Phase Hybrid Stepping Motor

1.8°

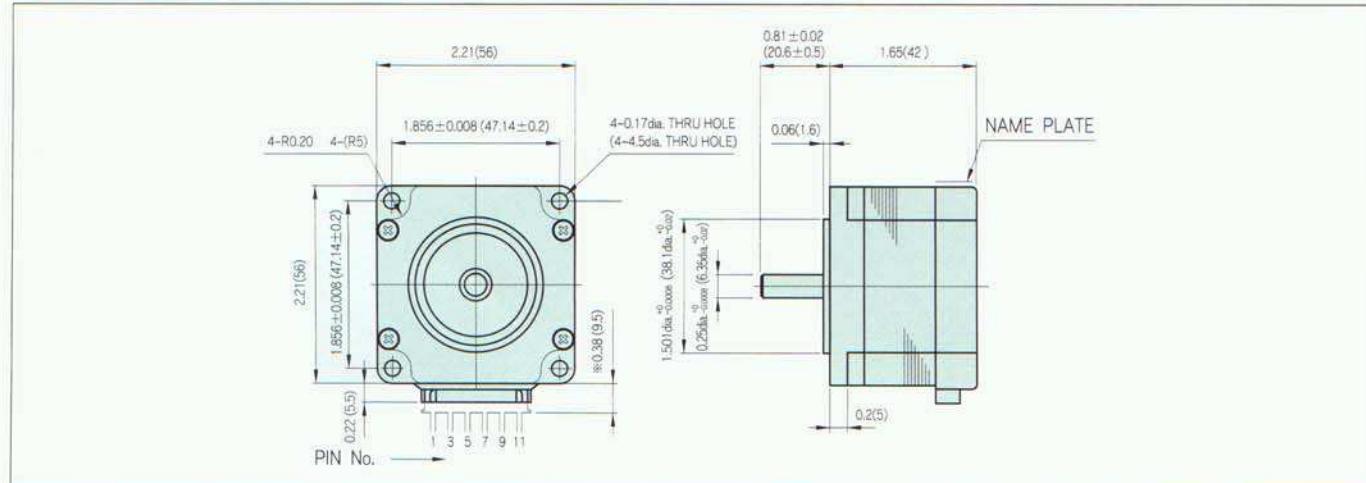
KH56 series 800 type

HIGH TORQUE, LOW VIBRATION AND LOW NOISE

■ STANDARD SPECIFICATIONS

M O D E L	U N I T	K H 5 6 J M 2			
		-801	-802	-803	-851
DRIVE METHOD		UNI-POLAR			BI-POLAR
NUMBER OF PHASES		2			2
STEP ANGLE	deg./step	1.8			1.8
VOLTAGE	V	1.65	2.5	4.8	3.51
CURRENT	A/PHASE	3.0	2.0	1.0	1.3
WINDING RESISTANCE	Ω/PHASE	0.55	1.25	4.8	2.7
INDUCTANCE	mH/PHASE	0.6	1.2	5.9	5.6
HOLDING TORQUE	kgf · cm	4.3	4.3	4.3	5.0
	oz · in	60	60	60	69
DETENT TORQUE	gf · cm	250	250	250	250
	oz · in	3.5	3.5	3.5	3.5
ROTOR INERTIA	g · cm²	140	140	140	140
	oz · in²	0.77	0.77	0.77	0.77
WEIGHTS	g	400	400	400	400
	lb	0.88	0.88	0.88	0.88
INSULATION CLASS		JIS Class E (120°C 248° F) (UL VALUE : CLASS B 130°C 266° F)			
INSULATION RESISTANCE		500VDC 100MΩ min.			
DIELECTRIC STRENGTH		500VAC 50HZ 1min.			
OPERATING TEMP. RANGE	°C	0 to 50			
ALLOWABLE TEMP.RISE	deg.	70			

■ DIMENSIONS unit = inch (mm)

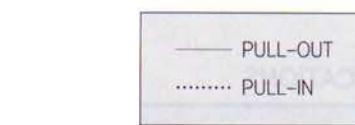
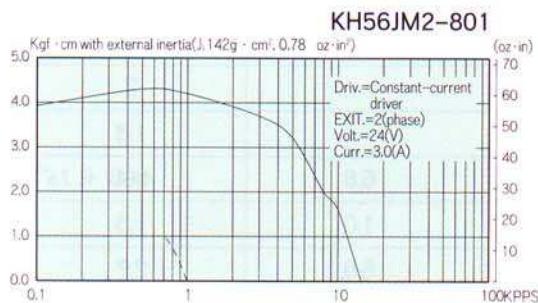




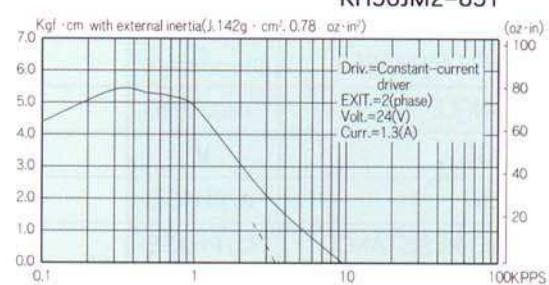
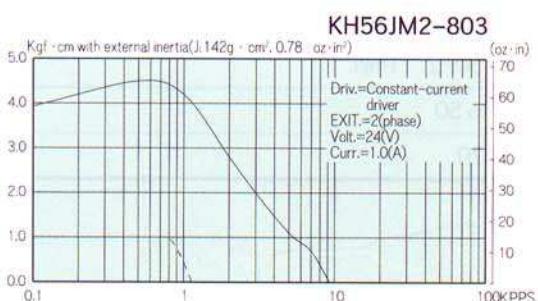
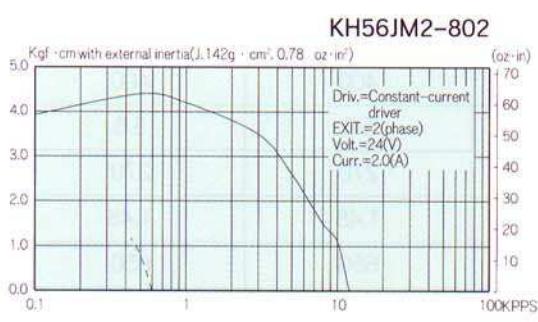
This is our new series 800 Type, which maintains the high torque performance of our conventional 500 Type and lowers vibration and noise.

- High efficiency and high torque have been achieved through our intensive research on magnetic circuitry.
- A unique tooth profile has been employed through the development of a new proven theory to achieve low vibration and low noise.

■ TORQUE CHARACTERISTICS vs. PULSE RATE UNI-POLAR

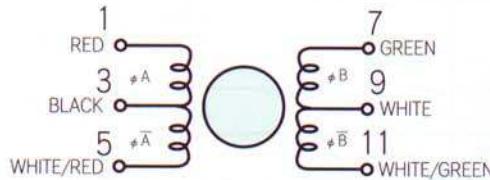


BI-POLAR



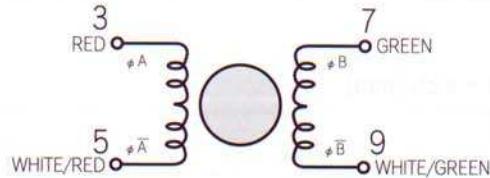
■ CONNECTION DIAGRAMS

UNI-POLAR (-801,2,3)



EXCITATION SEQUENCE				
STEP	1	2	3	4
RED	-	-	-	-
GREEN	-	-	-	-
WHITE/RED	-	-	-	-
WHITE/GREEN	-	-	-	-
BLACK	+	+	+	+
WHITE	+	+	+	+

BI-POLAR (-851)



EXCITATION SEQUENCE				
STEP	1	2	3	4
RED	+	+	-	-
GREEN	-	+	+	-
WHITE/RED	-	-	+	+
WHITE/GREEN	+	-	-	+

■ CONNECTION CABLE TO MOTOR unit = inch (mm)

